IBM DB2 10.1
for Linux, UNIX and Windows

Datenrecovery und hohe Verfügbarkeit Handbuch und Referenz

Aktualisierung: Januar 2013

SC12-4692-01

IBM DB2 10.1
for Linux, UNIX and Windows

Datenrecovery und hohe Verfügbarkeit Handbuch und Referenz

Aktualisierung: Januar 2013

SC12-4692-01

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen in Anhang B, „Bemerkungen", auf Seite 543 gelesen werden.

Impressum

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 10.1 for Linux, UNIX, and Windows, Data Recovery and High Availability Guide and Reference,
IBM Form SC27-3870-01,
herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
TSC Germany
Kst. 2877
Januar 2013

Inhaltsverzeichnis

Zu diesem Handbuch

In der vorliegenden Veröffentlichung '*Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz*' wird beschrieben, wie Sie hohe Verfügbarkeit für Ihre DB2 for Linux, UNIX and Windows-Datenbanklösungen gewährleisten und Datenverlust vermeiden.

'*Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz*' besteht aus zwei Teilen:

- Teil 1 enthält Informationen zur Hochverfügbarkeit und beschreibt Strategien sowie DB2-Datenbankkomponenten und -funktionen, die Sie bei der Aufrechterhaltung der hohen Verfügbarkeit Ihrer Datenbanklösungen unterstützen.
- Teil 2 enthält Informationen zur Datenrecovery und beschreibt, wie Sie die DB2-Backup- und -Restorefunktionen einsetzen können, um Datenverlust zu vermeiden.

Teil 1. Hohe Verfügbarkeit

Die Verfügbarkeit einer Datenbanklösung ist ein Maß dafür, wie erfolgreich die Benutzeranwendungen die erforderlichen Datenbanktasks ausführen.

Können Benutzeranwendungen keine Verbindung zur Datenbank herstellen oder schlagen ihre Transaktionen aufgrund von Fehlern oder Zeitlimitüberschreitungen durch zu hohe Arbeitslast des Systems fehl, ist die Datenbanklösung nicht sehr verfügbar. Können die Benutzeranwendungen aber erfolgreich Verbindungen zur Datenbank herstellen und ihre Aufgaben abarbeiten, weist die Datenbanklösung eine hohe Verfügbarkeit auf.

Der Entwurf einer hoch verfügbaren Datenbanklösung oder die Steigerung der Verfügbarkeit einer bereits vorhandenen Lösung setzt ein Verständnis der Erfordernisse jener Anwendungen voraus, die auf die Datenbank zugreifen. Wenn Sie den größtmöglichen Vorteil aus den Kosten für zusätzlichen Speicherplatz, schnellere Prozessoren oder weitere Softwarelizenzen erzielen möchten, empfiehlt es sich, Ihre Datenbanklösung insbesondere für die wichtigsten Anwendungen in Ihrem Geschäft so verfügbar wie möglich in den Zeiträumen zu gestalten, in denen die Anwendungen am stärksten darauf angewiesen sind.

Ungeplante Betriebsunterbrechungen

Zu den unerwarteten Systemausfällen, die die Verfügbarkeit Ihrer Datenbanklösung für den Benutzer beeinträchtigen könnten, gehören z. B. Stromausfälle, Netzunterbrechungen, Hardwarefehler, Fehler im Betriebssystem oder anderer Software sowie vollständige Systemausfälle im Katastrophenfall. Wenn ein solcher Ausfall zu einem Zeitpunkt auftritt, zu dem der Benutzer erwartet, mit der Datenbank arbeiten zu können, hat eine hoch verfügbare Datenbanklösung die folgenden Aufgaben:

- Abschirmen der Benutzeranwendungen vor dem Ausfall, sodass diese den Ausfall nicht bemerken. Beispiel: DB2 Data Server kann Datenbankclientverbindungen an alternative Datenbankserver weiterleiten, falls ein Datenbankserver ausfällt.
- Reagieren auf den Ausfall, um seine Auswirkungen einzudämmen. Beispiel: Falls auf einer Maschine eines Clusters ein Fehler auftritt, kann der Cluster-Manager diese Maschine aus dem Cluster entfernen, sodass keine weiteren Transaktionen zur Verarbeitung an die fehlerhafte Maschine weitergeleitet werden.
- Recovery nach dem Ausfall durchführen und das System wieder in den normalen Betriebszustand zurückführen. Beispiel: Eine Bereitschaftsdatenbank übernimmt ersatzweise die Datenbankoperationen einer fehlgeschlagenen Primärdatenbank, sodass die fehlgeschlagene Datenbank erneut starten, eine Recovery durchführen und wieder die Primärdatenbankfunktion übernehmen kann.

Diese drei Tasks dürfen mit nur minimalem Effekt auf die Verfügbarkeit der Lösung für die Benutzeranwendungen erfüllt werden.

Geplante Betriebsunterbrechung

Analog müssen in einer hoch verfügbaren Datenbanklösung die Auswirkungen von Verwaltungsaktivitäten auf die Verfügbarkeit der Datenbank für die Benutzeranwendungen minimiert werden.

Beispiel: Falls die Datenbanklösung für eine typische Ladenfront mit Geschäftszeiten zwischen 9.00 und 17.00 Uhr eingesetzt wird, können die Verwaltungsaktivitäten offline und außerhalb der Geschäftszeiten stattfinden, ohne die Verfügbarkeit der Datenbank für die Benutzeranwendungen zu beeinträchtigen. Falls die Datenbanklösung im Online-Bankinggeschäft eingesetzt wird und den Kunden 24 Stunden am Tag für den Zugriff über das Internet zur Verfügung stehen soll, müssen die Verwaltungsaktivitäten online ausgeführt oder für Aktivitätsperioden mit geringer Systemauslastung terminiert werden, damit sie eine möglichst geringe Auswirkung auf die Verfügbarkeit der Datenbank für die Kunden haben.

Die folgenden beiden Faktoren sind gegeneinander abzuwägen, wenn Sie Geschäftsentscheidungen und Entwurfsmöglichkeiten im Hinblick auf die Verfügbarkeit Ihrer Datenbanklösung in Betracht ziehen:

- Die Kosten, die Ihnen in Ihren Geschäftsaktivitäten entstehen, wenn die Datenbank den Kunden nicht zur Verfügung steht.
- Der Aufwand, der durch die Implementierung eines bestimmten Verfügbarkeitsgrades entsteht.

Beispiel: Ein internetbasiertes Geschäft erwirtschaftet pro Stunde, in der seine Datenbanklösung Kunden bedient, den Ertrag X. Eine Hochverfügbarkeitsstrategie, mit der jährlich zehn Stunden Ausfallzeit gespart werden, erbringt dem Geschäft pro Jahr den zusätzlichen Ertrag von zehn mal X. Wenn die Kosten der Implementierung dieser Hochverfügbarkeitsstrategie geringer sind als der erwartete Zusatzertrag ist die Implementierung lohnenswert.

Kapitel 1. Ausfallzeiten

Als Ausfall wird jede Störung bezeichnet, die es der Datenbanklösung unmöglich macht, Benutzeranwendungen zu bedienen. Ausfallzeiten können in zwei Gruppen eingeordnet werden: ungeplante Betriebsunterbrechungen und geplante Betriebsunterbrechungen

Ungeplante Betriebsunterbrechungen

Folgendes kann beispielsweise zu ungeplanten Betriebsunterbrechungen führen:

- Störung einer Systemkomponente, einschließlich Hardware- oder Softwarestörungen.
- Ungültige Aktionen von Verwaltungs- oder Benutzeranwendungen wie beispielsweise das unbeabsichtigte Löschen einer Tabelle, die für geschäftskritische Transaktionen erforderlich ist.
- Schlechte Leistung infolge einer suboptimalen Konfiguration oder unzulänglicher Hardware oder Software.

Geplante Betriebsunterbrechungen

Folgendes gehört zu den geplanten Betriebsunterbrechungen:

- Wartung. Einige Wartungsaktivitäten erfordern eine vollständige Betriebsunterbrechung. Andere Wartungsaktivitäten hingegen können ausgeführt werden, ohne die Datenbank zu stoppen, wirken sich jedoch möglicherweise negativ auf die Leistung aus. Letztere sind die häufigste Art geplanter Betriebsunterbrechungen.
- Upgrade. Das Durchführen eines Upgrades Ihrer Software oder Hardware erfordert bisweilen eine teilweise oder vollständige Betriebsunterbrechung.

Bei der Erörterung der Verfügbarkeit liegt der Schwerpunkt häufig auf Störungsszenarios oder Komponentenfehlern. Beim Entwurf einer stabilen Hochverfügbarkeitslösung müssen jedoch alle diese Ausfallarten berücksichtigt werden.

Ausfallkennzeichen

Bei einem Ausfallkennzeichen handelt es sich um eine Sammlung von Symptomen und Verhaltensweisen, die für einen bestimmten Ausfalltyp charakteristisch sind. Das Kennzeichen eines Ausfalls kann sich im Bereich von temporären Leistungsproblemen, die zu längeren Antwortzeiten für die Endbenutzer führen, bis hin zu einer vollständigen Standortstörung bewegen.

Bei der Erarbeitung von Strategien zur Vermeidung, Minimierung und Behebung von Ausfallzeiten müssen die unterschiedlichen Ausfalltypen und ihre jeweiligen Auswirkungen auf Ihren Geschäftsbetrieb berücksichtigt werden.

Blackout (Gesamtausfall)

Bei einem Ausfall vom Typ Blackout ist das gesamte System für die Endbenutzer nicht verfügbar. Dieser Ausfalltyp kann durch Probleme auf Hardware-, Betriebssystem- oder Datenbankebene verursacht werden. Tritt ein Blackout auf, ist es unbedingt erforderlich, unverzüglich das Ausmaß der Betriebsunterbrechung zu ermitteln. Tritt der Ausfall ausschließlich auf Da-

tenbankebene auf? Tritt der Ausfall auf Instanzebene auf? Oder tritt er auf Betriebssystem- oder Hardwareebene auf?

Brownout (Leistungsprobleme)

Bei einem Ausfall vom Typ Brownout ist die Systemleistung so langsam, dass die Endbenutzer ihre Arbeit nicht mehr effektiv erledigen können. Das System insgesamt kann durchaus betriebsbereit sein, funktioniert aus Sicht der Endbenutzer im Wesentlichen jedoch nicht wie erwartet. Dieser Ausfalltyp kann während Systemwartungszeiten oder Perioden mit hoher Auslastung auftreten. Bei derartigen Ausfällen sind die CPU- und Speicherkapazität in der Regel fast erschöpft. Nicht optimal konfigurierte oder überlastete Server tragen häufig zu einem Brownout bei.

Häufigkeit und Dauer von Ausfällen

Bei der Erörterung der Datenbankverfügbarkeit liegt der Schwerpunkt häufig auf der Gesamtsumme oder dem Prozentsatz der Ausfallzeit (oder umgekehrt auf dem Zeitraum, in dem das Datenbanksystem verfügbar ist) innerhalb einer bestimmten Periode. Die Häufigkeit und die Dauer geplanter und ungeplanter Betriebsunterbrechungen unterscheiden sich jedoch deutlich in den Auswirkungen, die sie auf Ihren Geschäftsbetrieb haben.

Angenommen, Sie müssen einige Upgrades an Ihrem Datenbanksystem durchführen, die insgesamt sieben Stunden in Anspruch nehmen, und Sie haben die Wahl, das Datenbanksystem entweder über sieben Tage jeweils eine Stunde pro Tag in Zeiten geringer Benutzeraktivität abzuschalten oder das Datenbanksystem sieben Stunden hintereinander während der größten Auslastung am Tag mit der größten Geschäftstätigkeit abzuschalten. Hier ist offensichtlich, dass mehrere kurze Ausfälle für Ihr Unternehmen weniger kostspielig und schädlich sind als der einmalige Ausfall für sieben Stunden. Oder nehmen Sie an, es treten bei Ihnen immer wieder Netzstörungen auf, die jede Woche insgesamt einige Minuten dauern, sodass eine kleine Anzahl an Transaktionen regelmäßig fehlschlägt. Diese sehr kurzen Ausfälle können letztlich hohe Umsatzeinbußen und einen nicht wiedergutzumachenden Vertrauensverlust seitens Ihrer Kunden nach sich ziehen, was sich wiederum in noch größeren zukünftigen Umsatzverlusten niederschlagen kann.

Daher sollten Sie sich nicht ausschließlich auf die Gesamtausfallszeit (bzw. Gesamtverfügbarkeit) konzentrieren. Wägen Sie die Kosten von wenigen längeren Ausfällen und die Kosten von mehreren kürzeren Ausfällen gegeneinander ab, wenn Sie Wartungsaktivitäten in Betracht ziehen oder auf eine ungeplante Betriebsunterbrechung reagieren. Während eines akuten Ausfalls kann es schwierig sein, eine solche Entscheidung zu treffen. Daher sollten Sie eine Formel oder Methode entwickeln, mit der die Kosten der verschiedenen Ausfalltypen mit ihren unterschiedlichen Kennzeichen für Ihr Unternehmen berechnet werden können, damit Sie im Bedarfsfall die beste Wahl treffen.

Mehrere und kaskadierende Ausfälle

Wenn Sie Ihre Datenbanklösung entwerfen, um Ausfälle zu vermeiden, zu minimieren oder zu beheben, müssen Sie auch die Möglichkeit bedenken, dass mehrere Komponenten gleichzeitig ausfallen können und dass sogar die Störung einer Komponente den Ausfall einer anderen Komponente verursachen kann (wie eine Kaskade).

Ausfallkosten

Die Kosten eines Ausfalls sind von Betrieb zu Betrieb unterschiedlich. Ein bewährtes Verfahren für jedes Unternehmen besteht darin, die Kosten eines Ausfalls im Hinblick auf die zum Erreichen der Unternehmensziele kritischen Geschäftsprozesse zu analysieren. Anhand der Ergebnisse dieser Analyse kann dann ein entsprechender Plan für die Wiederherstellung des Systems nach einem Ausfall entwickelt werden.

Dieser Plan umfasst auch eine nach Priorität geordnete Wiederherstellungsreihenfolge, wenn mehrere Prozesse betroffen sind.

Ausfallkosten

Sie können die Kosten schätzen, die Ihrem Unternehmen entstehen, wenn Ihre Kunden mit einem Datenbanksystem konfrontiert werden, das nicht für die Verarbeitung von Kundentransaktionen verfügbar ist. So können Sie beispielsweise die durchschnittlichen Kosten aufgrund von Umsatzverlusten für jede Stunde oder Minute berechnen, in der Ihr Datenbanksystem nicht verfügbar ist. Die Berechnung der potenziellen Umsatzeinbußen aufgrund gesunkenen Kundenvertrauens ist viel schwieriger. Dennoch sollten diese Kosten berücksichtigt werden, wenn Sie beurteilen wollen, welches Maß an Verfügbarkeit Ihr Unternehmen benötigt.

Darüber hinaus sollten Sie auch die Kosten einbeziehen, die entstehen, wenn interne Datenbanksysteme nicht für Ihre eigenen Geschäftsprozesse verfügbar sind. Sogar einfache Anwendungen wie E-Mail oder Kalender, die eine Stunde lang ausfallen, können den Geschäftsbetrieb zum Erliegen bringen, da die Mitarbeiter nicht mehr in der Lage sind, ihrer Arbeit nachzugehen.

Ausfalltoleranz

Wie viel Ausfallzeit toleriert werden kann, ist von Betrieb zu Betrieb unterschiedlich. Ein bewährtes Verfahren für jedes Unternehmen besteht darin, die Auswirkungen eines Ausfalls auf die zum Erreichen der Unternehmensziele kritischen Geschäftsprozesse zu analysieren. Anhand der Ergebnisse dieser Analyse kann dann ein entsprechender Plan für die Wiederherstellung des Systems nach einem Ausfall entwickelt werden.

Dieser Plan umfasst auch die Reihenfolge bei der Wiederherstellung, wenn mehrere Prozesse betroffen sind.

Ausfalltoleranz

Ein wesentlicher Faktor bei der Ermittlung der Anforderungen an die Verfügbarkeit besteht darin, sich zu fragen, wie tolerant das Unternehmen oder ein bestimmtes System im Unternehmen im Falle des Auftretens eines Ausfalls ist. Beispiel: Ein Restaurant, das seine Website primär betreibt, um die Speisekarte zu veröffentlichen, hat bei einem gelegentlichen Serverausfall kaum Umsatzeinbußen zu befürchten. Hingegen kann der Ausfall eines Servers an einer Börse, der zum Aufzeichnen von Transaktionen dient, katastrophale Folgen haben. Daher wäre der Einsatz großer Ressourcenmengen zur Gewährleistung einer Verfügbarkeit des Restaurantservers von 99,99 Prozent nicht kosteneffizient, im Falle der Börse aber sehr wohl.

Bei der Erörterung der Toleranz sollten zwei Konzepte berücksichtigt werden: die Zeit bis zur Recovery sowie der Punkt der Recovery.

Bei der Zeit bis zur Recovery geht es darum, wie lange es dauert, bis ein Geschäftsprozess oder System wieder online ist.

Der Punkt der Recovery ist der Zustand in der Vergangenheit, in dem der Geschäftsprozess oder das System bei der Recovery wiederhergestellt wird. Im Hinblick auf die Datenbank müssten im Rahmen eines Recovery-Plans die Vorteile einer schnellen Wiederherstellung, bei der möglicherweise einige Transaktionen verloren gehen, gegenüber denen einer vollständigen Wiederherstellung abgewogen werden, bei der zwar kein Datenverlust entsteht, die aber länger dauert.

Recovery und Vermeidungsstrategien

Wenn beim Einkauf und Systemaufbau Entscheidungen hinsichtlich der Verfügbarkeit anstehen, ist man häufig versucht, lange Listen von Komponenten und Technologien mit hoher Verfügbarkeit durchzugehen. Um dafür zu sorgen, dass ein System hoch verfügbar ist und bleibt, kommt es jedoch nicht nur auf den Kauf der richtigen Technologie an. Ebenso wichtig ist es, die richtigen Entscheidungen bei Design und Konfiguration zu treffen sowie tragfähige Verwaltungsprozesse und Notfallpläne zu entwickeln und umzusetzen.

Die umfassendste Verfügbarkeit Ihrer Investition erhalten Sie, wenn Sie zunächst diejenigen Strategien für hohe Verfügbarkeit ermitteln, die für Ihre Geschäftsanforderungen am besten geeignet sind. Anschließend können Sie diese Strategien dann mithilfe der am besten geeigneten Technologie umsetzen.

Wenn Sie Ihre Datenbanklösung für hohe Verfügbarkeit entwickeln oder konfigurieren, überlegen Sie, wie Sie Ausfallzeiten vermeiden oder deren Auswirkungen minimieren und Ihr System schnell wiederherstellen können.

Ausfallzeiten vermeiden

Ausfallzeiten sollten so weit wie möglich von vornherein vermieden werden. Beispiel: Entfernen Sie Single Points of Failure, um ungeplante Betriebsunterbrechungen zu vermeiden, oder prüfen Sie Methoden, mit denen Wartungsaktivitäten online durchgeführt werden können, um geplante Betriebsunterbrechungen zu vermeiden. Überwachen Sie Ihr Datenbanksystem, um Tendenzen beim Systemverhalten zu ermitteln, die auf Probleme hinweisen, damit Sie diese Probleme beheben können, bevor es zu einem Ausfall kommt.

Auswirkungen von Ausfällen minimieren

Sie können Ihre Datenbanklösung entsprechend entwickeln und konfigurieren, um die Auswirkungen geplanter und ungeplanter Betriebsunterbrechungen zu minimieren. Verteilen Sie Ihre Datenbanklösung beispielsweise so, dass Komponenten und Funktionen lokal vorhanden sind, damit bestimmte Benutzeranwendungen ihre Transaktionen auch dann weiter verarbeiten können, wenn eine Komponente offline ist.

Schnelle Recovery bei ungeplanten Betriebsunterbrechungen

Erstellen Sie einen Recovery-Plan: Entwickeln Sie klare und gut dokumentierte Prozesse, die von Administratoren schnell und ohne Schwierigkeiten befolgt werden können, falls eine ungeplante Betriebsunterbrechung eintritt. Verfassen Sie leicht verständliche Dokumente zur Systemarchitektur, die alle beteiligten Komponenten beschreiben. Halten Sie die Servicevereinbarungen und zugehörigen Kontaktdaten übersichtlich geordnet und griffbereit. Während eine schnelle Recovery zwar von größter Wichtigkeit ist, muss auch bekannt sein, welche Diagnoseinformationen zu erfassen sind, damit die Ursache für den Ausfall ermittelt werden kann und um zukünftige Ausfälle dieser Art zu vermeiden.

Kapitel 2. Strategien für hohe Verfügbarkeit

Einem Benutzer ist es egal, warum seine Datenbankanforderung fehlgeschlagen ist. Ob bei einer Transaktion das Zeitlimit wegen schlechten Leistungsverhaltens überschritten wurde, eine Komponente innerhalb der Lösung fehlgeschlagen ist oder ein Administrator die Datenbank in den Offlinestatus wegen Instandhaltungsarbeiten versetzt hat, das Ergebnis ist für den Benutzer das gleiche.

Die Datenbank steht nicht für die Verarbeitung von Anforderungen zur Verfügung.

Die folgenden Strategien tragen zur Verbesserung der Verfügbarkeit Ihrer Datenbanklösung bei:

Redundanz
Sie verfügen über Zweitkopien aller Komponenten in Ihrer Datenbanklösung, die im Falle einer Störung oder eines Fehlers Workload übernehmen können.

Systemüberwachung
Es werden Statistikdaten über die Komponenten in Ihrer Datenbanklösung erfasst, um den Lastausgleich zu erleichtern bzw. um festzustellen, ob Komponenten eventuell ausgefallen sind.

Lastausgleich
Ein Teil der Verarbeitungsprozesse einer überlasteten Komponente in Ihrer Datenbanklösung wird an eine andere Komponente in Ihrer Datenbanklösung übertragen, die zum aktuellen Zeitpunkt weniger Arbeitslast aufweist.

Funktionsübernahme
Alle Verarbeitungsprozesse einer fehlgeschlagenen Komponente in Ihrer Datenbanklösung werden auf eine sekundäre Komponente übertragen.

Leistung maximieren
Die Möglichkeit, dass Transaktionen sehr viel Zeit zum Beenden benötigen oder das Zeitlimit überschreiten, wird verringert.

Beeinträchtigungen durch Verwaltungsaktivitäten minimieren
Terminierung automatisierter und manueller Verwaltungsaktivitäten, um die Benutzeranwendungen so wenig wie möglich zu beeinträchtigen.

Hohe Verfügbarkeit durch Redundanz

Eine wichtige Strategie zur Aufrechterhaltung einer hohen Verfügbarkeit ist das Vorhandensein redundanter Komponenten. Für den Fall, dass eine Komponente ausfällt, kann eine sekundäre oder eine Backup-Kopie der ausgefallenen Komponente die Operationen übernehmen, wodurch die Datenbank den Benutzeranwendungen weiterhin zur Verfügung stehen kann.

Wenn eine dieser Komponenten des Systems nicht redundant ist, könnte sich die betreffende Komponente als SPoF (Single Point of Failure) für das gesamte System erweisen.

Redundanz ist im Systemaufbau in folgenden Bereichen üblich:

- Nicht unterbrochene oder Notstromversorgung
- Mehrere Netzverbindungsleitungen zwischen den einzelnen Komponenten

- Bonding oder Lastausgleich für Netzkarten
- Mehrere Festplattenlaufwerke in einem redundanten Array
- CPU-Cluster

Wenn eine dieser Komponenten des Systems nicht redundant ist, könnte sich die betreffende Komponente als SPoF für das gesamte System erweisen.

Sie können Redundanz auf Datenbankebene erstellen, indem Sie zwei Datenbanken betreiben: eine primäre Datenbank, die im Normalfall alle oder die Mehrzahl der Verarbeitungsprozesse der Anwendungen verarbeitet, und eine sekundäre Datenbank, die die Verarbeitungsprozess übernehmen kann, wenn die primäre Datenbank ausfällt. In einer DB2-HADR-Umgebung (High Availability Disaster Recovery) wird die sekundäre Datenbank als "Bereitschaftsdatenbank" bezeichnet.

Für DB2 Connect-Clients stellen Funktionen für den Sysplex-Lastausgleich auf DB2 for z/OS-Servern eine hohe Verfügbarkeit für Clientanwendungen bereit, die eine direkte Verbindung zu einer Gruppe mit gemeinsamer Datennutzung herstellen. Diese Funktionen umfassen den Lastausgleich und eine integrierte automatische Clientweiterleitung. Diese Unterstützung ist für Anwendungen verfügbar, die Java-Clients (JDBC, SQLJ oder pureQuery) oder andere Clients (ODBC, CLI, .NET, OLE DB, PHP, Ruby oder eingebettetes SQL) verwenden.

Hohe Verfügbarkeit durch Funktionsübernahme (Failover)

Die Funktionsübernahme (Failover) ist die Übertragung von Workload von einem primären System auf ein sekundäres System, wenn das primäre System ausfällt. Wenn Workload auf diese Weise übertragen wurde, lautet die Bezeichnung dafür auch, dass das sekundäre System die Verarbeitungsprozesse (Workload) des fehlgeschlagenen primären Systems "übernommen" hat.

Beispiel 1

Schlägt in einer Clusterumgebung eine der Maschinen im Cluster fehl, kann die Software für das Cluster-Management Prozesse, die auf der fehlgeschlagenen Maschine ausgeführt wurden, auf eine andere Maschine im Cluster versetzen.

Beispiel 2

Ist in einer Datenbanklösung mit mehreren IBM® Data Server-Servern eine Datenbank nicht mehr verfügbar, kann der Datenbankmanager Anwendungen, die mit dem nicht mehr verfügbaren Datenbankserver verbunden waren, an einen sekundären Datenbankserver weiterleiten.

Die beiden häufigsten im Markt vorhandenen Funktionsübernahmestrategien sind der Bereitschaftsmodus und die gegenseitige Übernahme:

Bereitschaftsmodus (Idle Standby)

In dieser Konfiguration verarbeitet ein primäres System die gesamte Workload, während sich ein sekundäres System im Bereitschaftsmodus befindet und bereit ist, diese Workload zu übernehmen, falls das primäre System ausfällt. In einer HADR-Konfiguration können bis zu drei Bereitschaftsdatenbanken vorhanden sein. Jede dieser Bereitschaftsdatenbanken kann für die Ausführung von schreibgeschützten Workloads konfiguriert werden.

Gegenseitige Übernahme (Mutual Takeover)

In dieser Konfiguration gibt es mehrere Systeme, die jeweils das designierte sekundäre System für ein anderes System sind. Wenn ein System fehlschlägt, wirkt sich dies negativ auf die Gesamtleistung aus, da das sekundäre System des ausgefallenen Systems nicht nur weiter seine eigene Workload, sondern auch die des ausgefallenen Systems verarbeiten muss.

Hohe Verfügbarkeit durch Clustering

Ein Cluster ist eine Gruppe verbundener Systeme, die wie ein einziges System zusammenarbeiten. Wenn eine Maschine in einem Cluster fehlschlägt, transferiert eine Clusterverwaltungssoftware die Verarbeitungsprozesse des fehlgeschlagenen Systems auf andere Systeme.

Überwachungssignalfunktion

Die Software für die Funktionsübernahme kann zur Feststellung eines Fehlers auf einem System des Clusters zwischen den Systemen die Überwachungssignalfunktion oder Keepalive-Pakete verwenden, um die Verfügbarkeit zu bestätigen. Die Überwachungssignalfunktion bezieht Systemservices ein, die die konstante Kommunikation zwischen allen Systemen eines Clusters verwalten. Wenn ein Überwachungssignal nicht ermittelt werden kann, wird die Funktionsübernahme durch ein Backup-System eingeleitet.

IP-Adressübernahme

Bei einer Störung auf einem System im Cluster können die Cluster-Manager Workload von einem System auf ein anderes übertragen, indem die IP-Adresse von einem System auf das andere übertragen wird. Dieser Vorgang wird als "IP-Adressübernahme" oder "IP-Übernahme" bezeichnet. Die Übertragung ist für Clientanwendungen nicht sichtbar, die auch weiterhin die ursprüngliche IP-Adresse verwenden und nicht bemerken, dass die physische Maschine, zu der die IP-Adresse gehört, sich geändert hat.

DB2 High Availability Feature ermöglicht die Integration zwischen dem IBM DB2-Server und der Cluster-Management-Software.

Datenbankprotokollierung

Die Datenbankprotokollierung ist ein wichtiger Teil Ihres Lösungsentwurfs für eine Datenbank mit hoher Verfügbarkeit, da man anhand von Datenbankprotokollen nach einem Fehler eine Recovery durchführen sowie primäre und sekundäre Datenbanken synchronisieren kann.

Jeder Datenbank sind Protokolldateien zugeordnet. In diesen Protokolldateien werden die Datenbankänderungen aufgezeichnet. Wenn eine Datenbank über den Stand des letzten vollständigen Offline-Backups hinaus wiederhergestellt werden muss, sind Protokolle erforderlich, um die Datenbank bis zu dem Zeitpunkt des Fehlers aktualisierend wiederherstellen zu können.

Es werden zwei Typen der Datenbankprotokollierung unterstützt: *Umlaufprotokollierung* und *Archivprotokollierung*. Jeder Typ stellt eine andere Stufe der Wiederherstellbarkeit bereit:

- „Umlaufprotokollierung” auf Seite 12
- „Archivprotokollierung” auf Seite 13

Der Vorteil der Archivprotokollierung liegt darin, dass von der aktualisierenden Recovery sowohl archivierte als auch aktive Protokolldateien verwendet werden können, um eine Datenbank entweder bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt wiederherzustellen. Die Archivprotokolldateien können zur Recovery von Änderungen verwendet werden, die nach dem Backup vorgenommen wurden. Bei der Umlaufprotokollierung hingegen ist eine Recovery nur bis zum Zeitpunkt des letzten Backups möglich, alle später vorgenommenen Änderungen gehen verloren.

Umlaufprotokollierung

Umlaufprotokollierung ist die Standardprotokollierung, wenn eine neue Datenbank erstellt wird. (Die Datenbankkonfigurationsparameter **logarchmeth1** und **logarchmeth2** sind auf OFF gesetzt.)

Bei dieser Art der Protokollierung sind nur vollständige Offline-Backups der Datenbank zulässig. Die Datenbank muss offline sein (d. h. für Benutzer nicht zugänglich), wenn ein Gesamtbackup erstellt wird.

Wie der Name bereits andeutet, verwendet die Umlaufprotokollierung einen *Ring* von Onlineprotokollen, die eine Recovery nach Transaktionsfehlern oder Systemabstürzen ermöglichen. Die Protokolle werden nur so lange verwendet und behalten, wie es erforderlich ist, um die Integrität der aktuellen Transaktionen zu gewährleisten. Bei der Umlaufprotokollierung ist es nicht möglich, eine Datenbank durch frühere Transaktionen, die seit dem letzten Gesamtbackup durchgeführt wurden, aktualisierend wiederherzustellen. Alle Änderungen, die seit dem letzten Backup vorgenommen wurden, gehen verloren. Da dieser Typ des Restores die Daten auf dem Stand zum Zeitpunkt des Gesamtbackups wiederherstellt, wird er als *Versionsrecovery* bezeichnet.

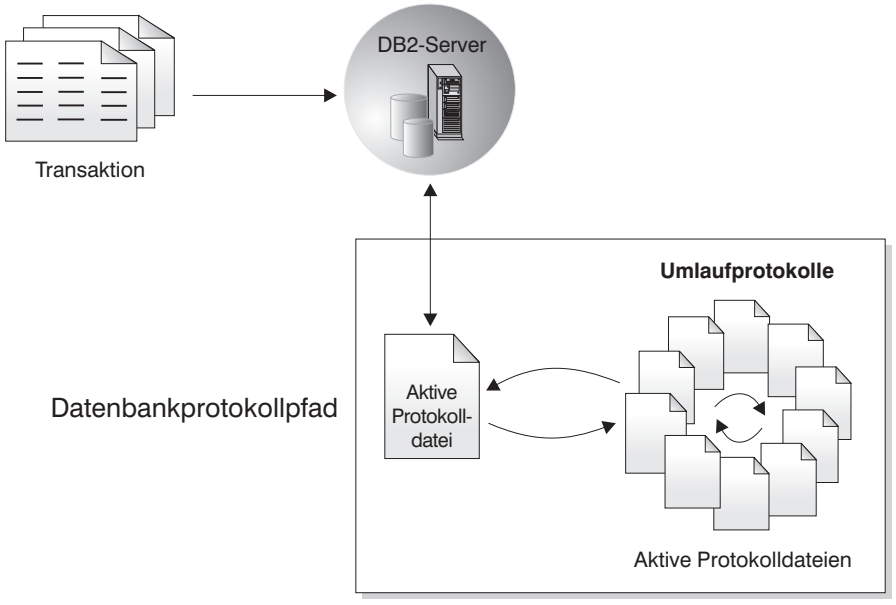


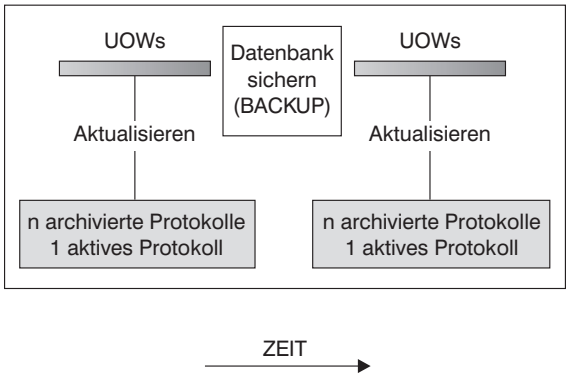
Abbildung 1. Umlaufprotokollierung

Aktive Protokolldateien werden bei der Recovery nach einem Systemabsturz verwendet, um zu verhindern, dass eine Datenbank nach einer Störung (Stromausfall oder Anwendungsfehler) in einem inkonsistenten Status zurückbleibt. Aktive Protokolldateien befinden sich im Verzeichnispfad der Datenbankprotokolle.

Archivprotokollierung

Archivprotokollierung wird insbesondere für die aktualisierende Recovery verwendet. Archivierte Protokolle sind Protokolldateien, die aus dem aktuellen Protokollpfad oder dem Pfad für Protokollspiegelung an eine andere Position kopiert werden.

Sie können den Datenbankkonfigurationsparameter **logarchmeth1** und/oder den Datenbankkonfigurationsparameter **logarchmeth2** verwenden, um sich selbst oder dem Datenbankmanager die Verwaltung des Protokollarchivierungsprozesses zu ermöglichen.



Protokolle werden zwischen Backups verwendet, um die Änderungen an den Datenbanken zu protokollieren.

Abbildung 2. Aktive und archivierte Datenbankprotokolle bei aktualisierender Recovery. Im Falle einer lange laufenden Transaktion kann es mehr als ein aktives Protokoll geben.

Das Erstellen von Online-Backups wird nur dann unterstützt, wenn die Datenbank für die Archivprotokollierung konfiguriert ist. Während eines Online-Backups werden alle Aktivitäten an der Datenbank protokolliert. Nachdem ein Online-Backup abgeschlossen ist, erzwingt der Datenbankmanager das Schließen des momentan aktiven Protokolls, das daraufhin archiviert wird. Hierdurch wird sichergestellt, dass Ihrem Online-Backup alle zur Recovery benötigten archivierten Protokolle zur Verfügung stehen. Wenn ein Online-Backup wiederhergestellt wird, muss mithilfe der Protokolle die Datenbank mindestens bis zu dem Zeitpunkt aktualisierend wiederhergestellt werden, zu dem das Backup abgeschlossen wurde. Zur Vereinfachung dieser Operation müssen archivierte Protokolle bei der Wiederherstellung der Datenbank verfügbar gemacht werden.

Sie können die Datenbankkonfigurationsparameter **logarchmeth1** und **logarchmeth2** verwenden, um anzugeben, wo die archivierten Protokolle gespeichert werden sollen. Sie können den Parameter **logarchmeth1** verwenden, um die Protokolldateien aus dem Pfad für aktive Protokolldateien zu archivieren, der im Konfigurationsparameter **logpath** angegeben wurde. Sie können den Parameter **logarchmeth2** verwenden, um zusätzliche Kopien der Protokolldateien aus dem Pfad für aktive Protokolldateien unter einer zweiten Speicherposition zu archivieren. Wenn die Protokollspiegelung nicht konfiguriert wird, dann werden die zusätzlichen Kopien demselben Protokollpfad entnommen, der vom Parameter **logarchmeth1** verwendet wird. Wenn Sie die Protokollspiegelung konfigurieren, archiviert der Konfigurationsparameter **logarchmeth2** mit dem Konfigurationsparameter **mirrorlogpath** Protokolldateien stattdessen aus dem Protokollspiegelungspfad. Dies kann die Ausfall-

sicherheit während einer aktualisierenden Recovery verbessern. Der Parameter **newlogpath** wirken sich außerdem auf die Speicherposition aus, an der die aktiven Protokolle gespeichert werden.

In bestimmten Fällen können archivierte Protokolldateien komprimiert werden, um die Speicherkosten in Verbindung mit diesen Dateien zu senken. Wenn die Konfigurationsparameter **logarchmeth1** und **logarchmeth2** auf DISK, TSM oder VENDOR gesetzt sind, können Sie die Komprimierung archivierter Protokolldateien aktivieren, indem Sie die Konfigurationsparameter **logarchcompr1** und **logarchcompr2** auf ON setzen. Wenn **logarchcompr1** und **logarchcompr2** dynamisch gesetzt werden, werden bereits archivierte Protokolldateien nicht komprimiert.

Wenn Sie die Option LOGRETAIN verwenden, um einen Wert anzugeben, den Sie bei der Verwaltung der aktiven Protokolle verwenden möchten, benennt der Datenbankmanager die Protokolldateien aus dem Pfad für aktive Protokolldateien um, nachdem diese Dateien archiviert wurden und nicht mehr für die Recovery nach einem Systemabsturz benötigt werden. Wenn Sie die Endlosprotokollierung aktivieren, wird für weitere aktive Protokolldateien zusätzlicher Speicherbereich benötigt. Der Datenbankserver benennt die Protokolldateien daher nach ihrer Archivierung um.

Protokollsteuerdateien

Wenn eine Datenbank nach einem Fehler erneut startet, wendet der Datenbankmanager in Protokolldateien gespeicherte Transaktionsinformationen an, um die Datenbank wieder in einen konsistenten Status zu versetzen. Für die Feststellung, welche Datensätze aus den Protokolldateien auf die Datenbank angewendet werden müssen, verwendet der Datenbankmanager Informationen, die in Protokollsteuerdateien aufgezeichnet sind.

Ausfallsicherheit der Datenbank durch Redundanz

Der Datenbankmanager pflegt jeweils zwei Kopien der Protokollsteuerdateien eines Members (SQLOGCTL.LFH.1 und SQLOGCTL.LFH.2) und jeweils zwei Kopien der globalen Protokollsteuerdatei (SQLOGCTL.GLFH.1 und SQLOGCTL.GLFH.2), sodass im Falle der Beschädigung einer Kopie der Datenbankmanager noch die andere Kopie verwenden kann.

Leistungsaspekte

Das Anwenden der in den Protokollsteuerdateien enthaltenen Transaktionsinformationen trägt zu dem Systemaufwand beim Neustart einer Datenbank nach einem Fehler bei. Sie können die Frequenz konfigurieren, mit der der Datenbankmanager Pufferpoolseiten auf Platte schreibt, um die Anzahl der Protokollsätze zu reduzieren, die während der Recovery nach einem Systemabsturz verarbeitet werden müssen. Verwenden Sie dazu den Konfigurationsparameter „softmax - Vor bedingtem Prüfpunkt zu schreibende Protokollsätze" in Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen.

Kapitel 3. Hohe Verfügbarkeit mit DB2-Server

Der IBM DB2-Server enthält Funktionalitäten, die zahlreiche Strategien für eine hohe Verfügbarkeit unterstützen.

Automatische Clientweiterleitung - Übersicht

Die automatische Clientweiterleitung ist eine Funktion des IBM DB2-Servers, die Clientanwendungen von einem fehlgeschlagenen Server an einen alternativen Server weiterleitet, sodass die Anwendungen mit nur minimaler Unterbrechung weiter funktionieren können. Die automatische Clientweiterleitung kann nur erfolgen, wenn vor dem Ausfall oder Fehler ein alternativer Server festgelegt worden ist.

Tabelle 1 listet die relevanten Themen in den jeweiligen Kategorien auf.

Tabelle 1. Übersicht über die Informationen zur automatischen Clientweiterleitung

Kategorie	Zugehörige Themen
Allgemeine Informationen	• „Automatische Clientweiterleitung - Einschränkungen" auf Seite 30 • „Automatische Clientweiterleitung - Beschreibung und Einrichtung" auf Seite 25 • „Automatische Clientweiterleitung - Beschreibung und Einrichtung (DB2 Connect)" im Handbuch *DB2 Connect - Installation und Konfiguration von DB2 Connect-Servern*
Konfiguration	• „Angeben eines alternativen Servers für die automatische Clientweiterleitung" auf Seite 29 • „Konfiguration der Unterstützung für hohe Verfügbarkeit bei DB2 Database for Linux, UNIX and Windows für Java-Clients" im Handbuch Developing Java Applications
Beispiele	• „Automatische Clientweiterleitung - Beispiele" auf Seite 260
Interaktion mit anderen DB2-Funktionen	• „Automatische Clientweiterleitung und HADR (High Availability Disaster Recovery) konfigurieren" auf Seite 41 • „Konfiguration der Unterstützung für hohe Verfügbarkeit bei DB2 Database for Linux, UNIX and Windows für Java-Clients" im Handbuch Developing Java Applications
Fehlerbehebung	• „Konfigurieren der automatischen Clientweiterleitung für die Distributortechnologie für die Clientverbindung" auf Seite 28

Anmerkung: Die automatische Clientweiterleitung für DB2 für z/OS Sysplex steht auch in IBM Data Server Clients und nicht auf Java basierten IBM Data Server Drivers zur Verfügung. Durch diese Unterstützung können Anwendungen, die auf ein DB2 für z/OS-Sysplex zugreifen, die vom Client bereitgestellten Funktionen der automatischen Clientweiterleitung verwenden und müssen hierfür keinen DB2 Connect-Server verwenden. Weitere Informationen zu dieser Funktion enthält der Abschnitt zur automatischen Clientweiterleitung (clientseitig) im *DB2 Information Center*.

DB2-Fehlermonitorfunktion für Linux und UNIX

Die nur auf UNIX-basierten Systemen verfügbaren DB2-Fehlermonitorfunktionalitäten sorgen dafür, dass die IBM DB2-Serverdatenbanken betriebsbereit sind, u. a. durch das Überwachen von DB2-Datenbankmanagerinstanzen und das erneute Starten etwaiger Instanzen, die vorher erstellt wurden.

Der FMC (Fault Monitor Coordinator - Fehlermonitorkoordinator) ist der Prozess der Fehlermonitorfunktion, der während der UNIX-Startreihenfolge gestartet wird. Der Dämon `init` startet den FMC und startet ihn erneut, falls der FMC abnormal beendet wird. Der FMC startet für jede DB2-Instanz einen Fehlermonitor. Jeder Fehlermonitor wird als Dämonprozess ausgeführt und hat dieselben Benutzerzugriffsrechte wie die DB2-Instanz.

Nach dem Start eines Fehlermonitors wird er überwacht, um sicherzustellen, dass er nicht vorzeitig beendet wird. Wenn ein Fehlermonitor fehlschlägt, wird er vom FMC erneut gestartet. Jeder Fehlermonitor überwacht wiederum eine DB2-Instanz. Wenn die DB2-Instanz vorzeitig beendet wird, wird sie vom Fehlermonitor neu gestartet. Die Fehlermonitorfunktion kann nur durch Absetzen des Befehls **db2stop** inaktiviert werden. Wenn eine DB2-Instanz auf andere Weise beendet wird, wird sie von der Fehlermonitorfunktion neu gestartet.

DB2-Fehlermonitor - Einschränkungen

Wenn Sie ein Clusteringprodukt mit hoher Verfügbarkeit wie z. B. IBM Tivoli System Automation for Multiplatforms (SA MP) oder IBM PowerHA SystemMirror for AIX verwenden, muss die Fehlermonitorfunktion inaktiviert werden, da der Start und das Herunterfahren der Instanz von dem Clusteringprodukt gesteuert werden.

Unterschiede zwischen dem DB2-Fehlermonitor und dem DB2-Diagnosemonitor

Der Diagnosemonitor und der Fehlermonitor sind Tools, die in einer einzigen Datenbankinstanz aktiv sind. Der Diagnosemonitor verwendet *Diagnoseanzeiger*, um bestimmte Leistungsaspekte des Datenbankmanagers oder von Datenbanken auf ihren ordnungsgemäßen Betrieb hin zu bewerten. Ein Diagnoseanzeiger misst den Status eines Aspekts einer bestimmten Klasse von Datenbankobjekten, wie zum Beispiel eines Tabellenbereichs. Diagnoseanzeiger können anhand spezieller Kriterien bewertet werden, um den Status der betreffenden Klasse von Datenbankobjekt festzustellen. Außerdem können Diagnoseanzeiger Alerts generieren, um Sie zu benachrichtigen, wenn ein Bezugswert einen Schwellenwert überschreitet oder angibt, dass ein Datenbankobjekt sich in einem nicht normalen Zustand befindet.

Demgegenüber ist der Fehlermonitor nur dafür zuständig, die von ihm überwachte Instanz betriebsbereit und aktiv zu halten. Wenn die überwachte DB2-Instanz unerwartet beendet wird, startet der Fehlermonitor die Instanz neu. Die Fehlermonitorfunktion steht unter Windows nicht zur Verfügung.

High Availability Disaster Recovery (HADR)

Die Funktion High Availability Disaster Recovery (HADR) stellt eine Hochverfügbarkeitslösung für Teil- oder Komplettausfälle von Standorten bereit. HADR schützt durch die Replikation von Datenänderungen aus einer Quellendatenbank, der sogenannten *Primärdatenbank*, in mindestens einer Zieldatenbank, der sogenannten *Bereitschaftsdatenbank*, vor Datenverlusten.

Ein Teilausfall eines Standorts kann durch einen Ausfall von Hardware, Software (DB2-Datenbanksystem oder Betriebssystem) oder des Netzes verursacht werden. Ohne HADR muss bei einem Teilausfall eines Standorts der DBMS-Server (DBMS - Datenbankmanagementsystem) mit der Datenbank erneut gestartet werden. Die erforderliche Zeit für den Neustart der Datenbank und des Servers, auf dem sich die Datenbank befindet, lässt sich nicht vorhersehen. Es kann einige Minuten in Anspruch nehmen, bevor die Datenbank wieder in einen konsistenten Zustand versetzt und verfügbar ist. Mit HADR kann eine Bereitschaftsdatenbank innerhalb von Sekunden den Datenbankbetrieb übernehmen. Außerdem können Sie die automatische Clientweiterleitung oder Wiederholungslogik in der Anwendung nutzen, um die Clients, die die ursprüngliche Primärdatenbank verwendet haben, an die neue Primärdatenbank umzuleiten.

Ein kompletter Standortausfall kann auftreten, wenn durch eine Katastrophe wie einen Brand der gesamte Standort zerstört wird. Da HADR zur Kommunikation zwischen der Primärdatenbank und der Bereitschaftsdatenbank jedoch TCP/IP verwendet, können sich die Datenbanken an verschiedenen Standorten befinden. Ihre Primärdatenbank könnte sich z. B. in Ihrer Zentrale in einer Stadt befinden, während Ihre Bereitschaftsdatenbank sich in einer Verkaufsniederlassung in einer anderen Stadt befindet. Im Fall eines Fehlers wird auf der primären Site die Datenverfügbarkeit dadurch gewährleistet, dass die ferne Bereitschaftsdatenbank die Rolle der Primärdatenbank mit vollständiger DB2-Funktionalität übernehmen kann. Nach einer Funktionsübernahmeoperation können Sie die ursprüngliche Primärdatenbank wieder online verfügbar machen und in ihren Status als Primärdatenbank zurückversetzen. Dies wird als *Zurücksetzung* bezeichnet. Eine Zurücksetzung kann eingeleitet werden, wenn die alte Primärdatenbank mit der neuen Primärdatenbank konsistent gemacht werden kann. Nachdem die alte Primärdatenbank der HADR-Konfiguration als Bereitschaftsdatenbank hinzugefügt wurde, können Sie die Rollen der Datenbanken tauschen, sodass die ursprüngliche Primärdatenbank erneut als Primärdatenbank genutzt werden kann.

Bei HADR basiert die Ebene des Schutzes vor potenziellem Datenverlust auf Ihren Entscheidungen in punkto Konfiguration und Topologie. Beispielsweise müssen die folgenden wichtigen Entscheidungen getroffen werden:

Welche Synchronisationsebene soll verwendet werden?

Die Bereitschaftsdatenbanken werden mit der Primärdatenbank über Protokolldaten synchronisiert, die in der Primärdatenbank erstellt und an die Bereitschaftsdatenbanken übertragen werden. Für die Bereitschaftsdatenbanken erfolgt so durch die Protokolle ständig eine aktualisierende Recovery. Sie können aus vier unterschiedlichen Synchronisationsmodi auswählen. Es handelt sich hierbei um die Modi SYNC, NEARSYNC, ASYNC und SUPERASYNC (in der Reihenfolge vom höchsten bis zum geringsten Schutz). Weitere Informationen hierzu finden Sie in „HADR-Synchronisationsmodus (High Availability Disaster Recovery)” auf Seite 67.

Wird ein Peerfenster verwendet?

Über die Funktion des Peerfensters wird angegeben, dass sich die Primär- und die Bereitschaftsdatenbank für einen konfigurierten Zeitraum so verhalten sollen, als ob sie sich weiterhin im Peerstatus befinden würden, wenn die Primärdatenbank die HADR-Verbindung im Peerstatus verliert. Wenn die Primärdatenbank im Peerstatus oder im Status 'Unterbrochener Peer' fehlschlägt, kommt es bei der Funktionsübernahme durch die Bereitschaftsdatenbank zu keinen Datenverlusten. Diese Funktion bietet den

höchsten Schutz. Weitere Informationen hierzu finden Sie in „Konfigurieren der Datenbankkonfigurationsparameter 'hadr_timeout' und 'hadr_peer_window'" auf Seite 52.

Wie viele Bereitschaftsdatenbanken sollen implementiert werden?
Mit HADR können Sie entweder den Modus für eine Bereitschaftsdatenbank oder den Modus für mehrere Bereitschaftsdatenbanken verwenden. Im Modus für mehrere Bereitschaftsdatenbanken können Sie Ihre Ziele im Hinblick auf Hochverfügbarkeit und Disaster-Recovery mit einer einzigen Technologie realisieren. Weitere Informationen hierzu finden Sie in „Mehrere HADR-Bereitschaftsdatenbanken" auf Seite 220.

Es gibt eine Reihe von Möglichkeiten, wie Sie Ihre HADR-Bereitschaftsdatenbank(en) über ihre HA- und DR-Zwecke hinaus nutzen können:

Leseoperationen in der Bereitschaftsdatenbank
Sie können die Funktion für Leseoperationen in der Bereitschaftsdatenbank dazu verwenden, um schreibgeschützte Workloads auf eine oder mehrere Bereitschaftsdatenbanken zu übertragen, ohne die HA- oder DR-Zuständigkeit der Bereitschaftsdatenbank zu beeinträchtigen. Diese Funktion kann dazu beitragen, die Verarbeitungsprozesse auf der Primärdatenbank zu reduzieren, ohne die Hauptzuständigkeit der Bereitschaftsdatenbank zu beeinträchtigen. Weitere Informationen zu diesem Thema finden Sie in „HADR-Funktion für Leseoperationen in der Bereitschaftsdatenbank" auf Seite 246.

Anwendungen können nur auf die aktuelle Primärdatenbank zugreifen, es sei denn, Leseoperationen in der Bereitschaftsdatenbank wurden ermöglicht. Wenn die Funktion für Leseoperationen in der Bereitschaftsdatenbank aktiviert ist, können Anwendungen mit Lesezugriff auf die Bereitschaftsdatenbank umgeleitet werden. Anwendungen, die eine Verbindung zur Bereitschaftsdatenbank herstellen, wirken sich im Falle einer Funktionsübernahme nicht auf die Verfügbarkeit der Bereitschaftsdatenbank aus.

Verzögerte Wiedergabe
Mit der verzögerten Wiedergabe können Sie angeben, dass die Bereitschaftsdatenbank im Hinblick auf die Protokollwiedergabe auf einem früheren Stand als die Primärdatenbank verbleiben soll. Gehen auf der Primärdatenbank Daten verloren oder werden Daten beschädigt, können diese Daten auf der zeitverzögerten Bereitschaftsdatenbank wiederhergestellt werden. Weitere Informationen hierzu finden Sie in „Verzögerte Wiedergabe bei HADR" auf Seite 213.

Schrittweise Aktualisierungen und Upgrades
Mit einer HADR-Konfiguration können ohne Betriebsunterbrechungen verschiedene Upgrades und DB2-Fixpack-Aktualisierungen für die Datenbank vorgenommen werden. Wenn der Modus für mehrere Bereitschaftsdatenbanken aktiviert ist, können Sie ein Upgrade durchführen und gleichzeitig den durch HADR bereitgestellten Schutz beibehalten. Weitere Informationen hierzu finden Sie in „Ausführen von schrittweisen Aktualisierungen und Upgrades in einer DB2-HADR-Umgebung" auf Seite 190.

HADR ist besonders geeignet, wenn Sie die meisten oder alle Daten in der Datenbank schützen müssen oder wenn Sie DDL-Operationen ausführen, die automatisch in eine Bereitschaftsdatenbank repliziert werden müssen. HADR ist jedoch nur eine von verschiedenen Replikationslösungen, die mit der DB2-Produktfamilie angeboten werden. Die InfoSphere Federation Server-Software und das DB2-Datenbanksystem enthalten SQL Replication- und Q Replication-Lösungen, die in einigen

Konfigurationen ebenfalls zur Einrichtung einer hohen Verfügbarkeit verwendet werden können. Diese Lösungen verwalten logisch konsistente Kopien von Datenbanktabellen an verschiedenen Standorten. Zusätzlich bieten sie Flexibilität und komplexe Funktionalität, wie Unterstützung für Spalten- und Zeilenfilter, Datenkonvertierung und Aktualisierungen an beliebigen Kopien einer Tabelle. Ferner können sie in Umgebungen mit partitionierten Datenbanken verwendet werden.

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Einrichten von HADR. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

DB2 High Availability Feature

DB2 High Availability Feature ermöglicht die Integration zwischen dem IBM DB2-Server und der Cluster-Management-Software.

Wenn Sie eine Datenbankmanagerinstanz in einer Clusterumgebung stoppen, müssen Sie den Cluster-Manager darüber informieren, dass die Instanz gestoppt wurde. Wenn der Cluster-Manager über das Stoppen der Instanz nicht informiert wird, versucht er möglicherweise, eine Operation wie beispielsweise eine Funktionsübernahme für die gestoppte Instanz auszuführen. DB2 High Availability Feature stellt eine Infrastruktur bereit, die es dem Datenbankmanager ermöglicht, mit dem Cluster-Manager zu kommunizieren, wenn Konfigurationsänderungen bei Instanzen (wie beispielsweise das Stoppen einer Datenbankmanagerinstanz) Clusteränderungen erforderlich machen.

Wenn der Datenbankmanager mit dem Cluster-Manager kommuniziert, sobald Änderungen bei Instanzen Clusteränderungen erforderlich machen, sind nach der Durchführung von Konfigurationsänderungen bei Instanzen keine separaten Clusteroperationen mehr erforderlich.

DB2 High Availability Feature besteht aus den folgenden Elementen:

- IBM Tivoli System Automation for Multiplatforms (SA MP) ist als Bestandteil von DB2 High Availability Feature mit dem DB2-Server unter AIX und Linux gebündelt und in das DB2-Installationsprogramm integriert. Sie können für SA MP mithilfe des DB2-Installationsprogramms oder der Scripts **installSAM** und **uninstallSAM**, die sich auf den DB2-Server-Installationsmedien befinden, eine Installation, ein Upgrade oder eine Deinstallation durchführen.
- In einer Clusterumgebung erfordern bestimmte Konfigurations- und Verwaltungsoperationen der Datenbankmanagerinstanz entsprechende Änderungen an der Clusterkonfiguration. Mit DB2 High Availability Feature kann der Datenbankmanager automatisch Änderungen an der Konfiguration des Cluster-Managers anfordern, wenn bestimmte Konfigurations- und Verwaltungsoperationen der Datenbankmanagerinstanz ausgeführt werden. Siehe: „Automatisches Konfigurieren eines Clusters mit DB2 High Availability Feature" auf Seite 100
- DB2 High Availability Instance Configuration Utility (db2haicu) ist ein textbasiertes Dienstprogramm zur Konfiguration und Verwaltung hoch verfügbarer Datenbanken in einer Clusterumgebung. Siehe: „DB2 High Availability Instance Configuration Utility (db2haicu)" auf Seite 110

Hohe Verfügbarkeit durch Protokollübertragung

Bei der Protokollübertragung handelt es sich um das Kopieren ganzer Protokolldateien auf eine Bereitschaftsmaschine, entweder von einer Archivierungseinheit aus oder mithilfe eines Benutzerexitprogramms, das für die primäre Datenbank ausgeführt wird.

Die Bereitschaftsdatenbank wird anhand der von der Produktionsmaschine generierten Protokolldateien kontinuierlich aktualisierend wiederhergestellt. Wenn die Produktionsmaschine ausfällt, findet eine Funktionsübernahme statt. Dabei wird Folgendes ausgeführt:

- Die verbleibenden Protokolle werden auf die Bereitschaftsmaschine übertragen.
- Die Bereitschaftsdatenbank wird bis zum Ende der Protokolle aktualisierend wiederhergestellt und gestoppt.
- Die Clients stellen die Verbindung zur Bereitschaftsdatenbank wieder her und nehmen ihre Operationen wieder auf.

Die Bereitschaftsmaschine verfügt über eigene Ressourcen (d. h. Platten), sie muss jedoch dieselben physischen und logischen Definitionen besitzen wie die Produktionsdatenbank. Erstellen Sie bei Verwendung dieses Ansatzes die anfängliche Bereitschaftsdatenbank mithilfe des Restoredienstprogramms (aus einem Backup der Primärdatenbank) oder mithilfe der Funktion zur Erstellung einer Spiegeldatenbank, sofern diese verfügbar ist.

Damit Sie Ihre Datenbank nach einem Katastrophenfall garantiert wiederherstellen können, sollten Sie Folgendes beachten:

- Die Position des Archivs sollte von der Primärstation geographisch getrennt sein.
- Führen Sie eine ferne Spiegelung des Protokolls am Standort der Bereitschaftsdatenbank durch.
- Verwenden Sie einen synchronen Spiegel, bei dem Sie keine Daten verlieren. Sie können hierzu moderne Plattensubsysteme wie ESS und EMC oder eine andere Technologie zum fernen Spiegeln verwenden. Es wird ebenfalls empfohlen, NV-RAM-Cache (lokal und fern) zu verwenden, um die Leistungsauswirkungen einer Recoverysituation nach einem Katastrophenfall zu minimieren.

Wenn Sie steuern möchten, welche Protokolldateien auf der Bereitschaftsmaschine aktualisierend wiederhergestellt werden sollen, können Sie das Abrufen archivierter Protokolle durch Verwendung der Option **NORETRIEVE** des Befehls **ROLLFORWARD DATABASE** inaktivieren. Diese Vorgehensweise bietet die folgenden Vorteile:

- Durch die Steuerung der aktualisierenden Recovery von Protokolldateien können Sie sicherstellen, dass die Bereitschaftsmaschine *X* Stunden hinter der Produktionsmaschine zurückliegt, und somit vermeiden, dass beide Systeme betroffen sind.
- Wenn das Bereitschaftssystem keinen Zugriff auf das Archiv hat (z. B. wenn TSM das Archiv ist), wird nur der ursprünglichen Maschine das Abrufen der Dateien erlaubt.
- Während der Archivierung einer Datei durch das Produktionssystem kann es vorkommen, dass das Bereitschaftssystem dieselbe Datei abruft und in diesem Fall unter Umständen eine unvollständige Protokolldatei erhält. Dieses Problem kann mit der Option **NORETRIEVE** gelöst werden.

Anmerkung:

1. Wenn die Bereitschaftsdatenbank einen Protokollsatz verarbeitet, der angibt, dass in der primären Datenbank eine Indexwiederherstellung erfolgte, werden

die Indizes auf dem Bereitschaftsserver nicht automatisch wiederhergestellt. Der Index auf dem Bereitschaftsserver wird entweder bei der ersten Verbindung zur Datenbank wiederhergestellt, oder beim ersten Versuch, auf den Index zuzugreifen, nachdem der Bereitschaftsserver aus dem Status 'Aktualisierende Recovery anstehend' genommen wurde. Wenn auf dem Primärserver Indizes wiederhergestellt werden, wird empfohlen, den Bereitschaftsserver erneut mit dem Primärserver zu synchronisieren. Sie können die erneute Erstellung von Indizes bei Rollforward-Operationen aktivieren, wenn Sie den Datenbankkonfigurationsparameter **logindexbuild** definieren.

2. Wenn für die primäre Datenbank das Dienstprogramm LOAD mit der Option **COPY YES** ausgeführt wird, muss der Bereitschaftsserver Zugriff auf das Kopierimage haben.
3. Wenn für die primäre Datenbank das Dienstprogramm LOAD mit der Option **COPY NO** ausgeführt wird, sollte die Bereitschaftsdatenbank erneut synchronisiert werden, da andernfalls der Tabellenbereich in den Status 'Restore anstehend' versetzt wird.
4. Eine Bereitschaftsmaschine kann auf zwei Weisen initialisiert werden:
   a. Durch Wiederherstellen auf die Maschine von einem Backup-Image.
   b. Durch Erstellen einer geteilten Spiegeldatenbank des Produktionssystems und Absetzen des Befehls **db2inidb** mit der Option **STANDBY**.

   Sie können den Befehl **ROLLFORWARD DATABASE** erst dann auf dem fehlertoleranten System absetzen, wenn die Bereitschaftsmaschine initialisiert wurde.
5. Operationen, die nicht protokolliert sind, werden nicht auf die Bereitschaftsdatenbank angewendet. Daher wird empfohlen, dass Sie die Bereitschaftsdatenbank nach solchen Operationen erneut synchronisieren. Dies ist durch die Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank möglich.

Spiegeln von Protokollen

Der IBM DB2-Server unterstützt das Spiegeln von Protokollen auf Datenbankebene. Durch das Spiegeln von Protokolldateien wird eine Datenbank vor dem versehentlichen Löschen einer aktiven Protokolldatei sowie vor Datenverlust durch Hardwarefehler geschützt.

Wenn Sie befürchten, dass Ihre aktiven Protokolle aufgrund eines Plattenfehlers beschädigt werden könnten, sollten Sie in Betracht ziehen, mit dem Konfigurationsparameter **mirrorlogpath** einen sekundären Pfad für die Datenbank zur Verwaltung von Kopien der aktiven Protokolldatei anzugeben, sodass die Datenträger gespiegelt werden, auf denen die Protokolle gespeichert sind.

Der Konfigurationsparameter **mirrorlogpath** ermöglicht es der Datenbank, eine identische zweite Kopie der Protokolldateien in einen anderen Pfad zu schreiben. Es wird empfohlen, den sekundären Protokollpfad auf einem physisch getrennten Datenträger einzurichten (der sich vorzugsweise ebenfalls auf einem anderen Plattencontroller befinden sollte). Auf diese Weise kann der Plattencontroller nicht zum SPoF werden.

Wenn Sie erstmalig einen Wert für den Konfigurationsparameter **mirrorlogpath** angeben, verwendet DB2 diesen Wert erst beim nächsten Datenbankstart. Dies verhält sich ähnlich wie beim Konfigurationsparameter **newlogpath**.

Wenn beim Schreiben in den Pfad für aktive Protokolldateien oder in den Pfad für gespiegelte Protokolldateien ein Fehler auftritt, markiert die Datenbank den ent-

sprechenden Pfad als „unbrauchbar", schreibt eine Nachricht in das Protokoll mit den Benachrichtigungen für die Systemverwaltung und schreibt alle folgenden Protokollsätze ausschließlich in den verbleibenden „brauchbaren" Protokollpfad. DB2 versucht nicht, den „unbrauchbaren" Pfad erneut zu verwenden, bis die aktuelle Protokolldatei entweder voll ist oder abgeschnitten wird. Wenn DB2 die nächste Protokolldatei öffnen muss, wird sichergestellt, dass dieser Pfad gültig ist. Ist dies der Fall, wird der Pfad verwendet. Ansonsten versucht DB2 nicht, den Pfad erneut zu verwenden, bis auf die nächste Protokolldatei zum ersten Mal zugegriffen wird. Es findet kein Versuch zur Synchronisierung der Protokollpfade statt, DB2 bewahrt jedoch die Informationen zu auftretenden Zugriffsfehlern auf, sodass die korrekten Pfade verwendet werden, wenn die Protokolldateien archiviert werden. Wenn beim Schreiben in den verbleibenden „brauchbaren" Pfad ebenfalls ein Fehler auftritt, wird die Datenbank abnormal beendet.

Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank

Durch die Unterstützung der ausgesetzten E/A des IBM DB2-Servers können Sie gespiegelte Kopien Ihrer Primärdatenbank teilen, ohne die Datenbank in den Offlinestatus versetzen zu müssen. Sie können diese Funktion verwenden, um rasch eine Bereitschaftsdatenbank zu erstellen, die die Arbeit übernimmt, wenn die Primärdatenbank fehlschlägt.

Die Plattenspiegelung ist ein Vorgang, bei dem Daten gleichzeitig auf zwei unterschiedliche Festplatten geschrieben werden. Die eine Kopie der Daten wird als der "Spiegel" der anderen Kopie bezeichnet. Das Teilen einer solchen Spiegelung bedeutet, dass die beiden Kopien voneinander getrennt werden.

Sie können mit der Plattenspiegelung eine Kopie Ihrer Primärdatenbank verwalten. Sie können die DB2-Server-Funktionalität der ausgesetzten E/A dazu verwenden, die primären und sekundären gespiegelten Kopien der Datenbank zu teilen, ohne die Datenbank in den Offlinestatus versetzen zu müssen. Sobald die primären und sekundären Kopien der Datenbank geteilt sind, kann die sekundäre Datenbank die Operationen übernehmen, falls die Primärdatenbank ausfällt.

Wenn Sie eine große Datenbank lieber nicht mit dem Backup-Dienstprogramm des DB2-Servers sichern möchten, können Sie Kopien von einem Spiegelimage erstellen, indem Sie dazu die ausgesetzte E/A und die Funktion zur Erstellung einer Spiegeldatenbank verwenden. Außerdem erreichen Sie mit dieser Methode Folgendes:

- Sie vermeiden hohen Backup-Aufwand auf der Produktionsmaschine.
- Sie verfügen über eine schnelle Methode, Systeme zu klonen.
- Sie verfügen über eine schnelle Implementierung der Funktionsübernahme aus dem Bereitschaftsmodus. Ein einleitender Restore findet nicht statt, und sollte eine aktualisierende Recovery sich als zu langsam erweisen oder sollten Fehler auftreten, ist die erneute Initialisierung sehr schnell.

Mit dem Befehl **db2inidb** wird die geteilte Spiegeldatenbank initialisiert, sodass Sie sie wie folgt einsetzen können:

- Als Klondatenbank
- Als Bereitschaftsdatenbank
- Als Backup-Image

Dieser Befehl kann nur für eine geteilte Spiegeldatenbank abgesetzt werden, und die geteilte Spiegeldatenbank kann erst nach Ausführung des Befehls verwendet werden.

In einer Umgebung mit partitionierten Datenbanken müssen Sie E/A-Schreibvorgänge nicht auf allen Datenbankpartitionen gleichzeitig aussetzen. Sie können die E/A für eine Untergruppe aus einer oder mehreren Datenbankpartitionen aussetzen, um zum Ausführen von Offline-Backups geteilte Spiegeldatenbanken zu erstellen. Wenn in dieser Untergruppe die Katalogpartition enthalten ist, muss das Aussetzen für diese Datenbankpartition zuletzt erfolgen.

In einer Umgebung mit partitionierten Datenbanken müssen Sie den Befehl **db2inidb** auf jeder Datenbankpartition ausführen, bevor Sie das geteilte Image irgendeiner der Datenbankpartitionen verwenden können. Sie können das Tool in allen Datenbankpartitionen gleichzeitig ausführen, indem Sie den Befehl **db2_all** verwenden. Wenn Sie jedoch die Option `RELOCATE USING` verwenden, können Sie den Befehl **db2inidb** nicht mit dem Befehl **db2_all** auf allen Datenbankpartitionen gleichzeitig ausführen. Für jede Datenbankpartition muss eine eigene Konfigurationsdatei angegeben werden, die den Wert für NODENUM der zu ändernden Datenbankpartition enthält. Wenn beispielsweise der Name der Datenbank geändert wird, sind alle Datenbankpartitionen von dieser Änderung betroffen, und der Befehl **db2relocatedb** muss auf jeder Datenbankpartition mit einer eigenen Konfigurationsdatei ausgeführt werden. Wenn Container versetzt werden, die zu einer einzelnen Datenbankpartition gehören, muss der Befehl **db2relocatedb** nur einmal auf dieser Datenbankpartition ausgeführt werden.

Anmerkung: Stellen Sie sicher, dass die geteilte Spiegeldatenbank alle Container und Verzeichnisse enthält, aus denen die Datenbank besteht, einschließlich des Datenträgerverzeichnisses. Eine Zusammenstellung dieser Informationen finden Sie in der Verwaltungssicht DBPATHS, in der alle Dateien und Verzeichnisse der Datenbank angezeigt werden, die geteilt werden müssen.

Kapitel 4. Konfiguration für hohe Verfügbarkeit

Zur Konfiguration Ihrer DB2-Datenbanklösung für hohe Verfügbarkeit müssen Sie folgende Aufgaben erledigen: Datenbankverwaltungsaktivitäten terminieren, die Primär- und Bereitschaftsdatenbankserver so konfigurieren, dass sie aufeinander eingestellt sind und ihre jeweiligen Rollen im Falle einer Störung kennen, sowie das Konfigurieren von Cluster-Management-Software, um ggf. die Workload von einem fehlgeschlagenen Cluster-Knoten übertragen zu können.

Vorbereitende Schritte

Bevor Sie Ihre Datenbanklösung konfigurieren, müssen die folgenden Schritte ausgeführt werden:

- Montieren und installieren Sie die zugrunde liegenden Hardware- und Softwarekomponenten, aus denen Ihre Lösung besteht. Zu diesen zugrunde liegenden Komponenten können Folgende gehören: Stromversorgung, Netzkonnektivität, Netzkarten, Platten oder andere Speichereinheiten, Betriebssysteme und Software für das Cluster-Management.
- Testen Sie diese zugrunde liegenden Komponenten jeweils ohne eine Auslastung der Datenbank, um sicherzustellen, dass Sie ordnungsgemäß funktionieren, bevor Sie versuchen, sie in Operationen wie Datenbanklastausgleich, Übernahme oder Recovery zu verwenden.

Informationen zu diesem Vorgang

Redundanz ist ein wichtiger Teil einer Hochverfügbarkeitslösung. Wenn Sie jedoch Ihre Verwaltungsaktivitäten auf einen ungünstigen Zeitpunkt legen, nicht mehr über ausreichend Speicherplatz für benötigte Recoveryprotokolle verfügen oder Ihre Cluster-Management-Software nicht korrekt konfiguriert ist, ist Ihre Lösung möglicherweise nicht verfügbar, wenn die Benutzer kritische Aufgaben mit der Datenbank zu erledigen haben.

Vorgehensweise

Zur Konfiguration für hohe Verfügbarkeit gehört Folgendes:

- Konfiguration der Clientweiterleitung
- Konfiguration des Fehlermonitors
- Konfiguration von DB2 High Availability Disaster Recovery (HADR)
- Terminierung von Verwaltungsaktivitäten
- Konfiguration der Protokollierung
- Konfiguration der Cluster-Management-Software

Automatische Clientweiterleitung - Beschreibung und Einrichtung

Ziel der Funktion für automatische Clientweiterleitung ist es in erster Linie, nach einem Übertragungsausfall bei einer IBM Data Server Client-Anwendung eine Wiederherstellung zu ermöglichen, sodass die Anwendung den Betrieb nach einer möglichst kurzen Unterbrechung fortsetzen kann.

Die Weiterleitung ist für die Unterstützung des Dauerbetriebs von zentraler Bedeutung. Eine Weiterleitung ist jedoch nur möglich, wenn für die Clientverbindung eine alternative Position angegeben und verfügbar ist.

Die Funktion für automatische Clientweiterleitung kann in den folgenden konfigurierbaren Umgebungen verwendet werden, wenn es sich bei dem Server um DB2 for Linux, UNIX and Windows handelt:

1. DB2 Enterprise Server Edition mit DB2 Database Partitioning Feature
2. DB2 Enterprise Server Edition mit IBM DB2 pureScale Feature
3. InfoSphere Replication Server
4. IBM PowerHA SystemMirror for AIX
5. High Availability Disaster Recovery (HADR)

   Die automatische Clientweiterleitung funktioniert in Verbindung mit HADR oder DB2 pureScale Feature, um einer Clientanwendung die Fortsetzung ihrer Arbeit bei minimaler Unterbrechung nach einer Funktionsübernahme (Failover) der Datenbank zu ermöglichen, auf die zugegriffen wird.

Die Funktion für automatische Clientweiterleitung wird ebenfalls in den folgenden Konfigurationen verwendet, wenn der Datenbankserver auf einem System i oder System z ausgeführt wird:

1. IBM Data Server-Client stellt eine Verbindung zu einem z/OS- oder einem i5/OS-System über einen DB2 Connect-Server her, der über einen alternativen Server verfügt. Die automatische Clientweiterleitung wird zwischen IBM Data Server Client und zwei DB2 Connect-Servern eingesetzt.
2. DB2 Connect-Clients oder -Serverprodukte, die auf eine DB2 for z/OS Parallel Sysplex-Umgebung für die gemeinsame Nutzung von Daten zugreifen. Die automatische Clientumleitung wird zwischen DB2 Connect und dem z/OS Parallel Sysplex-System eingesetzt. Die Funktion der automatischen Clientweiterleitung unterstützt die nahtlose Funktionsübernahme zwischen einem für DB2 Connect lizenzierten Client und dem Parallel Sysplex. Weitere Informationen zur nahtlosen Funktionsübernahme enthält der Abschnitt zur automatischen Clientweiterleitung (clientseitig) im *DB2 Information Center*.

Da es nicht erforderlich ist, lokale Datenbanken zu synchronisieren, müssen Sie beim DB2 Connect-Server lediglich feststellen, dass für den ursprünglichen und den alternativen DB2 Connect-Server die Zielhost- bzw. System i-Datenbank so katalogisiert ist, dass sie über einen identischen Aliasnamen für die Datenbank zugänglich ist.

Um dem DB2-Datenbanksystem die Möglichkeit zu geben, einen Ausfall der Kommunikationsverbindung zu beheben, muss ein alternativer Serverstandort angegeben werden, bevor der Verlust der Kommunikation auftritt. Zum Definieren des alternativen Serverstandorts in einer bestimmten Datenbank wird der Befehl **UPDATE ALTERNATE SERVER FOR DATABASE** verwendet.

Wenn Sie den alternativen Serverstandort in einer bestimmten Datenbank auf der Serverinstanz angegeben haben, werden die Positionsinformationen dieses alternativen Servers IBM Data Server-Client im Rahmen des Verbindungsaufbauprozesses zurückgegeben. Bei einer automatischen Clientweiterleitung zwischen DB2 Connect-Clients oder -Serverprodukten und einem Host- oder System i-Datenbankserver muss der ferne Server mindestens eine alternative Adresse für sich selbst bereitstellen. Im Fall von DB2 for z/OS sind mehrere Adressen bekannt, wenn es sich bei der Datenbank um eine Sysplexumgebung für die gemeinsame Nutzung von Daten handelt. Daher ist es nicht erforderlich, unter DB2 Connect einen alternati-

ven Server zu katalogisieren. Wenn die Kommunikation zwischen dem Client und dem Server aus irgendeinem Grund ausfällt, versucht der IBM Data Server Client die Verbindung unter Verwendung der Informationen zu dem alternativen Server wiederherzustellen. IBM Data Server-Client wird versuchen, die Verbindung über einen Datenbankserver wiederherzustellen, bei dem es sich um den ursprünglichen Server, einen alternativen Server, der in der Datenbankverzeichnisdatei auf dem Server enthalten ist, oder einen alternativen Server handeln kann, der in der vom z/OS Parallel Sysplex-System zurückgegebenen Serverliste enthalten ist. Der Zeitraum zwischen diesen Versuchen zum Wiederherstellen einer Verbindung beginnt mit sehr kurzen Intervallen, die anschließend im Laufe der weiteren Versuche schrittweise verlängert werden.

Nachdem eine Verbindung erfolgreich hergestellt werden konnte, wird SQL30108N zurückzugeben, um anzuzeigen, dass eine erneute Datenbankverbindung nach dem Kommunikationsausfall hergestellt wurde. Es werden der Hostname oder die IP-Adresse und der Servicename oder die Portnummer zurückgegeben. IBM Data Server-Client gibt den Fehler des ursprünglichen Kommunikationsausfalls nur an die Anwendung zurück, wenn die erneute Herstellung der Clientkommunikation mit dem ursprünglichen oder dem alternativen Server nicht möglich ist.

Ab V10.1 Fixpack 2 gilt beim Herstellen einer Verbindung zur DB2 for z/OS-Gruppe mit gemeinsamer Datennutzung mit aktivierter Lastausgleichsfunktion (Workload Balancing, WLB) ein geändertes Verhalten der nicht nahtlosen automatischen Clientweiterleitung (ACR):

- Der CLI-Treiber sucht bei einem Verbindungsfehler nicht sofort nach einem neuen Transport. Der CLI-Treiber ordnet einen Transport zu, wenn die Anwendung die Anweisung SET (Sonderregister) oder die SQL-Anweisung erneut übergibt. Wenn jedoch die Funktion für die nicht nahtlose automatische Clientweiterleitung aktiviert und die Lastausgleichsfunktion inaktiviert ist, sucht der CLI-Treiber sofort nach einem neuen Transport und stellt die Verbindung zum nächsten verfügbaren Member erneut her.
- SQL30108N wird zweimal an die Anwendung zurückgegeben, wenn der CLI-Treiber keine erneute Verbindung zur Membern der Primärgruppe herstellen kann und zur alternativen Gruppe wechseln muss. Der Fehler wird zweimal zurückgegeben, wenn die alternative Gruppe in der Datei `db2dsdriver.cfg` mit dem Parameter **`alternategroup`** angegeben ist und für **`enableAlternateGroupSeamlessAcr`** der Wert `FALSE`definiert wird. Die erste SQL30108N-Fehlernachricht mit dem Ursachencode 2 wird zurückgegeben, wenn die bestehende Verbindung zu einem Member in der aktuellen Gruppe fehlschlägt. Die zweite SQL30108N-Fehlernachricht mit dem Ursachencode 4 wird zurückgegeben, wenn alle Verbindungsversuche zu allen Membern in der vorhandenen Primärgruppe fehlschlagen. Die Anwendung kann dann die Anweisung SET bzw. die SQL-Anweisung ein zweites Mal übergeben, wenn die erneute Verbindungsherstellung zur alternativen Gruppe bestätigt ist. Der CLI-Treiber überwacht das fehlgeschlagene Member über dieselbe Verbindungskennung, wenn der ACR-Verbindungsfehler (SQL30108N) zurückgegeben wird, um zu vermeiden, dass die Anweisung erneut an das fehlgeschlagene Member übergeben wird.

  Anmerkung: SQL30108N wird in den folgenden Szenarios nicht zweimal ausgegeben:

  – Wenn der DB2 Connect-Server als Gateway verwendet wird.
  – Wenn die automatische Clientweiterleitung (ACR) explizit aktiviert wird, ohne die Lastausgleichsfunktion (WLB) zu aktivieren.

Beim Herstellen einer Verbindung zur DB2 for z/OS-Gruppe mit gemeinsamer Datennutzung dürfen die Funktion für die nahtlose automatische Clientweiterleitung und die Lastausgleichsfunktion (WLB) nicht inaktiviert werden, es sei denn, dies wird von IBM Support explizit verlangt.

Darüber hinaus sind folgende Aspekte der Konnektivität alternativer Server in einer DB2 Connect-Serverumgebung zu beachten:

- Wenn Sie einen DB2 Connect-Server verwenden, um den Zugriff auf eine Host- oder System i-Datenbank für ferne und lokale Clients bereitzustellen, kann es hinsichtlich der Informationen zur Verbindung zu alternativen Servern in einem Eintrag des Systemdatenbankverzeichnisses zu Unklarheiten kommen. Wenn Sie diese Unklarheiten möglichst gering halten möchten, empfiehlt es sich, zwei Einträge im Systemdatenbankverzeichnis zu katalogisieren, die für denselben Host oder dieselbe System i-Datenbank stehen. Katalogisieren Sie einen Eintrag für ferne Clients und einen weiteren für lokale Clients.
- Informationen zum Parallel Sysplex, die von einem DB2 for z/OS-Zielserver zurückgegeben werden, werden auf dem DB2 Connect-Server nur im Cache gespeichert. Es werden nur Informationen eines einzigen alternativen Servers auf Platte geschrieben. Sind mehrere alternative Server oder mehrere aktive Server vorhanden, werden die Informationen nur im Hauptspeicher verwaltet und stehen nach Prozessende nicht mehr zur Verfügung.

Im Allgemeinen wird bei angegebenem alternativen Server die automatische Clientweiterleitung aktiviert, wenn ein Kommunikationsfehler erkannt wird. In einer HADR-Umgebung (High Availability Disaster Recovery) wird sie darüber hinaus aktiviert, wenn SQL1776N vom HADR-Bereitschaftsserver zurückgegeben wird.

Für Lastausgleich und automatische Clientweiterleitung müssen für jedes Member im Cluster, das in der Datei `/etc/hosts` enthalten ist, Einträge auf dem Client vorhanden sein. Beispiel:

```
10.10.10.1 hostname01.linux hostname01
10.10.10.2 hostname02.linux hostname02
```

Konfigurieren der automatischen Clientweiterleitung für die Distributortechnologie für die Clientverbindung

Distributor- oder Dispatchertechnologien wie z. B. WebSphere Edge Server Load Balancer verteilen Verbindungswiederholungsanforderungen der Clientanwendung an eine definierte Gruppe von Systemen, wenn ein primärer Datenbankserver fehlschlägt.

Wenn Sie Distributortechnologie zusammen mit der automatischen Clientweiterleitung von DB2 verwenden, müssen Sie den Distributor selbst als alternativen Server für die automatische Clientweiterleitung von DB2 angeben.

Sie arbeiten möglicherweise mit Distributortechnologie in einer Umgebung, die der folgenden Umgebung ähnlich ist:

Client -> Distributortechnologie -> (DB2 Connect-Server 1 oder DB2 Connect-Server 2) ->DB2 for z/OS

Dabei gilt:

- Die Komponente der Distributortechnologie hat den TCP/IP-Hostnamen 'DThostname'
- Der DB2 Connect-Server 1 hat den TCP/IP-Hostnamen 'GWYhostname1'.

- Der DB2 Connect-Server 2 hat den TCP/IP-Hostnamen 'GWYhostname2'.
- Der DB2 for z/OS-Server hat den TCP/IP-Hostnamen 'zOShostname'.

Auf dem Client wird **DThostname** katalogisiert, damit die Distributortechnologie für den Zugriff auf einen der DB2 Connect-Server genutzt werden kann. Die zwischengeschaltete Distributortechnologie entscheidet darüber, ob **GWYhostname1** oder **GWYhostname2** verwendet wird. Nachdem die Entscheidung getroffen ist, verfügt der Client über eine direkte Socketverbindung zu einem dieser beiden DB2 Connect-Gateways. Sobald die Socketverbindung zum gewünschten DB2 Connect-Server besteht, verfügen Sie über die typische Konnektivität zwischen Client, DB2 Connect-Server und DB2 for z/OS.

Nehmen Sie zum Beispiel an, dass der Distributor **GWYhostname2** auswählt. Dadurch ergibt sich die folgende Umgebung:

Client -> DB2 Connect-Server 2 -> DB2 for z/OS

Wenn ein Kommunikationsfehler auftritt, versucht der Distributor nicht, Verbindungen wiederherzustellen. Wenn Sie die Funktion der automatischen Clientweiterleitung für eine Datenbank in einer solchen Umgebung aktivieren möchten, müssen Sie als alternativen Server für die zugeordnete Datenbank (bzw. Datenbanken) auf dem DB2 Connect-Server (DB2 Connect-Server 1 oder DB2 Connect-Server 2) den Distributor (DThostname) angeben. Wenn dann der DB2 Connect-Server 1 aus irgendeinem Grund gesperrt wird, wird die automatische Clientweiterleitung ausgelöst, und es wird versucht, erneut eine Clientverbindung herzustellen, wobei der Distributor sowohl als primärer als auch als alternativer Server eingesetzt wird. Diese Option ermöglicht es Ihnen, die Funktionen des Distributors mit der DB2-Funktion zur automatischen Clientweiterleitung zu kombinieren und zu verwalten. Wenn Sie den alternativen Server auf einen anderen Hostnamen setzen als den Hostnamen des Distributors, wird den Clients die Funktion der automatischen Clientweiterleitung weiterhin zur Verfügung gestellt. Die Clients stellen jedoch direkte Verbindungen zu dem definierten alternativen Server her und umgehen die Distributortechnologie, wodurch der Nutzen des Distributors verloren geht.

Die Funktion für die automatische Clientweiterleitung fängt die folgenden SQL-Codes ab:

- SQL20157N
- SQL1768N (Ursachencode: 7)

Anmerkung: Die Clientweiterleitung wird möglicherweise nicht in zeitgerechter Form über Socketfehler informiert, wenn der Konfigurationsparameter "TCP Keepalive" des Betriebssystems auf einen zu hohen Wert eingestellt ist. (Beachten Sie, dass der Name dieses Konfigurationsparameters je nach Plattform variiert.)

Angeben eines alternativen Servers für die automatische Clientweiterleitung

Sobald ein DB2-Server oder ein DB2 Connect-Server ausfällt, empfängt jeder mit diesem Server verbundene Client einen Kommunikationsfehler, der die Verbindung beendet und einen Anwendungsfehler verursacht.

In Fällen, in denen die Verfügbarkeit eine wichtige Rolle spielt, sollten Sie entweder eine redundante Konfiguration oder die Möglichkeit zur Übernahme der Serverfunktion durch einen Bereitschaftsknoten implementieren. In beiden Fällen versucht der DB2-Clientcode die Verbindung zu dem ursprünglichen Server, der jetzt

auf einem Übernahmeknoten (die IP-Adresse wird ebenfalls übernommen) ausgeführt wird, oder zu einem neuen Server wiederherzustellen.

Vorgehensweise

Verwenden Sie den Befehl **UPDATE ALTERNATE SERVER FOR DATABASE** oder **UPDATE ALTERNATE SERVER FOR LDAP DATABASE**, um einen neuen oder alternativen Server zu definieren.
Diese Befehle aktualisieren die Informationen zum alternativen Server für einen Aliasnamen einer Datenbank im Systemdatenbankverzeichnis.

Automatische Clientweiterleitung - Einschränkungen

Beachten Sie die Einschränkungen bei der Clientweiterleitung für DB2-Datenbanken, wenn Sie Ihre hoch verfügbare DB2-Datenbanklösung gestalten.

Im Folgenden werden die Einschränkungen aufgelistet, die für die Funktion zur automatischen Clientweiterleitung für DB2-Datenbanken gelten:

- Die automatische Clientweiterleitung wird nur unterstützt, wenn das Kommunikationsprotokoll TCP/IP zur Herstellung der Verbindung zum DB2-Datenbankserver bzw. zum DB2 Connect Server verwendet wird. Dies bedeutet, dass die Funktion zur automatischen Clientweiterleitung nicht aktiviert wird, wenn die Verbindung mit einem anderen Protokoll als TCP/IP arbeitet. Auch wenn die DB2-Datenbank mit der Loopback-Funktion konfiguriert ist, muss das Kommunikationsprotokoll TCP/IP verwendet werden, um die Funktion der automatischen Clientweiterleitung nutzen zu können.
- Bei einer automatischen Weiterleitung zwischen den DB2 Connect-Client- oder Serverprodukten und einem Host- oder System i-Datenbankserver sind in den folgenden Situationen gewisse Aspekte zu beachten:
  - Wenn Sie einen DB2 Connect Server verwenden, um den Zugriff auf eine Host- oder System i-Datenbank für ferne und lokale Clients bereitzustellen, kann es hinsichtlich der Informationen zur Verbindung zu alternativen Servern in einem Eintrag des Systemdatenbankverzeichnisses zu Unklarheiten kommen. Wenn Sie diese Unklarheiten möglichst gering halten möchten, empfiehlt es sich, zwei Einträge im Systemdatenbankverzeichnis zu katalogisieren, die für denselben Host oder dieselbe System i-Datenbank stehen. Katalogisieren Sie einen Eintrag für ferne Clients und einen weiteren für lokale Clients.
  - SYSPLEX-Informationen, die von einem DB2 for z/OS-Zielserver zurückgegeben werden, werden auf dem DB2 Connect Server nur im Cache gespeichert. Es werden nur Informationen eines einzigen alternativen Servers auf Platte geschrieben. Sind mehrere alternative Server oder mehrere aktive Server vorhanden, werden die Informationen nur im Hauptspeicher verwaltet und stehen nach Prozessende nicht mehr zur Verfügung.
- Wenn die Verbindung zum alternativen Serverstandort wiederhergestellt wird, wird jede neue Verbindung zum gleichen Aliasnamen der Datenbank zum alternativen Serverstandort hergestellt. Wenn Sie wollen, dass neue Verbindungen zum ursprünglichen Standort hergestellt werden, nachdem das Problem am ursprünglichen Standort behoben ist, steht eine Reihe von Optionen dazu zur Auswahl:
  - Sie müssen den alternativen Server offline nehmen, sodass die Verbindungen zum ursprünglichen Server wiederhergestellt werden. (Dies setzt voraus, dass der ursprüngliche Server mit dem Befehl **UPDATE ALTERNATE SERVER** katalogisiert wurde, sodass er als alternativer Standort für den alternativen Server definiert ist.)

- Sie könnten einen neuen Aliasnamen der Datenbank katalogisieren, der von den neuen Verbindungen zu verwenden ist.
- Sie könnten den Datenbankeintrag entkatalogisieren und erneut katalogisieren.

- DB2 for Linux, UNIX and Windows unterstützt die Funktion der automatischen Clientweiterleitung sowohl für den Client als auch für den Server, wenn der Client und der Server diese Funktion unterstützen. Andere Familien von DB2-Datenbankprodukten unterstützen diese Funktion zurzeit nicht.
- Das Verhalten der Funktion zur automatischen Clientweiterleitung und das Verhalten der automatischen Clientweiterleitung in einer DB2 for z/OS-Sysplexumgebung sind in gewisser Hinsicht anders. Dabei ist insbesondere zu beachten:
  - Die automatische Clientweiterleitungsfunktion setzt voraus, dass der Primärserver nur einen einzigen alternativen Server angibt. Dies geschieht mithilfe des Befehls **UPDATE ALTERNATE SERVER FOR DATABASE** oder **UPDATE ALTERNATE SERVER FOR LDAP DATABASE**, der auf dem primären Server ausgeführt wird. Dieser Befehl aktualisiert das lokale Datenbankverzeichnis mit den Informationen zum alternativen Server, sodass andere Anwendungen auf dem gleichen Client Zugriff auf diese Informationen haben. Im Gegensatz dazu verwaltet ein Sysplex mit gemeinsamer Datennutzung, das für DB2 for z/OS verwendet wird, im Speicher eine Liste mit einem Server oder mit mehreren Servern, zu denen der Client eine Verbindung herstellen kann. Wenn ein Kommunikationsfehler auftritt, bestimmt der Client mithilfe dieser Liste von Servern die Position des entsprechenden alternativen Servers.
  - Im Fall der automatischen Clientweiterleitungsfunktion informiert der Server den Client über die aktuellsten Einstellungen von Sonderregistern, wenn sich die Einstellung eines Sonderregisters ändert. Dies erlaubt dem Client, die Laufzeitumgebung nach besten Möglichkeiten wiederherzustellen, nachdem eine Weiterleitung stattgefunden hat. Im Gegensatz dazu gibt ein für DB2 for z/OS verwendetes Sysplex die Sonderregistereinstellungen zu Commitgrenzen an den Client zurück. Änderungen an innerhalb der weitergeleiteten UOW verwendeten Sonderregistern müssen deshalb nachvollzogen werden. Alle sonstigen Änderungen werden automatisch nachvollzogen.

  Eine vollständige Unterstützung für die automatische Clientweiterleitung steht nur zwischen einem Linux-, UNIX- oder Windows-Client und einem Linux-, UNIX- oder Windows-Server zur Verfügung. Sie steht nicht zwischen einem Linux-, UNIX- oder Windows-Client und einem Sysplexserver für DB2 for z/OS (unabhängig von der unterstützten Version) zur Verfügung. Lediglich die Weiterleitungsfunktion wird unterstützt.
- Der DB2-Datenbankserver, der auf dem alternativen Host-Server installiert ist, muss dieselbe Version wie die DB2-Datenbankinstanz haben, die auf dem ursprünglichen Host-Server installiert ist (jedoch kann die Fixpack-Version höher sein).
- Unabhängig davon, ob Sie die Berechtigung zum Aktualisieren des Datenbankverzeichnisses auf dem Clientsystem haben, werden die Informationen über den alternativen Server immer im Arbeitsspeicher verwaltet. Das heißt mit anderen Worten, dass andere Anwendungen, wenn Sie keine Berechtigung zum Aktualisieren des Datenbankverzeichnisses hatten (oder weil es sich um ein schreibgeschütztes Datenbankverzeichnis handelt), nicht in der Lage sind, den alternativen Server zu bestimmen und zu verwenden, weil der Arbeitsspeicher nicht von Anwendungen gemeinsam genutzt wird.
- Für alle alternativen Standorte wird die gleiche Authentifizierung ausgeführt. Das bedeutet, dass der Client keine neue Datenbankverbindung herstellen kann, wenn der alternative Standort mit einem anderen Authentifizierungstyp als der ursprüngliche Standort arbeitet.

- Wenn es zu einem Kommunikationsausfall kommt, gehen alle Sitzungsressourcen wie zum Beispiel globale temporäre Tabellen, Identitätswerte, Sequenzen, Cursor, Serveroptionen (SET SERVER OPTION) zur Verarbeitung in föderierten Umgebungen und Sonderregister verloren. Die Anwendung ist dafür zuständig, die Sitzungsressourcen neu einzurichten, um die Verarbeitung fortzusetzen. Sie brauchen nach Wiederherstellung der Verbindung keine der Anweisungen für Sonderregister auszuführen, weil die DB2-Datenbank die Anweisungen für Sonderregister automatisch noch einmal nachvollzieht, die vor dem Verbindungsfehler ausgeführt wurden. Allerdings werden einige Sonderregister nicht neu eingestellt. Dazu gehören die Folgenden:
  - SET ENCRYPTPW
  - SET EVENT MONITOR STATE
  - SET SESSION AUTHORIZATION
  - SET TRANSFORM GROUP

  Bei Problemen mit DB2 Connect empfiehlt es sich, die Liste der für DB2 Connect auf Datenservern spezifischen Sonderregister hinzuzuziehen.
- Wenn der Client mit einem CLI-JCC-Treiber des Typs 2 oder 4 arbeitet, werden nach der Wiederherstellung der Verbindung nach einem Übertragungsfehler die SQL- und XQuery-Anweisungen, die mit dem ursprünglichen Server bereits vorbereitet wurden, implizit mit dem neuen Server erneut vorbereitet. Eingebettete SQL-Routinen (z. B. SQC- oder SQX-Anwendungen) werden jedoch nicht mit dem neuen Server erneut vorbereitet.
- Führen Sie HADR-Befehle (**START HADR**, **STOP HADR** oder **TAKEOVER HADR**) nicht mit Aliasnamen von Datenbanken mit aktivierter Clientweiterleitung aus. HADR-Befehle sind so implementiert, dass sie die Zieldatenbank mithilfe von Datenbankaliasnamen ermitteln. Wenn für die Zieldatenbank eine alternative Datenbank definiert ist, haben HADR-Befehle daher Schwierigkeiten, die Datenbank zu bestimmen, an der der Befehl tatsächlich ausgeführt wird. Ein Client muss die Verbindung möglicherweise über einen Aliasnamen mit aktivierter Clientweiterleitung herstellen, während HADR-Befehle auf eine bestimmte Datenbank angewendet werden. Um diesem Umstand Rechnung zu tragen, können Sie Aliasnamen definieren, die für die Primär- und die Bereitschaftsdatenbank spezifisch sind, und lediglich HADR-Befehle mit diesen Aliasnamen ausführen.
- Da für jeden Datenbankserver nur ein alternativer Server definiert werden kann, müssen Sie in einer Konfiguration mit mehreren HADR-Bereitschaftsdatenbanken eine Bereitschaftsdatenbank (vermutlich die Hauptbereitschaftsdatenbank) als alternativen Server der Primärdatenbank auswählen.

Eine alternative Methode zur Implementierung der automatischen Clientweiterleitung ist die Verwendung des DNS-Eintrags zur Angabe einer alternativen IP-Adresse für einen DNS-Eintrag. Dieser Methode liegt die Idee zugrunde, eine zweite IP-Adresse (einen alternativen Serverstandort) in dem DNS-Eintrag anzugeben. Der Client besitzt dann zwar keine Informationen über einen alternativen Server, jedoch kann das DB2-Datenbanksystem beim Verbindungsaufbau zwischen zwei IP-Adressen für den DNS-Eintrag alternierend auswählen.

Konfigurieren von TCP/IP-Keepalive-Parametern

DB2-Verbindungen zwischen Clients und Servern verwenden für die Kommunikation das Protokoll TCP/IP. Zur Vermeidung potenzieller Probleme bei der Funktionsübernahme, die von Zeitlimitüberschreitungen innerhalb der TCP/IP-Schicht verursacht werden, müssen die TCP/IP-Keepalive-Parameter auf dem Client angepasst werden.

Werden die Keepalive-Werte auf dem Client gesenkt, können Serverfehler schneller erkannt werden.

Es gibt mehrere Methoden für die Aktualisierung der TCP/IP-Keepalive-Parameter des Clients. Welche Methode Sie verwenden, hängt davon ab, ob Ihre Clientverbindung auf dem IBM Data Server Driver for JDBC and SQLJ basiert.

Konfigurieren von TCP/IP-Keepalive-Parametern für Clients mit hoher Verfügbarkeit (JDBC)

Für ein Clientsystem, das den IBM Data Server Driver for JDBC and SQLJ verwendet, werden TCP/IP-Keepalive-Einstellungen durch Anpassen bestimmter Parameter auf der Betriebssystemebene definiert.

Informationen zu diesem Vorgang

Die in diesen Befehlen bereitgestellten Werte sind Vorschläge; Sie sollten diese Einstellungen auf der Basis Ihrer individuellen Netz- und Serverfunktionalität optimieren.

Anmerkung: Da diese Einstellungen auf der Betriebssystemebene geändert werden, wirken sie sich auf die gesamte TCP/IP-Kommunikation des Clients aus.

Vorgehensweise

1. **Aktualisieren von AIX**

   Auf einem AIX-Client müssen drei Keepalive-Parameter des Betriebssystems geändert werden:

   - **tcp_keepidle** - Länge der Zeit, während der eine TCP-Verbindung ohne Datenverkehr aktiv bleibt.
   - **tcp_keepintvl** - Intervall für das Senden von Paketen, um die TCP-Verbindung zu prüfen.
   - **tcp_keepcnt** - Anzahl der Keepalive-Prüfpakete, die gesendet werden, bevor die Verbindung beendet wird.

   Aktualisieren Sie diese Parameter unter dem Betriebssystem AIX mit dem Befehl "network option":

   ```
   no -o tcp_keepidle=12
   no -o tcp_keepintvl=2
   no -o tcp_keepcnt=10
   ```

   Die Werte für **tcp_keepidle** und **tcp_keepintvl** werden in halben Sekunden ausgedrückt.

2. **Aktualisieren von Linux**

   Auf einem Linux-Client müssen vier Keepalive-Parameter des Betriebssystems geändert werden:

   - **tcp_keepalive_probes** - Anzahl der gesendeten und nicht bestätigten Prüfpakete, bis der Client die Verbindung als unterbrochen ansieht und die Anwendungsebene benachrichtigt.
   - **tcp_keepalive_time** - Intervall zwischen dem letzten gesendeten Datenpaket und dem ersten Keepalive-Paket.
   - **tcp_keepalive_intvl** - Intervall zwischen nachfolgenden Keepalive-Prüfpaketen.
   - **tcp_retries2** - Maximale Anzahl der erneuten Versuche, ein Paket zu übertragen.

Aktualisieren Sie diese Parameter unter dem Betriebssystem Linux mit dem Befehl "echo":

```
echo "6" > /proc/sys/net/ipv4/tcp_keepalive_time
echo "1" > /proc/sys/net/ipv4/tcp_keepalive_intvl
echo "10" > /proc/sys/net/ipv4/tcp_keepalive_probes
echo "3" > /proc/sys/net/ipv4/tcp_retries2
```

Die Werte für **tcp_keepalive_time** und **tcp_keepalive_intvl** werden in Sekunden ausgedrückt. Sollen diese Werte auch nach einem Systemwiederanlauf aktiv sein, müssen sie der Datei `/etc/sysctl.conf` hinzugefügt werden.

Nächste Schritte

Bei anderen Clientplattformen finden Sie in der Dokumentation zu Ihrem Betriebssystem Informationen zum Einstellen der TCP/IP-Keepalive-Werte.

Zugehörige Informationen:

- AIX-Befehl 'network option'
- Verwenden von TCP/IP-Keepalive unter Linux
- Microsoft Windows-TCP/IP-Registrierungseinträge

Konfigurieren von TCP/IP-Keepalive-Parametern für Nicht-JDBC-Clients mit hoher Verfügbarkeit (AIX, HP-UX, Linux, Windows)

Die empfohlene Methode zum Einstellen der Keepalive-Parameter auf dem Client besteht darin, den Parameter **keepAliveTimeout** in der Konfigurationsdatei `db2dsdriver.cfg` zu verwenden.

Informationen zu diesem Vorgang

Die in diesen Befehlen bereitgestellten Werte sind Vorschläge; Sie sollten diese Einstellungen auf der Basis Ihrer individuellen Netz- und Serverfunktionalität optimieren.

Vorgehensweise

Bei einem Client, der nicht den IBM Data Server Driver for JDBC and SQLJ verwendet, gibt es zwei Methoden zum Aktualisieren der TCP/IP-Keepalive-Parameter:

- Ändern Sie die Datei `db2dsdriver.cfg`.

  Zum Definieren dieses Parameters bearbeiten Sie die Datei `db2dsdriver.cfg` und versetzen die **keepAliveTimeout**-Zeile so, dass sie sich außerhalb des Abschnitts <acr>, jedoch innerhalb des übergeordneten Abschnitts <databases> befindet. Beispiel:

  ```
  <configuration>
    <dsncollection>
      <dsn alias="D3D" name="D3D" host="DB2PS-member0" port="5912" />
    </dsncollection>
    <databases>
      <database name="D3D" host="DB2PS-member0" port="5912">
      <parameter name="keepAliveTimeout" value="20"/>
      <acr>
        <parameter name="enableAcr" value="true"/>
        <parameter name="enableSeamlessAcr" value="true"/>
  ```

```
      <parameter name="affinityFailbackInterval" value="15"/>
  </databases>
...
</configuration>
```

Diese Methode wird empfohlen, da sie sowohl für instanzbasierte Clients als auch für Treiber ohne Instanz verwendet werden kann. Durch die Verwendung der Datei `db2dsdriver.cfg` kann darüber hinaus für jede einzelne Datenbank eine andere **keepAliveTimeout**-Einstellung definiert werden.

- Ändern Sie den Parameter **DB2TCP_CLIENT_KEEPALIVE_TIMEOUT**.

  Die zweite Methode zum Aktualisieren der Keepalive-Parameter auf einem Client dieses Typs besteht darin, den Parameter **DB2TCP_CLIENT_KEEPALIVE_TIMEOUT** für die Erkennung von Fehlern in der TCP/IP-Kommunikationsebene zu definieren.

  Setzen Sie den folgenden Befehl in einem Befehlsfenster oder Terminal auf dem Client ab, um diesen Parameter zu aktualisieren:

  ```
  db2set DB2TCP_CLIENT_KEEPALIVE_TIMEOUT=20
  ```

  Dieser Wert wird in Sekunden angegeben.

  Anmerkung: TCP/IP-Zeitlimits mit Keepalive werden zwar auch für Instanzverbindungen (mit ATTACH) unterstützt; sie können jedoch nur mit dieser zweiten Methode definiert werden, indem ein Wert für den Parameter **DB2TCP_CLIENT_KEEPALIVE_TIMEOUT** angegeben wird. Beachten Sie, dass die automatische Clientweiterleitung (ACR - Automatic Client Reroute) bei Instanzverbindungen nicht anwendbar ist.

DB2-Registrierungsdatei für Fehlermonitore

Wenn der Dämonprozess des Fehlermonitors gestartet wird, wird eine Registrierungsdatei für Fehlermonitore für jede DB2-Datenbankmanagerinstanz auf allen physischen Maschinen gestartet. Die in dieser Datei enthaltenen Schlüsselwörter und Werte legen das Verhalten des Fehlermonitors fest.

Die Registrierungsdatei für Fehlermonitore befindet sich im Verzeichnis `/sqllib/` und hat den Namen `fm.`*`machine_name`*`.reg`. Diese Datei kann mithilfe des Befehls **db2fm** geändert werden.

Wenn keine Registrierungsdatei für Fehlermonitore vorhanden ist, werden die Standardwerte verwendet.

Es folgt ein Beispiel für den Inhalt der Registrierungsdatei für Fehlermonitore:

```
FM_ON = no
FM_ACTIVE = yes
START_TIMEOUT = 600
STOP_TIMEOUT = 600
STATUS_TIMEOUT = 20
STATUS_INTERVAL = 20
RESTART_RETRIES = 3
ACTION_RETRIES = 3
NOTIFY_ADDRESS = instance_name@machine_name
```

Schlüsselwörter der Registrierungsdatei für Fehlermonitore

FM_ON

Gibt an, ob der Fehlermonitor gestartet werden soll oder nicht. Wenn der Wert auf NO gesetzt ist, wird der Fehlermonitordämon nicht gestartet bzw. wird inaktiviert, falls er bereits gestartet wurde. Der Standardwert ist NO.

FM_ACTIVE

Gibt an, ob der Fehlermonitor aktiv ist. Der Fehlermonitor wird nur Aktionen ausführen, wenn die Parameter **FM_ON** und **FM_ACTIVE** beide auf YES gesetzt sind. Wenn **FM_ON** auf YES und **FM_ACTIVE** auf NO gesetzt ist, wird der Fehlermonitordämon gestartet, er wird jedoch keine Aktionen ausführen. Das bedeutet, er wird nicht versuchen, DB2 erneut in den Onlinestatus zu versetzen, falls dieses beendet wird. Der Standardwert ist YES.

START_TIMEOUT

Gibt die Zeitspanne an, innerhalb deren der Fehlermonitor den Service starten muss, den er überwacht. Der Standardwert ist 600 Sekunden.

STOP_TIMEOUT

Gibt die Zeitspanne an, innerhalb deren der Fehlermonitor den Service beenden muss, den er überwacht. Der Standardwert ist 600 Sekunden.

STATUS_TIMEOUT

Gibt die Zeitspanne an, innerhalb deren der Fehlermonitor den Status des Service abrufen muss, den er überwacht. Der Standardwert ist 20 Sekunden.

STATUS_INTERVAL

Gibt die Mindestzeit zwischen zwei aufeinander folgenden Aufrufen zum Abrufen des Status des überwachten Service an. Der Standardwert ist 20 Sekunden.

RESTART_RETRIES

Gibt an, wie oft der Fehlermonitor versuchen wird, nach einem fehlgeschlagenen Aufruf den Status des überwachten Service erneut abzurufen. Wenn der hier angegebene Wert erreicht wird, versucht der Fehlermonitor, den Service erneut in den Onlinestatus zu versetzen. Der Standardwert ist 3.

ACTION_RETRIES

Gibt an, wie oft der Fehlermonitor versuchen wird, den überwachten Service wieder in den Onlinestatus zu versetzen. Der Standardwert ist 3.

NOTIFY_ADDRESS

Gibt die E-Mail-Adresse an, an die der Fehlermonitor Benachrichtigungen sendet. Der Standardwert ist *instanzname@name_der_maschine*.

DB2-Fehlermonitor mit dem Befehl db2fm konfigurieren

Sie können die DB2-Registrierdatei für Fehlermonitore mit dem Befehl **db2fm** ändern.

Es folgen einige Beispiele für die Verwendung des Befehls **db2fm** zur Aktualisierung der Registrierdatei für Fehlermonitore:

Beispiel 1: START_TIMEOUT aktualisieren

Wenn Sie den Wert für START_TIMEOUT für die Instanz DB2INST1 auf 100 Sekunden aktualisieren möchten, geben Sie den folgenden Befehl in ein DB2-Datenbankbefehlsfenster ein:

```
db2fm -i db2inst1 -T 100
```

Beispiel 2: STOP_TIMEOUT aktualisieren

Wenn Sie den Wert für STOP_TIMEOUT für die Instanz DB2INST1 auf 200 Sekunden aktualisieren wollen, geben Sie den folgenden Befehl ein:

```
db2fm -i db2inst1 -T /200
```

Beispiel 3: START_TIMEOUT und STOP_TIMEOUT aktualisieren

Wenn Sie für die Instanz DB2INST1 den Wert START_TIMEOUT auf 100 Sekunden und den Wert für STOP_TIMEOUT auf 200 aktualisieren wollen, geben Sie den folgenden Befehl ein:

```
db2fm -i db2inst1 -T 100/200
```

Beispiel 4: Fehlermonitor aktivieren

Wenn Sie die Fehlermonitorfunktion für die Instanz DB2INST1 aktivieren wollen, geben Sie den folgenden Befehl ein:

```
db2fm -i db2inst1 -f yes
```

Beispiel 5: Fehlermonitor inaktivieren

Wenn Sie die Fehlermonitorfunktion für die Instanz DB2INST1 inaktivieren wollen, geben Sie den folgenden Befehl ein:

```
db2fm -i db2inst1 -f no
```

Zum Nachweis, dass die Fehlermonitorfunktion für DB2INST1 nicht mehr aktiv ist, geben Sie auf UNIX-Systemen den folgenden Befehl ein:

```
ps -ef|grep -i fm
```

Unter Linux geben Sie den folgenden Befehl ein:

```
ps auxw|grep -i fm
```

Ein Eintrag, der **db2fmd** und DB2INST1 enthält, gibt an, dass der Fehlermonitor für diese Instanz immer noch aktiv ist. Zur Inaktivierung des Fehlermonitors geben Sie den folgenden Befehl als Instanzeigner ein:

```
db2fm -i db2inst1 -D
```

Konfigurieren des DB2-Fehlermonitors mit db2fmcu und Systembefehlen

Sie können den DB2-Fehlermonitor mithilfe des Befehls **db2fmcu** des DB2-Fehlermonitorcontrollers (Fault Monitor Controller - FMC) oder mit Systembefehlen konfigurieren.

Es folgen einige Beispiele für die Verwendung des Befehls **db2fmcu** und von Systembefehlen, um den Fehlermonitor zu konfigurieren:

Beispiel 1: Verhindern, dass der FMC gestartet wird

Mithilfe des FCM-Befehls von DB2 können Sie verhindern, dass der FMC gestartet wird. Der Befehl **db2fmcu** muss mit Rootberechtigung ausgeführt werden, da er auf die Datei `inittab` des Systems zugreift. Führen Sie den folgenden Befehl aus, um zu verhindern, dass der FMC ausgeführt wird:

```
db2fmcu -d
```

Anmerkung: Wenn Sie ein DB2 Data Server-Fixpack anwenden, wird dieses zurückgesetzt, sodass die Datei 'inittab' erneut so konfiguriert wird, dass sie den FMC enthält. Um zu verhindern, dass der FMC gestartet wird, nachdem Sie ein Fixpack angewendet haben, müssen Sie den in diesem Beispiel angegebenen Befehl erneut ausführen.

Beispiel 2: Den FMC für den Start wiederaufnehmen

Wenn Sie den Befehl **db2fmcu -d** rückgängig machen und den FMC wieder in die Datei 'inittab' aufnehmen möchten, geben Sie den folgenden Befehl ein:

```
db2fmcu -u -p  vollständiger_pfad
```

Dabei gilt: *vollständiger_pfad* gibt den vollständigen Pfad zum **db2fmcd**-Objekt an, z. B. '/opt/IBM/db2/bin/db2fmcd'.

Beispiel 3: Die DB2-Datenbankmanagerinstanz automatisch starten

Darüber hinaus können Sie den FMC so einrichten, dass er die Instanz startet, wenn das System anfangs gebootet wird. Zur Aktivierung dieser Funktion für die Instanz DB2INST1 geben Sie den folgenden Befehl ein:

```
db2iauto -on db2inst1
```

Anmerkung: Auf Systemen unter Red Hat Enterprise Linux 6 (RHEL6) wird der DB2 Fault Monitor Coordinator-Dämon (db2fmcd) nach einem Systemwiederanlauf nicht erneut gestartet, sodass die DB2-Instanzen trotz eines korrekt konfigurierten automatischen Starts nicht erneut gestartet werden. Die folgenden technischen Hinweise (Technote) enthalten Informationen zur Aktivierung des Fehlermonitors, sodass 'db2fmcd' auf RHEL6-Systemen automatisch gestartet wird: http://www-01.ibm.com/support/docview.wss?uid=swg21497220.

Beispiel 4: Automatisches Starten der Instanz inaktivieren

Wenn Sie die automatische Startfunktion inaktivieren möchten, geben Sie den folgenden Befehl ein:

```
db2iauto -off db2inst1
```

Beispiel 5: Verhindern, dass Fehlermonitorprozesse gestartet werden

Sie haben auch die Möglichkeit, das Starten von Fehlermonitorprozessen für bestimmte Instanzen auf dem System zu verhindern, indem Sie ein Feld im globalen Registrierdatensatz für die Instanz ändern. Zum Ändern des globalen Registrierfeldes zur Inaktivierung von Fehlermonitoren für die Instanz DB2INST1 geben Sie zum Beispiel den folgenden Befehl als Rootbenutzer ein:

```
db2greg -updinstrec instancename=db2inst1!startatboot=0
```

Wenn Sie diesen Befehl rückgängig machen und Fehlermonitore für die Instanz DB2INST1 reaktivieren möchten, geben Sie den folgenden Befehl als Rootbenutzer ein:

```
db2greg -updinstrec instancename=db2inst1!startatboot=1
```

Initialisieren von High Availability Disaster Recovery (HADR)

Verwenden Sie diese Prozedur, um die primäre und die Bereitschaftsdatenbank für DB2 High Availability Disaster Recovery (HADR) im Modus für eine Bereitschaftsdatenbank einzurichten und zu initialisieren.

Informationen zu diesem Vorgang

HADR kann über den Befehlszeilenprozessor (CLP) oder duch Aufrufen der API 'db2HADRStart' initialisiert werden.

Vorgehensweise

Gehen Sie wie folgt vor, um auf Ihrem System HADR über den Befehlszeilenprozessor zum ersten Mal zu initialisieren:

1. Ermitteln Sie für jede der HADR-Datenbanken den Hostnamen, die IP-Adresse und den Servicenamen oder die Portnummer.

   Wenn ein Host über mehrere Netzschnittstellen verfügt, stellen Sie sicher, dass der HADR-Hostname oder die IP-Adresse mit der gewünschten übereinstimmt. Sie müssen für jede geschützte Datenbank separate HADR-Ports in der Datei `/etc/services` zuordnen. Diese dürfen nicht mit den der Instanz zugeordneten Ports identisch sein. Der Hostname kann nur einer einzigen IP-Adresse zugeordnet werden.

   Anmerkung: Die Instanznamen für die Primärdatenbank und die Bereitschaftsdatenbanken müssen nicht identisch sein.
2. Erstellen Sie die Bereitschaftsdatenbank, indem Sie ein Backup-Image wiederherstellen oder indem Sie basierend auf der vorhandenen Datenbank, die als Primärdatenbank verwendet werden soll, eine geteilte Spiegeldatenbank initialisieren.

   Im folgenden Beispiel werden die Befehle **BACKUP DATABASE** und **RESTORE DATABASE** verwendet, um die Datenbank SOCKS als Bereitschaftsdatenbank zu initialisieren. Im Fall dieses Beispiels kann auf ein angehängtes NFS-Dateisystem von beiden Standorten aus zugegriffen werden.

   Setzen Sie in der Primärdatenbank den folgenden Befehl ab:

   ```
   backup db socks to /nfs1/backups/db2/socks
   ```

   Setzen Sie in der Bereitschaftsdatenbank den folgenden Befehl ab:

   ```
   restore db socks from /nfs1/backups/db2/socks replace history file
   ```

   Das folgende Beispiel zeigt, wie mit dem Dienstprogramm **db2inidb** unter Verwendung einer geteilten Spiegeldatenbank der Primärdatenbank die Bereitschaftsdatenbank initialisiert wird. Diese Prozedur ist eine Alternative zur oben beschriebenen Prozedur zum Sichern und Wiederherstellen.

   Setzen Sie in der Bereitschaftsdatenbank den folgenden Befehl ab:

   ```
   db2inidb socks as standby
   ```

   Anmerkung:

   a. Die Datenbanknamen für die Primärdatenbank und die Bereitschaftsdatenbank müssen identisch sein.
   b. Setzen Sie nach der Restoreoperation oder der Initialisierung der geteilten Spiegeldatenbank nicht den Befehl **ROLLFORWARD DATABASE** für die Bereitschaftsdatenbank ab. Die Ergebnisse einer Operation zur aktualisierenden Recovery können geringfügig von dem Ergebnis abweichen, das durch die Anwendung der Protokolle auf die Bereitschaftsdatenbank unter Verwendung von HADR erzielt wird. Wenn die Datenbanken nicht identisch sind, schlägt der Start der Bereitschaftsdatenbank fehl.
   c. Verwenden Sie die Option `REPLACE HISTORY FILE` mit dem Befehl **RESTORE DATABASE**.

d. Wenn Sie die Bereitschaftsdatenbank mithilfe des Befehls **RESTORE DATABASE** erstellen, stellen Sie sicher, dass diese im Modus "aktualisierende Recovery anstehend" oder "aktualisierende Recovery läuft" verbleibt. Das bedeutet, dass Sie den Befehl **ROLLFORWARD DATABASE** weder mit der Option COMPLETE noch mit der Option STOP ausführen können. Wenn in der Datenbank nach Beendigung der aktualisierenden Recovery versucht wird, den Befehl **START HADR** mit der Option AS STANDBY auszuführen, wird ein Fehler zurückgegeben.

e. Beim Konfigurieren der Bereitschaftsdatenbank sollten Sie die folgenden Optionen des Befehls **RESTORE DATABASE** vermeiden: TABLESPACE, INTO, REDIRECT, und WITHOUT ROLLING FORWARD.

f. Wenn Sie die Bereitschaftsdatenbank mit dem Dienstprogramm **db2inidb** konfigurieren, verwenden Sie nicht die Optionen SNAPSHOT oder MIRROR. Sie können die Option RELOCATE USING angeben, um wahlweise die folgenden Konfigurationsattribute zu ändern: Instanzname, Protokollpfad und Datenbankpfad. Sie dürfen jedoch nicht den Namen der Datenbank oder die Pfade der Tabellenbereichscontainer ändern.

3. Setzen Sie die folgenden HADR-Konfigurationsparameter in der Primärdatenbank und der Bereitschaftsdatenbank:
   - **hadr_local_host**
   - **hadr_local_svc**
   - **hadr_remote_host**
   - **hadr_remote_svc**
   - **hadr_remote_inst**

   Diese Konfigurationsparameter müssen nach dem Erstellen der Bereitschaftsdatenbank gesetzt werden. Wenn diese Parameter vor dem Erstellen der Bereitschaftsdatenbank definiert werden, geben die Einstellungen in der Bereitschaftsdatenbank die Einstellungen der Primärdatenbank wieder.

   Anmerkung: Dies ist eine allgemeine HADR-Konfiguration. Erweiterte Konfigurationsoptionen und -einstellungen finden Sie unter den folgenden Links.

4. Stellen Sie eine Verbindung zur Bereitschaftsinstanz her, und starten Sie HADR für die Bereitschaftsdatenbank, wie im folgenden Beispiel gezeigt:

   ```
   START HADR ON DB SOCKS AS STANDBY
   ```

   Anmerkung: In der Regel wird die Bereitschaftsdatenbank zuerst gestartet. Wenn Sie die Primärdatenbank zuerst starten, schlägt diese Startprozedur fehl, wenn die Bereitschaftsdatenbank nicht innerhalb des vom Datenbankkonfigurationsparameter **hadr_timeout** angegebenen Zeitlimits gestartet wird.
   Nach dem Start der Bereitschaftsdatenbank wechselt diese in den Status *Lokales Catch-up*, in dem lokal verfügbare Protokolldateien gelesen und wiedergegeben werden. Nachdem alle lokalen Protokolle wiedergegeben wurden, geht die Datenbank in den Status *Fernes Catch-up anstehend* über.

5. Stellen Sie eine Verbindung zur Primärinstanz her, und starten Sie HADR für die Primärdatenbank, wie im folgenden Beispiel gezeigt:

   ```
   START HADR ON DB SOCKS AS PRIMARY
   ```

   Nach dem Start der Primärdatenbank wechselt diese in den Status *Fernes Catch-up*, in dem Protokollseiten von der Primärdatenbank erhalten und wiedergegeben werden. Nachdem alle auf der Festplatte der Primärdatenbank gespeicherten Protokolldateien wiedergegeben wurden, wechseln die Primär- und die Bereitschaftsdatenbank in den Status *Peer*.

Automatische Clientweiterleitung und HADR (High Availability Disaster Recovery) konfigurieren

Sie können die Funktion für automatische Clientweiterleitung mit der Funktion HADR (High Availability Disaster Recovery) verwenden, um die Anforderungen der Clientanwendungen eines fehlgeschlagenen Datenbankservers an einen Bereitschaftsdatenbankserver weiterzuleiten.

Einschränkungen

- Eine Weiterleitung ist nur möglich, wenn auf dem Server eine alternative Datenbankposition angegeben wurde.
- Die automatische Clientweiterleitung wird nur mit TCP/IP-Protokoll unterstützt.

Konfigurationsdetails

- Verwenden Sie den Befehl **UPDATE ALTERNATE SERVER FOR DATABASE**, um die automatische Clientweiterleitung zu aktivieren.
- Die automatische Clientweiterleitung verwendet die Datenbankkonfigurationsparameter **hadr_remote_host** und **hadr_remote_svc** nicht.
- Im Modus für mehrere Bereitschaftsdatenbanken kann für die automatische Clientweiterleitung nur eine Bereitschaftsdatenbank angegeben werden.
- Die alternative Hostposition ist in der Systemdatenbankverzeichnisdatei auf dem Server gespeichert.
- Wenn die automatische Clientweiterleitung nicht aktiviert ist, erhalten Clientanwendungen die Fehlernachricht SQL30081N, und es werden keine weiteren Versuche unternommen, eine Verbindung zum Server herzustellen.

Befehl UPDATE ALTERNATE SERVER FOR DATABASE zum Einrichten der automatischen Clientweiterleitung mit HADR verwenden

Ihr System ist wie folgt eingerichtet:

- Sie haben einen Client, auf dem die Datenbank MUSIC mit der Information katalogisiert ist, dass sie sich auf dem Host HORNET befindet.
- Die Datenbank MUSIC ist die Primärdatenbank, und ihre zugehörige Bereitschaftsdatenbank, die ebenfalls den Namen MUSIC hat, befindet sich auf Host MONTERO mit der Portnummer 456, die durch den Konfigurationsparameter **svcename** zugewiesen ist.

Um die automatische Clientweiterleitung zu aktivieren, aktualisieren Sie den alternativen Server für die Datenbank MUSIC auf Host HORNET:

```
db2 update alternate server for database music using hostname montero port 456
```

Nachdem dieser Befehl abgesetzt wurde, muss der Client eine erfolgreiche Verbindung zum Host HORNET herstellen, um die Informationen zum alternativen Server abzurufen. Wenn dabei ein Kommunikationsfehler zwischen dem Client und der Datenbank MUSIC auf dem Host HORNET auftritt, versucht der Client zunächst, erneut eine Verbindung zur Datenbank MUSIC auf dem Host HORNET herzustellen. Wenn dies fehlschlägt, versucht der Client anschließend, eine Verbindung zur Bereitschaftsdatenbank MUSIC auf dem Host MONTERO herzustellen.

Indexprotokollierung und High Availability Disaster Recovery (HADR)

Es empfiehlt sich, die Datenbankkonfigurationsparameter **logindexbuild** und **indexrec** für HADR-Datenbanken einzurichten.

Verwenden des Datenbankkonfigurationsparameters logindexbuild

Empfehlung: Setzen Sie den Datenbankkonfigurationsparameter **logindexbuild** für HADR-Datenbanken auf ON, um sicherzustellen, dass für die Erstellung, Neuerstellung und Reorganisation von Indizes die vollständigen Informationen protokolliert werden. Auch wenn dies bedeutet, dass die Indexerstellung auf dem primären System mehr Zeit beansprucht und mehr Protokollspeicherbereich erforderlich ist, werden die Indizes während der Wiederholung des HADR-Protokolls erneut erstellt und stehen für den Fall einer Funktionsübernahme zur Verfügung. Andernfalls markiert das Bereitschaftssystem bei einer wiederholten Indexerstellung oder einem Rebuild-Ereignis den Index als ungültig, da die Protokolleinträge nicht genügend Informationen zum Auffüllen des neuen Index enthalten. Wenn die Indexerstellung auf dem primären System nicht protokolliert wird und eine Funktionsübernahme erforderlich wird, müssen alle ungültigen Indizes, die nach Abschluss der Funktionsübernahme übrig bleiben, erneut erstellt werden, bevor auf sie zugegriffen werden kann. Während der erneuten Erstellung der Indizes stehen diese anderen Anwendungen nicht zur Verfügung.

Anmerkung: Wenn das Tabellenattribut LOG INDEX BUILD auf seinen Standardwert NULL gesetzt ist, verwendet DB2 den für den Datenbankkonfigurationsparameter **logindexbuild** angegebenen Wert. Wenn das Tabellenattribut LOG INDEX BUILD auf ON oder OFF gesetzt ist, wird der Wert für den Datenbankkonfigurationsparameter **logindexbuild** ignoriert.

Sie können das Tabellenattribut LOG INDEX BUILD aus einem der folgenden Gründe für eine Tabelle oder mehrere Tabellen auf OFF setzen:

- Sie verfügen nicht über ausreichend Speicherbereich für die aktiven Protokolldateien, um die Protokollierung der Indexerstellung zu unterstützen.
- Die Indexdaten sind sehr umfangreich, und auf die Tabelle wird nicht häufig zugegriffen; daher ist es vertretbar, die Indizes im Anschluss an die Übernahmeoperation erneut zu erstellen. Setzen Sie in diesem Fall den Konfigurationsparameter **indexrec** auf RESTART. Da auf die Tabelle nicht häufig zugegriffen wird, bewirkt diese Einstellung, dass das System die Indizes am Ende der Übernahmeoperation erneut erstellt und nicht auf den ersten Zugriff auf die Tabelle nach der Übernahmeoperation wartet.

Wenn das Tabellenattribut LOG INDEX BUILD für mindestens eine Tabelle auf OFF gesetzt ist, kann eine beliebige Indexerstellungsoperation für diese Tabellen dazu führen, dass die Indizes bei jedem Auftreten einer Funktionsübernahmeoperation erneut erstellt werden. Ist das Tabellenattribut LOG INDEX BUILD auf seinen Standardwert NULL und der Datenbankkonfigurationsparameter **logindexbuild** auf OFF gesetzt, kann ebenfalls eine beliebige Indexerstellungsoperation für diese Tabellen dazu führen, dass die Indizes bei jedem Auftreten einer Übernahmeoperation erneut erstellt werden. Sie können die erneute Erstellung der Indizes verhindern, indem Sie eine der folgenden Aktionen ausführen:

- Nachdem alle ungültigen Indizes in der neuen Primärdatenbank erneut erstellt wurden, erstellen Sie ein Backup der Datenbank, und wenden Sie dieses Backup auf die Bereitschaftsdatenbank an. Dadurch muss die Bereitschaftsdatenbank

nicht die zur erneuten Erstellung ungültiger Indizes verwendeten Protokolle auf die Primärdatenbank anwenden, wodurch diese Indizes in der Bereitschaftsdatenbank eine Markierung erhalten würden, dass eine erneute Erstellung erforderlich ist.

- Setzen Sie das Tabellenattribut LOG INDEX BUILD auf ON, oder setzen Sie das Tabellenattribut LOG INDEX BUILD auf NULL und den Konfigurationsparameter **logindexbuild** für die Bereitschaftsdatenbank auf ON, um sicherzustellen, dass die Indexerstellung protokolliert wird.

Verwenden des Datenbankkonfigurationsparameters indexrec

Empfehlung: Setzen Sie den Datenbankkonfigurationsparameter **indexrec** für die Primärdatenbank und die Bereitschaftsdatenbank auf RESTART (Standardeinstellung). Diese Einstellung bewirkt, dass ungültige Indizes nach Abschluss einer Übernahmeoperation erneut erstellt werden. Wenn noch keine Indexerstellungen protokolliert wurden, ermöglicht diese Einstellung DB2 die Suche nach ungültigen Indizes und das erneute Erstellen dieser Indizes. Dieser Prozess findet im Hintergrund statt, und die Datenbank steht nach der erfolgreichen Beendigung der Übernahmeoperation wieder zur Verfügung.

Wenn eine Transaktion auf eine Tabelle zugreift, die ungültige Indizes enthält, bevor die Indizes durch den Hintergrundprozess erneut erstellt wurden, werden die ungültigen Indizes durch diese erste Transaktion erneut erstellt.

Datenbankkonfiguration für HADR (High Availability Disaster Recovery)

Mit der Hilfe von Datenbankkonfigurationsparametern können Sie mit DB2 HADR eine optimale Leistung erreichen.

In den meisten Fällen sollten die Einstellungen für die Datenbankkonfigurationsparameter und die Einstellungen für die Datenbankmanagerkonfigurationsparameter auf allen Systemen, auf denen sich die Primärdatenbank und die Bereitschaftsdatenbank befinden, identisch definiert sein. Wenn sich die Einstellungen für die Konfigurationsparameter auf der Bereitschaftsdatenbank von den Einstellungen auf der Primärdatenbank unterscheiden, können die folgenden Probleme auftreten:

- Während der Wiedergabe der von der Primärdatenbank übertragenen Protokolldateien werden für die Bereitschaftsdatenbank möglicherweise Fehlernachrichten zurückgegeben.
- Nach einer Übernahmeoperation kann die neue Primärdatenbank möglicherweise die Auslastung nicht bewältigen, wodurch es zu Leistungsproblemen oder zur Ausgabe von Fehlernachrichten kommen kann, welche die Anwendungen während ihrer Verbindungen zur ursprünglichen Primärdatenbank nicht empfingen.

Änderungen an den Konfigurationsparametern in der Primärdatenbank werden nicht automatisch an die Bereitschaftsdatenbank weitergegeben. Diese müssen in der Bereitschaftsdatenbank manuell vorgenommen werden. Für dynamische Konfigurationsparameter werden diese Änderungen wirksam, ohne dass das Datenbankverwaltungssystem (DBMS - Database Management System) oder die Datenbank heruntergefahren und erneut gestartet werden müssen. Bei nicht dynamischen Konfigurationsparametern werden die Änderungen nach dem Neustart der Bereitschaftsdatenbank wirksam.

In den folgenden Abschnitten werden bestimmte Konfigurationsthemen für HADR behandelt:

- „Konfigurationsparameter für die Größe der Protokolldateien in der Bereitschaftsdatenbank"
- „Größe des Protokollempfangspuffers in einer Bereitschaftsdatenbank"
- „Ladeoperationen und HADR" auf Seite 45
- „Registrierdatenbankvariable DB2_HADR_PEER_WAIT_LIMIT" auf Seite 46
- „HADR-Konfigurationsparameter" auf Seite 47

Konfigurationsparameter für die Größe der Protokolldateien in der Bereitschaftsdatenbank

Eine Ausnahme von der im obigen Abschnitt beschriebenen Funktionsweise der Konfigurationsparameter ist der Datenbankkonfigurationsparameter **logfilsiz**. Auch wenn der Wert dieses Parameters nicht in die Bereitschaftsdatenbank repliziert wird, ignoriert die Bereitschaftsdatenbank die Einstellung für den Konfigurationsparameter **logfilsiz**, um identische Protokolldateien für beide Datenbanken zu garantieren. Stattdessen erstellt die Datenbank lokale Protokolldateien, die in der Größe den Protokolldateien der Primärdatenbank entsprechen.

Nach einer Übernahme verwendet die ursprüngliche Bereitschaftsdatenbank (die neue Primärdatenbank) den Wert für **logfilsiz**, der auf der ursprünglichen Primärdatenbank festgelegt wurde, bis die Datenbank erneut gestartet wird. An diesem Punkt kehrt die neue Primärdatenbank zu ihrem lokal konfigurierten Wert zurück. Darüber hinaus schneidet die neue Primärdatenbank die aktuelle Protokolldatei ab und ändert die Größen aller im Voraus erstellten Protokolldateien.

Wenn die Datenbanken die Rollen als Ergebnis einer nicht erzwungenen Übernahme tauschen und keine Datenbank inaktiviert wird, bleibt die Protokolldateigröße immer diejenige, die auf der ursprünglichen Primärdatenbank festgelegt wurde. Wenn die ursprüngliche Bereitschaftsdatenbank (die neue Primärdatenbank) jedoch inaktiviert und erneut gestartet wird, verwendet die neue Primärdatenbank die lokal konfigurierte Protokolldateigröße. Diese Protokolldateigröße wird anschließend auch dann weiter verwendet, wenn die ursprüngliche Primärdatenbank erneut übernimmt. Erst nach einer Inaktivierung und einem Neustart des ursprünglich primären Systemen nimmt die Protokolldateigröße wieder den Wert des ursprünglich primären Systems an.

Größe des Protokollempfangspuffers in einer Bereitschaftsdatenbank

Standardmäßig entspricht die Größe des Protokollempfangspuffers in einer Bereitschaftsdatenbank dem doppelten Wert des in der Primärdatenbank angegebenen Konfigurationsparameters **logbufsz**. Diese Größe ist möglicherweise nicht ausreichend. Denken Sie beispielsweise daran, was passieren könnte, wenn der HADR-Synchronisationsmodus auf ASYNC gesetzt ist und sich die Primär- und die Bereitschaftsdatenbank im Peerstataus befinden. Die Kapazität des Protokollempfangspuffers in der Bereitschaftsdatenbank kann sich bei Auftreten einer hohen Transaktionslast in der Primärdatenbank erschöpfen und die Protokollübertragungsoperation von der Primärdatenbank können zum Erliegen kommen. Nehmen Sie zur Bewältigung dieser hohen temporären Systemauslastungen eine der folgenden Konfigurationsänderungen vor:

- Erhöhen Sie die Größe des Protokollempfangspuffers in der Bereitschaftsdatenbank, indem Sie den Wert der Registrierdatenbankvariable **DB2_HADR_BUF_SIZE** ändern.

- Aktivieren Sie auf einer Bereitschaftsdatenbank das Protokollspooling, indem Sie den Konfigurationsparameter **hadr_spool_limit** definieren.

Ladeoperationen und HADR

Wenn Sie in der Primärdatenbank den Befehl **LOAD** mit dem Parameter **COPY YES** absetzen, wird der Befehl in der Primärdatenbank ausgeführt, und die Daten werden in die Bereitschaftsdatenbank repliziert, solange über den mit dem Befehl angegebenen Pfad oder die angegebene Einheit auf die Ladekopie zugegriffen werden kann. Kann die Bereitschaftsdatenbank auf diese Daten nicht zugreifen, wird der Tabellenbereich, in dem die Tabelle gespeichert ist, in der Bereitschaftsdatenbank als ungültig markiert. Zukünftige Protokollsätze, die zu diesem Tabellenbereich gehören, werden übersprungen. Um sicherzustellen, dass die Ladeoperation auf die Ladekopie in der Bereitschaftsdatenbank zugreifen kann, müssen Sie eine gemeinsam genutzte Speicherposition für die Ausgabedatei des Parameters **COPY YES** verwenden. Alternativ können Sie auch die Bereitschaftsdatenbank für die Dauer der Ladeoperation in der Primärdatenbank inaktivieren, eine Kopie der Ausgabedatei in den Bereitschaftspfad stellen und anschließend die Bereitschaftsdatenbank wieder aktivieren.

Wenn Sie den Befehl **LOAD** mit dem Parameter **NONRECOVERABLE** für die Primärdatenbank absetzen, wird der Befehl in der Primärdatenbank ausgeführt, und die Tabelle wird in der Bereitschaftsdatenbank als ungültig markiert. Zukünftige Protokollsätze, die zu dieser Tabelle gehören, werden übersprungen. Sie können den Befehl **LOAD** mit den Parametern **COPY YES** und **REPLACE** angeben, um die Tabelle zurückzuholen, oder Sie können die Tabelle löschen, um den Speicherbereich freizugeben.

Da HADR Ladeoperationen mit dem Parameter **COPY NO** nicht unterstützt, wird die Operation automatisch mit dem Parameter **NONRECOVERABLE** in eine Ladeoperation konvertiert. Definieren Sie in der Primärdatenbank die Registrierdatenbankvariable **DB2_LOAD_COPY_NO_OVERRIDE**, wodurch eine Ladeoperation mit dem Parameter **COPY NO** in eine Ladeoperation mit dem Parameter **COPY YES** konvertiert wird. Diese Registrierdatenbankvariable wird in der Bereitschaftsdatenbank ignoriert. Stellen Sie sicher, dass die Bereitschaftsdatenbank auf die in der Primärdatenbank angegebene Einheit oder das angegebene Verzeichnis über denselben Pfad bzw. mit derselben Einheit oder Ladebibliothek zugreifen kann.

Wenn Sie die Tivoli Storage Manager-Software (TSM) zum Ausführen einer Ladeoperation mit dem Parameter **COPY YES** verwenden, müssen Sie möglicherweise in der Primärdatenbank und in der Bereitschaftsdatenbank den Konfigurationsparameter **vendoropt** setzen. Je nach Konfiguration von TSM sind die Werte in der Primärdatenbank und der Bereitschaftsdatenbank möglicherweise unterschiedlich. Wenn Sie TSM zum Ausführen einer Ladeoperation mit dem Parameter **COPY YES** verwenden, müssen Sie außerdem den Befehl **db2adutl** mit dem Parameter **GRANT** absetzen, um der Bereitschaftsdatenbank Lesezugriff auf die geladenen Dateien zu erteilen.

Wenn Tabellendaten von einer Ladeoperation mit dem Parameter **COPY YES** repliziert werden, werden die Indizes wie folgt repliziert:

- Wenn Sie als Option für den Indexierungsmodus REBUILD zusammen mit dem Befehl **LOAD** angeben und das Tabellenattribut LOG INDEX BUILD auf den Wert ON gesetzt ist (mithilfe der Anweisung ALTER TABLE) oder wenn das Attribut auf den Wert NULL gesetzt ist und der Datenbankkonfigurationsparameter **logindexbuild** die Einstellung ON aufweist, schließt die Primärdatenbank das erneut erstellte Indexobjekt (d. h. alle für die Tabelle definierten Indizes) in die Kopierdatei ein, um der Bereitschaftsdatenbank das Replizieren des Indexobjekts

zu ermöglichen. Wenn das Indexobjekt in der Bereitschaftsdatenbank vor der Ladeoperation als ungültig markiert wurde, wird es nach der Ladeoperation durch die erneute Indexerstellung wieder verwendbar.

- Wenn Sie als Option für Indexierungsmodus `INCREMENTAL` zusammen mit dem Befehl **`LOAD`** angeben und das Tabellenattribut LOG INDEX BUILD ist auf den Wert `ON` gesetzt (mithilfe der Anweisung ALTER TABLE) oder wenn das Attribut auf den Wert `NULL` gesetzt ist und der Datenbankkonfigurationsparameter **`logindexbuild`** in der Primärdatenbank weist den Wert `ON` auf, wird das Indexobjekt in der Bereitschaftsdatenbank nur aktualisiert, wenn es vor der Ladeoperation nicht als ungültig markiert wurde. Andernfalls wird der Index in der Bereitschaftsdatenbank als ungültig markiert.

Registrierdatenbankvariable DB2_HADR_PEER_WAIT_LIMIT

Einschränkung: Im Modus für mehrere Bereitschaftsdatenbanken gelten die Informationen in diesem Abschnitt nicht für die Nebenbereitschaftsdatenbanken, da sich diese im Synchronisationsmodus SUPERASYNC befinden und deshalb niemals in den Peerstatus wechseln.

Wenn Sie die Registrierdatenbankvariable **`DB2_HADR_PEER_WAIT_LIMIT`** definieren, wird für die HADR-Primärdatenbank der Peerstatus aufgehoben, wenn die Protokollierung in der Primärdatenbank für die angegebene Anzahl von Sekunden blockiert wurde, weil das Protokoll in die Bereitschaftsdatenbank repliziert wird. Wenn dieser Grenzwert erreicht wird, unterbricht die Primärdatenbank die Verbindung zur Bereitschaftsdatenbank. Wenn Sie das Peerfenster inaktivieren, indem Sie den Konfigurationsparameter **`hadr_peer_window`** auf `0` setzen, wechselt die Primärdatenbank in den Status 'Unterbrochener Peer', und die Protokollierung wird wieder aufgenommen. Wenn Sie das Peerfenster aktivieren, wird die Primärdatenbank in den Status 'Unterbrochener Peer' versetzt, in dem die Protokollierung weiterhin blockiert ist. Die Primärdatenbank verlässt den Status 'Unterbrochener Peer', wenn die Verbindung wieder hergestellt wird oder das Peerfenster abläuft. Die Protokollierung wird wieder aufgenommen, sobald der Status 'Unterbrochener Peer' für die Primärdatenbank aufgehoben ist.

Anmerkung: Falls Sie **`DB2_HADR_PEER_WAIT_LIMIT`** festlegen, verwenden Sie mindestens den Wert `10`, um das Auslösen von Fehlalarmen zu verhindern.

Die Berücksichtigung der Statusänderung für das Peerfenster beim Aufheben des Peerstatus für eine Datenbank stellt die Peerfenstersemantik für eine sichere Übernahme in allen Fällen sicher. Bei einem Ausfall der Primärdatenbank während der Statusänderung gilt weiterhin der normale Peerfensterschutz (sichere Übernahme von der Bereitschaftsdatenbank, sofern sie sich weiterhin im Status 'Unterbrochener Peer' befindet).

Auf der Seite der Bereitschaftsdatenbank werden nach dem Trennen der Verbindung die bereits empfangenen Protokolle wiederholt. Nachdem die empfangenen Protokolle wiederholt wurden, stellt die Bereitschaftsdatenbank die Verbindung zur Primärdatenbank wieder her. Nachdem die empfangenen Protokolle wiederholt wurden, stellt die Bereitschaftsdatenbank die Verbindung zur Primärdatenbank wieder her. Bei der Wiederherstellung der Verbindung erfolgt der normale Statusübergang (zuerst in den Status 'Fernes Catch-up', dann in den Peerstatus).

Beziehung zum Datenbankkonfigurationsparameter 'hadr_timeout'

Der Datenbankkonfigurationsparameter **`hadr_timeout`** bewirkt keine Aufhebung des Peerstatus für die Primärdatenbank, wenn die Primärdatenbank während der Blockierung weiterhin Überwachungssignalnachrichten von

der Bereitschaftsdatenbank empfängt. Der Datenbankkonfigurationsparameter **hadr_timeout** gibt ein Zeitlimit für die HADR-Netzebene an. Eine HADR-Datenbank trennt die Verbindung zu ihrer Partnerdatenbank, wenn sie von der Partnerdatenbank innerhalb des durch den Konfigurationsparameter **hadr_timeout** angegebenen Zeitraums keine Nachrichten erhalten hat. Das Zeitlimit für Operationen höherer Ebenen, wie beispielsweise Protokollübertragungs- und Empfangsbestätigungssignale, wird durch dieses Zeitlimit nicht gesteuert. Wenn die Protokollwiederholung in der Bereitschaftsdatenbank bei einer großen Operation, wie beispielsweise dem Laden oder der Reorganisation, blockiert ist, sendet die HADR-Komponente dennoch dem normalen Zeitplan entsprechend Überwachungssignalnachrichten an die Primärdatenbank. In einem solchen Szenario bleibt die Primärdatenbank so lange blockiert wie die Wiederholung in der Bereitschaftsdatenbank, es sei denn, die Registrierdatenbankvariable **DB2_HADR_PEER_WAIT_LIMIT** ist definiert.

Die Registrierdatenbankvariable **DB2_HADR_PEER_WAIT_LIMIT** hebt die Blockierung der Protokollierung in der Primärdatenbank unabhängig vom Verbindungsstatus auf. Auch wenn die Registrierdatenbankvariable **DB2_HADR_PEER_WAIT_LIMIT** nicht definiert ist, wird der Peerstatus für die Primärdatenbank stets aufgehoben, wenn ein Netzfehler festgestellt oder die Verbindung getrennt wird (z. B. als Ergebnis des Konfigurationsparameters **hadr_timeout**).

HADR-Konfigurationsparameter

Einige HADR-Konfigurationsparameter wie beispielsweise **hadr_local_host** und **hadr_remote_host** sind statisch. Statische Parameter werden beim Datenbankstart geladen und Änderungen werden während der Laufzeit ignoriert. HADR-Parameter werden ebenfalls geladen, wenn der Befehl **START HADR** ausgeführt wurde. Auf der Primärdatenbank kann HADR dynamisch gestartet und gestoppt werden, wobei die Datenbank weiterhin online bleibt. Daher besteht eine Möglichkeit zur Aktualisierung des tatsächlichen Werts eines HADR-Konfigurationsparameters ohne Herunterfahren der Datenbank darin, HADR zu stoppen und erneut zu starten. Im Gegensatz dazu führt der Befehl **STOP HADR** auf der Bereitschaftsmaschine dazu, dass die Datenbank heruntergefahren wird. Die Parameter in der Bereitschaftsdatenbank können daher nicht aktualisiert werden, während die Datenbank online ist.

Parameter für Hostnamen und Parameter für Service- und Portnamen (Modus für eine Bereitschaftsdatenbank)

Ein HADR-Paar verfügt über fünf in Wechselbeziehung zueinander stehende Konfigurationsparameter, die Sie definieren sollten:

- **hadr_local_host**
- **hadr_remote_host**
- **hadr_local_svc**
- **hadr_remote_svc**
- **hadr_remote_inst**

Für die Kommunikation zwischen der Primärdatenbank und der Bereitschaftsdatenbank werden TCP-Verbindungen verwendet. Die lokalen Parameter („local") geben die lokale Adresse und die fernen Parameter („remote") geben die ferne Adresse an. Eine Primärdatenbank ist an ihrer lokalen Adresse für neue Verbindungen empfangsbereit. Eine Bereitschaftsdatenbank, die nicht mit einer Primärdatenbank verbunden ist, unternimmt Verbindungsversuche zu ihrer fernen Adresse.

Die Bereitschaftsdatenbank ist auch an ihrer lokalen Adresse empfangsbereit. In einigen Szenarios kann eine andere HADR-Datenbank die Bereitschaftsdatenbank unter dieser Adresse kontaktieren und Nachrichten senden.

Sofern die Registrierdatenbankvariable **HADR_NO_IP_CHECK** nicht definiert ist, gleicht HADR beim Verbindungsaufbau lokale und ferne Adressen wie folgt ab:

```
Meine lokale Adresse = Ihre ferne Adresse
```

und

```
Meine ferne Adresse = Ihre lokale Adresse
```

Dieser Abgleich wird unter Verwendung von IP-Adresse und Portnummer und nicht unter Verwendung der Literalzeichenfolge in den Konfigurationsparametern vorgenommen. Zur Umgehung des Abgleichs muss die Registrierdatenbankvariable **HADR_NO_IP_CHECK** in der NAT-Umgebung (NAT - Network Address Translation) definiert werden.

Eine HADR-Datenbank kann zur Verwendung von IPv4 oder IPv6 zur Lokalisierung der Partnerdatenbank konfiguriert werden. Wenn der Host-Server IPv6 nicht unterstützt, muss IPv4 verwendet werden. Wenn der Server IPv6 unterstützt, hängt die Verwendung von IPv4 oder IPv6 durch die Datenbank von dem Format der Adresse ab, die Sie in den Konfigurationsparametern **hadr_local_host** und **hadr_remote_host** angeben. Die Datenbank versucht, die beiden Parameter in dasselbe IP-Format aufzulösen und nach Möglichkeit IPv6 zu verwenden. Die folgende Tabelle zeigt, wie der IP-Modus durch IPv6-fähige Server bestimmt wird:

Im Parameter hadr_local_host verwendeter IP-Modus	**Im Parameter hadr_remote_host verwendeter IP-Modus**	**Zur HADR-Kommunikation verwendeter IP-Modus**
IPv4-Adresse	IPv4-Adresse	IPv4
IPv4-Adresse	IPv6-Adresse	Fehler
IPv4-Adresse	Hostname, nur in IPv4-Adresse auflösbar	IPv4
IPv4-Adresse	Hostname, nur in IPv6-Adresse auflösbar	Fehler
IPv4-Adresse	Hostname, in IPv4- und v6-Adresse auflösbar	IPv4
IPv6-Adresse	IPv4-Adresse	Fehler
IPv6-Adresse	IPv6-Adresse	IPv6
IPv6-Adresse	Hostname, nur in IPv4-Adresse auflösbar	Fehler
IPv6-Adresse	Hostname, nur in IPv6-Adresse auflösbar	IPv6
IPv6-Adresse	Hostname, in IPv4- und IPv6-Adresse auflösbar	IPv6
Hostname, nur in IPv4-Adresse auflösbar	IPv4-Adresse	IPv4
Hostname, nur in IPv4-Adresse auflösbar	IPv6-Adresse	Fehler

Im Parameter **hadr_local_host** verwendeter IP-Modus	Im Parameter **hadr_remote_host** verwendeter IP-Modus	Zur HADR-Kommunikation verwendeter IP-Modus
Hostname, nur in IPv4-Adresse auflösbar	Hostname, nur in IPv4-Adresse auflösbar	IPv4
Hostname, nur in IPv4-Adresse auflösbar	Hostname, nur in IPv6-Adresse auflösbar	Fehler
Hostname, nur in IPv4-Adresse auflösbar	Hostname, in IPv4- und IPv6-Adresse auflösbar	IPv4
Hostname, nur in IPv6-Adresse auflösbar	IPv4-Adresse	Fehler
Hostname, nur in IPv6-Adresse auflösbar	IPv6-Adresse	IPv6
Hostname, nur in IPv6-Adresse auflösbar	Hostname, nur in IPv4-Adresse auflösbar	Fehler
Hostname, nur in IPv6-Adresse auflösbar	Hostname, nur in IPv6-Adresse auflösbar	IPv6
Hostname, nur in IPv6-Adresse auflösbar	Hostname, in IPv4- und IPv6-Adresse auflösbar	IPv6
Hostname, in IPv4- und IPv6-Adresse auflösbar	IPv4-Adresse	IPv4
Hostname, in IPv4- und IPv6-Adresse auflösbar	IPv6-Adresse	IPv6
Hostname, in IPv4- und IPv6-Adresse auflösbar	Hostname, nur in IPv4-Adresse auflösbar	IPv4
Hostname, in IPv4- und IPv6-Adresse auflösbar	Hostname, nur in IPv6-Adresse auflösbar	IPv6
Hostname, in IPv4- und IPv6-Adresse auflösbar	Hostname, in IPv4- und IPv6-Adresse auflösbar	IPv6

Die Primärdatenbank und die Bereitschaftsdatenbank können HADR-Verbindungen nur herstellen, wenn sie dasselbe IPv4- oder IPv6-Format verwenden. Wenn auf dem einen Server IPv6 eingerichtet ist (jedoch auch IPv4 unterstützt wird) und der andere Server nur IPv4 unterstützt, muss mindestens einer der Parameter **hadr_local_host** und **hadr_remote_host** auf dem IPv6-Server eine IPv4-Adresse angeben, um die Datenbank auf diesem Server zur Verwendung von IPv4 zu veranlassen.

Die Parameter für lokalen und fernen HADR-Service (**hadr_local_svc** und **hadr_remote_svc**) können entweder auf eine Portnummer oder auf einen Servicenamen gesetzt werden. Die angegebenen Werte müssen Ports zugeordnet werden, die von keinem anderen Service verwendet werden, einschließlich anderer DB2-Komponenten oder HADR-Datenbanken. Insbesondere dürfen Sie keinen der Parameterwerte auf den TCP/IP-Port, über den der Server die Kommunikation von fernen Clients erwartet (der Wert des Konfigurationsparameters **svcename** des Datenbankmanagers) oder den nächsten Port (der Wert des Parameters **svcename** + 1) setzen.

Wenn sich die Primärdatenbank und die Bereitschaftsdatenbank auf verschiedenen Servern befinden, können sie die gleiche Portnummer bzw. den gleichen Servicenamen verwenden. Ansonsten müssen unterschiedliche Werte verwendet werden.

Parameter für Hostnamen, Service- oder Portname und Zielliste (Modus für mehrere Bereitschaftsdatenbanken)

Auch im Modus für mehrere Bereitschaftsdatenbanken sollten die Konfigurationsparameter **hadr_local_host**, **hadr_local_svc**, **hadr_remote_host**, **hadr_remote_host** und **hadr_remote_inst** definiert werden. Wenn Sie diese Parameter nicht ordnungsgemäß definieren, werden sie nach der Herstellung einer Verbindung zwischen der Primärdatenbank und den Bereitschaftsdatenbanken unter Verwendung der Einstellungen des Konfigurationsparameters **hadr_target_list** automatisch aktualisiert. Dieser Parameter gibt die Host- und Portnamen aller Bereitschaftsdatenbanken an. Die erste in der Zielliste angegebene Bereitschaftsdatenbank wird als *HADR-Hauptbereitschaftsdatenbank* bezeichnet.

Auch im Modus für mehrere Bereitschaftsdatenbanken sollten die Konfigurationsparameter **hadr_local_host**, **hadr_local_svc**, **hadr_remote_host**, **hadr_remote_host** und **hadr_remote_inst** definiert werden. Die Parameter **hadr_local_host** und **hadr_local_svc** haben dieselbe Bedeutung wie im Modus für eine Bereitschaftsdatenbank. Definieren Sie **hadr_remote_host**, **hadr_remote_host** und **hadr_remote_inst** auf der Primärdatenbank so, dass die zugehörige Hauptbereitschaftsdatenbank angegeben wird. Mit dem neuen Parameter **hadr_target_list** können alle Bereitschaftsdatenbanken aufgelistet werden, wobei der erste Eintrag die Hauptbereitschaftsdatenbank ist. Definieren Sie auf der Bereitschaftsdatenbank die fernen Parameter, um die Primärdatenbank anzugeben. In bestimmten Situationen können die fernen Parameter (auf der Primär- und auf der Bereitschaftsdatenbank) automatisch aktualisiert werden. Weitere Informationen hierzu finden Sie im Abschnitt „Automatische Rekonfiguration von HADR-Parametern" in „Datenbankkonfiguration für mehrere HADR-Bereitschaftsdatenbanken" auf Seite 227.

Synchronisationsmodus

Im Modus für eine Bereitschaftsdatenbank muss der Synchronisationsmodus, der durch den Konfigurationsparameter **hadr_syncmode** angegeben wird, für die Primärdatenbank und die Bereitschaftsdatenbank identisch sein. Die Konsistenz des Werts für diesen Konfigurationsparameter wird überprüft, wenn ein HADR-Paar eine Verbindung herstellt.

Im Modus für mehrere Bereitschaftsdatenbanken muss der Synchronisationsmodus nicht unbedingt identisch sein. Alle Bereitschaftsdatenbanken verfügen über einen *effektiven Synchronisationsmodus*, der sich nach dem Typ der jeweiligen Bereitschaftsdatenbank richtet. Die Hauptbereitschaftsdatenbank verwendet den Synchronisationsmodus der Primärdatenbank und die Nebenbereitschaftsdatenbanken verwenden den Modus SUPERASYNC. Alle Bereitschaftsdatenbanken verfügen über einen *konfigurierten Synchronisationsmodus*, der der expliziten Einstellung für **hadr_syncmode** entspricht und verwendet wird, wenn eine Bereitschaftsdatenbank die Funktion der neuen Primärdatenbank übernimmt.

Ausführliche Informationen hierzu finden Sie in „HADR-Synchronisationsmodus (High Availability Disaster Recovery)".

HADR-Zeitlimit und Peerfenster

Das über den Konfigurationsparameter **hadr_timeout** angegebene Zeitlimitintervall muss für die Primärdatenbank und die Bereitschaftsdatenbank identisch sein. Die Konsistenz der Werte dieser Konfigurationsparameter wird überprüft, wenn ein HADR-Paar eine Verbindung herstellt.

Mit einer Ausnahme wartet die Primärdatenbank beim Start für den längeren der beiden folgenden Zeiträume auf die Verbindung zu einer Bereitschaftsdatenbank:

- Mindestens 30 Sekunden
- Für die Anzahl der durch den Datenbankkonfigurationsparameter **hadr_timeout** angegebenen Sekunden.

Wenn die Bereitschaftsdatenbank innerhalb des angegebenen Zeitraums keine Verbindung herstellt, schlägt der Start fehl. Die einzige Ausnahme gilt für den Fall, dass der Befehl **START HADR** mit dem Parameter **BY FORCE** abgesetzt wird. Dann startet die Primärdatenbank, ohne auf die Verbindung zur Bereitschaftsdatenbank zu warten.

Im Modus für mehrere Bereitschaftsdatenbanken wartet die Primärdatenbank nur auf die Verbindung zur Hauptbereitschaftsdatenbank. Die Verbindung zu einer Nebenbereitschaftsdatenbank ist optional.

Nach der Herstellung einer Verbindung zwischen einem HADR-Paar tauschen die Datenbanken Überwachungssignalnachrichten aus. Das Intervall der Überwachungssignale wird anhand von Faktoren wie die Werte der Konfigurationsparameter **hadr_timeout** und **hadr_peer_window** berechnet. Es wird im Feld HEARTBEAT_INTERVAL der Tabellenfunktion MON_GET_HADR angegeben. Wenn eine Datenbank überhaupt keine Nachricht von der anderen Datenbank innerhalb der durch den Konfigurationsparameter **hadr_timeout** definierten Sekunden empfängt, leitet sie eine Trennung der Verbindung ein. Dies bedeutet, dass es höchstens die im Konfigurationsparameter **hadr_timeout** angegebene Anzahl von Sekunden dauert, bis eine HADR-Datenbank den Ausfall entweder der Partnerdatenbank oder des verbindenden Netzes erkennt. Wenn Sie den Konfigurationsparameter **hadr_timeout** zu niedrig einstellen, sind Fehlalarme und häufige Trennungen von Verbindungen die Folge.

Wenn Sie den Konfigurationsparameter **hadr_peer_window** auf einen Wert ungleich null gesetzt haben und die Primärdatenbank die Verbindung zur Bereitschaftsdatenbank im Peerstatus verliert, werden Transaktionen von der Primärdatenbank erst dann festgeschrieben, wenn die Verbindung mit der Bereitschaftsdatenbank wiederhergestellt ist bzw. der Zeitwert des Konfigurationsparameters **hadr_peer_window** abläuft.

Für maximale Verfügbarkeit wird der Standardwert für den Datenbankkonfigurationsparameter **hadr_peer_window** auf 0 gesetzt. Wenn dieser Parameter auf 0 gesetzt ist, verlässt die Primärdatenbank den Peerstatus, sobald die Verbindung zwischen der Primär- und der Bereitschaftsdatenbank geschlossen wurde, um eine Blockierung von Transaktionen zu vermeiden. Die Verbindung kann geschlossen werden, weil die Bereitschaftsdatenbank die Verbindung geschlossen hat, ein Netzfehler festgestellt oder das Zeitlimit erreicht wurde. Für höhere Datenkonsistenz (was jedoch eine geringere Verfügbarkeit zur Folge hat) können Sie den Datenbankkonfigurationsparameter **hadr_peer_window** auf einen Wert ungleich null setzen.

Weitere Informationen finden Sie in „Konfigurieren der Datenbankkonfigurationsparameter **hadr_timeout** und **hadr_peer_window**".

Im Folgenden sehen Sie eine Beispielkonfiguration für die Primärdatenbank und die Bereitschaftsdatenbank:

Primärdatenbank:

```
HADR_LOCAL_HOST           host1.ibm.com
HADR_LOCAL_SVC            hadr_service
HADR_REMOTE_HOST          host2.ibm.com
HADR_REMOTE_SVC           hadr_service
HADR_REMOTE_INST          dbinst2
HADR_TIMEOUT              120
HADR_SYNCMODE             NEARSYNC   HADR_PEER_WINDOW          120
```

Bereitschaftsdatenbank:

```
HADR_LOCAL_HOST           host2.ibm.com
HADR_LOCAL_SVC            hadr_service
HADR_REMOTE_HOST          host1.ibm.com
HADR_REMOTE_SVC           hadr_service
HADR_REMOTE_INST          dbinst1
HADR_TIMEOUT              120
HADR_SYNCMODE             NEARSYNC   HADR_PEER_WINDOW          120
```

Konfigurieren der Datenbankkonfigurationsparameter 'hadr_timeout' und 'hadr_peer_window'

Sie können die Datenbankkonfigurationsparameter **hadr_timeout** und **hadr_peer_window** so konfigurieren, dass Sie bei einem Verbindungsfehler eine optimale Reaktion erhalten.

Konfigurationsparameter hadr_timeout

Wenn eine HADR-Datenbank für länger als ein Zehntel der Zeit, die durch den Datenbankkonfigurationsparameter **hadr_timeout** vorgegeben ist, keine Kommunikation von ihrer Partnerdatenbank erhält, geht die Datenbank davon aus, dass die Verbindung zur Partnerdatenbank unterbrochen wurde. Wenn sich die Datenbank zum Zeitpunkt der Verbindungsunterbrechung im Peerzustand befindet, geht sie in den Status 'Unterbrochener Peer' (der Wert des Datenbankkonfigurationsparameters **hadr_peer_window** ist größer als Null) oder in den Status 'Fernes Catch-up anstehend' (der Wert von **hadr_peer_window** ist nicht größer als Null) über. Die Statusänderung betrifft sowohl Primär- als auch Bereitschaftsdatenbanken.

Konfigurationsparameter hadr_peer_window

Der Konfigurationsparameter **hadr_peer_window** ersetzt den Konfigurationsparameter **hadr_timeout** nicht. Der Konfigurationsparameter **hadr_timeout** bestimmt, wie lange eine HADR-Datenbank wartet, bevor sie davon ausgeht, dass ihre Verbindung zur Partnerdatenbank unterbrochen wurde. Der Konfigurationsparameter **hadr_peer_window** bestimmt, ob die Datenbank in den Status 'Unterbrochener Peer' übergeht, wenn die Verbindung unterbrochen wird, und wie lange die Datenbank in diesem Status verbleiben soll. HADR unterbricht die Verbindung, wenn ein Netzfehler während eines Sende-, Empfangs- oder Polling-Vorgangs am TCP-Socket festgestellt wird. HADR fragt das Socket alle 100 Millisekunden ab. Dadurch kann rasch auf Netzfehler reagiert werden, die vom Betriebssystem erkannt werden. Nur im ungünstigsten Fall wartet HADR bis zur Zeitlimitüberschreitung, um eine fehlerhafte Verbindung zu trennen. In diesem Fall kann eine Datenbankanwendung, die bei Auftreten des Fehlers ausgeführt wird, für einen bestimmten Zeitraum blockiert werden, der der Summe der Datenbankkonfigurationsparameter **hadr_timeout** und **hadr_peer_window** entspricht.

Konfigurieren der Datenbankkonfigurationsparameter hadr_timeout und hadr_peer_window

Es ist wünschenswert, die Wartezeit für eine Datenbankanwendung zu minimieren. Wenn für die Konfigurationsparameter **hadr_timeout** und **hadr_peer_window** niedrige Werte angegeben werden, verkürzt dies die

Wartezeit für die Datenbankanwendung, wenn die Verbindung einer HA-DR-Bereitschaftsdatenbank zur Primärdatenbank unterbrochen wird. Bei der Festlegung der Werte für die Konfigurationsparameter **hadr_timeout** und **hadr_peer_window** sollte aber Folgendes beachtet werden:

- Der Datenbankkonfigurationsparameter **hadr_timeout** sollte auf einen Wert gesetzt werden, der lang genug ist, um Fehlalarme bei der HADR-Verbindung zu vermeiden, die durch kurze vorübergehende Netzunterbrechungen verursacht werden. Beispielsweise beträgt der Standardwert von **hadr_timeout** 120 Sekunden - ein Wert, der für viele Netze geeignet ist.
- Der Datenbankkonfigurationsparameter **hadr_peer_window** sollte auf einen Wert eingestellt werden, der lang genug ist, um es dem System zu ermöglichen, beim Auftreten von Fehler automatisch zu intervenieren. Wenn das HA-System - z. B. ein Cluster-Manager - den Primärdatenbankfehler erkennt, bevor der Status 'Unterbrochener Peer' endet, erfolgt eine Funktionsübernahme durch die Bereitschaftsdatenbank. Bei der Funktionsübernahme gehen keine Daten verloren, da alle Daten von der alten Primärdatenbank in die neue Primärdatenbank repliziert werden. Wenn der Wert für **hadr_peer_window** zu kurz ist, hat das HA-System möglicherweise nicht genug Zeit, um den Fehler zu erkennen und entsprechend zu intervenieren.

  Anmerkung: Im Modus für mehrere HADR-Bereitschaftsdatenbanken verwendet die Hauptbereitschaftsdatenbank für **hadr_peer_window** (das *effektive Peerfenster*) die Einstellung der Primärdatenbank. Die Einstellung für **hadr_peer_window** ist für Nebenbereitschaftsdatenbanken ohne Bedeutung, da dieser Typ von Bereitschaftsdatenbank immer im Modus SUPERASYNC ausgeführt wird.

HADR-Protokollspooling

Die Funktion für das HADR-Protokollspooling ermöglicht die fortschreitende Ausführung von Transaktionen auf der Primärdatenbank, ohne auf die Protokollwiedergabe auf der Bereitschaftsdatenbank warten zu müssen.

Bei Aktivierung dieser Funktion werden von der Primärdatenbank gesendete Protokolldaten in der Bereitschaftsdatenbank auf Platte *gespoolt* bzw. geschrieben. Diese Protokolldaten werden dann später von der Protokollwiedergabe gelesen.

Das Protokollspooling, das über den Datenbankkonfigurationsparameter **hadr_spool_limit** aktiviert wird, stellt eine Verbesserung der HADR-Funktion dar. Wenn die Wiedergabe langsam ist, werden neue Transaktionen auf der Primärdatenbank möglicherweise blockiert, weil die Primärdatenbank keine Protokolldaten an das Bereitschaftssystem senden kann, wenn im Puffer kein Platz für den Empfang von Daten vorhanden ist. Bei Verwendung der Protokollspoolingfunktion ist die Bereitschaftsdatenbank nicht durch die Größe des Puffers eingeschränkt. Wenn eine größere Menge Daten empfangen wird, die nicht in den Puffer gestellt werden können, liest die Protokollwiedergabe die Daten von Platte. Dies hat zur Folge, dass das System Spitzen im Transaktionsvolumen auf der Primärdatenbank oder eine Verlangsamung der Protokollwiedergabe (aufgrund der Wiedergabe eines bestimmten Typs von Protokollsätzen) auf der Bereitschaftsdatenbank besser ausgleichen kann.

Diese Funktion führt möglicherweise zu einem größeren Abstand zwischen der Protokollposition der empfangenen Protokolle auf der Bereitschaftsdatenbank und der Protokollwiedergabeposition auf der Bereitschaftsdatenbank, was eine längere Übernahmezeit zur Folge haben kann. Überlegen Sie genau, wie Sie die Einstellung

für das Spoollimit festlegen möchten, da die alte Bereitschaftsdatenbank erst als neue Primärdatenbank gestartet werden und Transaktionen empfangen kann, wenn die Wiedergabe der gespoolten Protokolle beendet wurde.

Konfiguration der Protokollarchivierung für DB2 High Availability Disaster Recovery (HADR)

Konfigurieren Sie für die Verwendung der Protokollarchivierung mit DB2 High Availability Disaster Recovery (HADR) die Primärdatenbank und die Bereitschaftsdatenbank so, dass die Funktion für das automatische Abrufen von Protokollen aus allen Protokollarchivpositionen eingerichtet ist. Konfigurieren Sie die Archivierung bei Systemen mit mehreren Bereitschaftsdatenbanken für die Primärdatenbank und alle Bereitschaftsdatenbanken.

Nur die aktuelle Primärdatenbank kann eine Protokollarchivierung ausführen. Wenn die Primär- und die Bereitschaftsdatenbank mit separaten Archivierungspositionen konfiguriert sind, werden nur an der Archivierungsposition der Primärdatenbank Protokolle archiviert. Im Falle einer Übernahme wird die Bereitschaftsdatenbank zur neuen Primärdatenbank. Alle nachfolgend archivierten Protokolle werden an der Archivierungsposition der ursprünglichen Bereitschaftsdatenbank gespeichert. In einer solchen Konfiguration werden Protokolle entweder an der einen oder an der anderen Position gespeichert, jedoch nicht an beiden. Eine Ausnahme ist möglich, wenn die neue Primärdatenbank nach einer Übernahme einige wenige Protokolle archiviert, die von der ursprünglichen Primärdatenbank bereits archiviert wurden. In einem System mit mehreren Bereitschaftsdatenbanken können die archivierten Protokolldateien über die Archivierungseinheiten aller Datenbanken (Primärdatenbank und Bereitschaftsdatenbanken) hinweg verteilt sein. Ein gemeinsam genutztes Archiv ist von Vorteil, da alle Dateien in einer einzigen Position gespeichert werden.

Archivierte Protokolldateien müssen von zahlreichen Operationen abgerufen werden. Zu diesen Operationen gehören die aktualisierende Recovery einer Datenbank, das Abrufen von Protokolldateien durch die HADR-Primärdatenbank, um diese an die Bereitschaftsdatenbank im Status 'Fernes Catch-up' zu senden, oder Protokolle lesende Replikationsprogramme (z. B. Q Replication). Deshalb ist für ein HADR-System ein gemeinsam genutztes Archiv von Vorteil, da die erforderlichen Dateien ansonsten über mehrere Archivierungseinheiten hinweg verteilt sein können und Benutzereingriffe erforderlich sind, um die benötigten Dateien zu suchen und in die anfordernde Datenbank zu kopieren. Die empfohlene Zieladresse für den Kopiervorgang ist eine Archivierungseinheit. Wenn das Kopieren in ein Archiv nicht möglich ist, kopieren Sie die Protokolle in den Überlaufprotokollpfad. Als letzte Ausweichmöglichkeit können Sie die Protokolle in den Protokollpfad kopieren (wobei Sie sich darüber im Klaren sein müssen, dass die aktiven Protokolldateien möglicherweise beschädigt werden). DB2 löscht die von einem Benutzer in den Überlaufprotokollpfad und den Pfad für aktive Protokolldateien kopierten Dateien nicht automatisch. Deshalb müssen diese Dateien manuell entfernt werden, wenn sie von keinen HADR-Bereitschaftsdatenbanken oder Anwendungen mehr benötigt werden.

Ein mögliches Szenario in diesem Fall wäre eine Übernahme im Modus für mehrere Bereitschaftsdatenbanken. Nach der Übernahme verfügt die neue Primärdatenbank möglicherweise nicht über alle Protokolldateien, die von anderen Bereitschaftsdatenbanken benötigt werden (weil sich eine Bereitschaftsdatenbank an einer älteren Protokollposition befindet). Wenn die Primärdatenbank eine angeforderte Protokolldatei nicht finden kann, wird die Bereitschaftsdatenbank entsprechend benachrichtigt. Die Bereitschaftsdatenbank schließt die Verbindung und stellt

die Verbindung nach einigen Sekunden für einen neuen Versuch wieder her. Das Wiederholungsintervall ist auf einige Minuten beschränkt. Sobald das Wiederholungsintervall abgelaufen ist, wird die Bereitschaftsdatenbank heruntergefahren. In diesem Fall sollten Sie die Dateien in die Primärdatenbank kopieren, um sicherzustellen, dass alle Dateien von der ersten fehlenden Datei bis zur aktuellen Protokolldatei vorhanden sind. Führen Sie nach dem Kopieren der Dateien gegebenenfalls einen Neustart für die Bereitschaftsdatenbank durch.

Die Bereitschaftsdatenbank verwaltet automatisch Protokolldateien in ihrem Protokollpfad. Die Bereitschaftsdatenbank löscht keine Protokolldatei aus ihrem lokalen Protokollpfad, bis sie von der Primärdatenbank benachrichtigt wurde, das die Primärdatenbank diese archiviert hat. Diese Funktionsweise stellt einen zusätzlichen Schutz gegen den Verlust von Protokolldateien bereit. Wenn die Primärdatenbank ausfällt und ihre Protokollplatte beschädigt wird, bevor eine bestimmte Protokolldatei in der Primärdatenbank archiviert wurde, löscht die Bereitschaftsdatenbank diese Protokolldatei von ihrer eigenen Platte nicht, weil sie keine Benachrichtigung empfangen hat, dass die Primärdatenbank die Protokolldatei erfolgreich archiviert hat. Wenn die Bereitschaftsdatenbank später als neue Primärdatenbank übernimmt, archiviert sie diese Protokolldatei, bevor sie die Datei wieder freigibt. Wenn sowohl der Konfigurationsparameter **logarchmeth1** als auch der Konfigurationsparameter **logarchmeth2** verwendet werden, gibt die Bereitschaftsdatenbank eine Protokolldatei erst frei, wenn die Primärdatenbank sie mit beiden Methoden archiviert hat.

Zusätzlich zu den bereits genannten Vorteilen verbessert eine gemeinsam genutzte Protokollarchivierungseinheit den Catch-up-Prozess, da die Bereitschaftsdatenbank ältere Protokolldateien direkt aus dem Archiv im lokalen Catch-up-Status abrufen kann, anstatt diese Dateien indirekt über die Primärdatenbank im fernen Catch-up-Status abzurufen. Es wird jedoch empfohlen, keine serielle Archivierungseinheit wie z. B. ein Bandlaufwerk für HADR-Datenbanken zu verwenden. Bei seriellen Einheiten kann es aufgrund von gemischten Lese- und Schreiboperationen sowohl auf der Primär- als auch auf der Bereitschaftsdatenbank zu Leistungseinbußen kommen. Die Primärdatenbank schreibt auf die Einheit, wenn sie Protokolldateien archiviert, und die Bereitschaftsdatenbank liest von der Einheit, wenn sie Protokolle wiedergibt. Diese Leistungsbeeinträchtigung kann auch dann auftreten, wenn die Einheit nicht als gemeinsam genutzt konfiguriert ist.

Gemeinsam genutzte Protokollarchive in Tivoli Storage Manager

Durch die Verwendung eines gemeinsam genutzten Protokollarchivs in IBM Tivoli Storage Manager (TSM) können ein oder mehrere Knoten für den TSM-Server als einzelner Knoten dargestellt werden. Dies ist vor allem in einer HADR-Umgebung nützlich, in der jede Maschine zu jedem beliebigen Zeitpunkt das Primärsystem sein kann.

Zum Einrichten eines gemeinsam genutzten Protokollarchivs müssen Sie Proxy-Knoten verwenden, die es den TSM-Clientknoten ermöglichen, Datenschutzoperationen für einen zentralen Namespace auf dem TSM-Server durchzuführen. Der Zielclientknoten ist Eigner der Daten und Agentenknoten agieren im Auftrag der Zielknoten, um die Backup-Daten zu verwalten. Das Proxy-Knotenziel ist der auf dem TSM-Server definierte Knotenname, dem die Backup-Versionen der verteilten Daten zugeordnet werden. Die Daten werden in einem einzelnen Namespace auf dem TSM-Server verwaltet, als ob sie alle zu diesem einen Knoten gehören. Der Name des Proxy-Knotenziels kann ein realer Knoten (z. B. einer der Anwendungshosts) oder ein virtueller Knotenname (d. h. ohne zugehörigen physischen Knoten) sein. Verwenden Sie die folgenden Befehle auf dem TSM-Server, um einen virtuellen Proxy-Knotennamen zu erstellen:

```
Grant proxynode target=virtueller Knotenname agent=Name der HADR-Primärdatenbank
Grant proxynode target=virtueller Knotenname agent=Name der HADR-Bereitschaftsdatenbank
```

Anschließend müssen Sie für die folgenden Datenbankkonfigurationsparameter in der Primär- und der Bereitschaftsdatenbank den Wert von *virtueller Knotenname* definieren:

- **vendoropt**
- **logarchopt**

In einer Konfiguration mit mehreren Bereitschaftsdatenbanken müssen Sie den Proxy-Knotenzugriff allen Maschinen des TSM-Servers erteilen und die Konfigurationsparameter **vendoropt** und **logarchopt** auf allen Bereitschaftsdatenbanken definieren.

HADR-Leistung

Das Konfigurieren verschiedener Aspekte Ihres Datenbanksystems einschließlich der Netzbandbreite, der Leistung der Systemeinheit und der Puffergröße kann die Leistung Ihrer DB2-HADR-Datenbanken verbessern.

Das Netz ist der wichtigste Bestandteil Ihrer HADR-Konfiguration, da zum Replizieren von Datenbankänderungen von der Primärdatenbank in die Bereitschaftsdatenbank zur Synchronisation der beiden Datenbanken eine Netzkonnektivität erforderlich ist.

Empfehlungen für eine optimale Netzleistung:

- Die Netzbandbreite muss größer als die Generierungsgeschwindigkeit des Datenbankprotokolls sein.
- Berücksichtigen Sie bei der Auswahl des HADR-Synchronisationsmodus mögliche Verzögerungen bei der Netzübertragung. Verzögerungen bei der Netzübertragung wirken sich nur auf die Modi SYNC und NEARSYNC aus.

  Die Verlangsamung der Systemleistung infolge des SYNC-Modus kann erheblich größer sein als die der anderen Synchronisationsmodi. Im SYNC-Modus sendet die Primärdatenbank Protokollseiten an die Bereitschaftsdatenbank erst, nachdem die Protokollseiten erfolgreich auf die Protokollplatte der Primärdatenbank geschrieben wurden. Zum Schutz der Integrität des Systems wartet die Primärdatenbank auf eine Bestätigung von der Bereitschaftsdatenbank, bevor sie eine Anwendung benachrichtigt, dass eine Transaktion vorbereitet (prepared) oder festgeschrieben (committed) wurde. Die Bereitschaftsdatenbank sendet die Bestätigung erst, nachdem sie die empfangenen Protokollseiten auf ihre Platte, d. h. die Platte der Bereitschaftsdatenbank, geschrieben hat. Die Leistungseinbuße ist gleichzusetzen mit der Zeit, die für das Schreiben von Protokollseiten in der Bereitschaftsdatenbank erforderlich ist, plus die Zeit, die für das Senden der Nachrichten zurück an die Primärdatenbank erforderlich ist.

  Im NEARSYNC-Modus schreibt die Primärdatenbank Protokollseiten und sendet sie gleichzeitig in einer parallelen Operation. Anschließend wartet die Primärdatenbank auf die Bestätigung von der Bereitschaftsdatenbank. Die Bereitschaftsdatenbank sendet die Bestätigung, sobald sie die Protokollseiten in ihrem Speicher empfangen hat. In einem schnellen Netz ist der Zeitaufwand für die Primärdatenbank minimal. Die Bestätigung kann bereits eingetroffen sein, wenn die Primärdatenbank das lokale Schreiben ihrer Protokolle beendet.

  Beim ASYNC-Modus erfolgen das Schreiben und Senden von Protokollseiten ebenfalls parallel. In diesem Modus wartet die Primärdatenbank jedoch nicht auf eine Bestätigung von der Bereitschaftsdatenbank. Daher spielen Verzöge-

rungen bei der Netzübertragung keine Rolle. Die Leistungseinbußen sind beim ASYNC-Modus sogar noch geringer als beim NEARSYNC-Modus.

Was den Modus SUPERASYNC betrifft, so werden Transaktionen niemals geblockt; es kommt auch niemals zu ausgedehnten Antwortzeiten aufgrund von Netzunterbrechungen oder Überlastung. Neue Transaktionen können verarbeitet werden, sobald zuvor übergebene Transaktionen in die Primärdatenbank geschrieben werden. Daher spielen Verzögerungen bei der Netzübertragung keine Rolle. Die verstrichene Zeit für den Abschluss nicht erzwungener Übernahmeoperationen kann länger als in anderen Modi dauern, da die Protokollabstimmungsdiskrepanz zwischen der Primär- und der Bereitschaftsdatenbank möglicherweise relativ groß ist.

- Sie sollten die Registry-Variablen **DB2_HADR_SOSNDBUF** und **DB2_HADR_SORCVBUF** optimieren.

  Die HADR-Protokollübertragungsauslastung, die Netzbandbreite und die Übertragungsverzögerung sind wichtige Faktoren, die hierbei berücksichtigt werden müssen. Mithilfe der beiden Registry-Variablen **DB2_HADR_SOSNDBUF** und **DB2_HADR_SORCVBUF** können die TCP-Socketsendepuffergröße und die TCP-Socketempfangspuffergröße speziell für HADR-Verbindungen optimiert werden. Der Wertebereich dieser beiden Variablen ist `1024` bis `4294967295`. Der Standardwert ist die Socketpuffergröße des Betriebssystems, die je nach Betriebssystem variiert. Es wird dringend empfohlen, einen Mindestwert von `16384` (16 K) für die Einstellungen **DB2_HADR_SOSNDBUF** und **DB2_HADR_SORCVBUF** zu verwenden. Auf manchen Betriebssystemen wird der vom Benutzer angegebenen Wert automatisch gerundet oder begrenzt.

Sie können den HADR-Simulator (ein Befehlszeilentool, das eine simulierte HADR-Arbeitslast erstellt) zum Messen der Netzleistung und zum Testen verschiedener Netzoptimierungsoptionen verwenden. Der Simulator steht im Web unter http://www.ibm.com/developerworks/wikis/display/data/HADR_sim zum Download zur Verfügung.

Netzüberlastung

Für jeden Schreibvorgang von Protokollseiten in der Primärdatenbank werden diese Protokollseiten auch an die Bereitschaftsdatenbank gesendet. Jede Schreiboperation wird als *Flushoperation* bezeichnet. Der Umfang der Flushoperation ist auf die Protokollpuffergröße in der Primärdatenbank begrenzt (die durch den Datenbankkonfigurationsparameter **logbufsz** gesteuert wird). Die exakte Größe der einzelnen Flushoperationen ist nicht deterministisch. Ein größerer Protokollpuffer führt nicht zwangsläufig zu einem größeren Flushvolumen.

Wenn die Bereitschaftsdatenbank bei der Wiederholung der Protokollseiten zu langsam arbeitet, füllt sich der Protokollempfangspuffer möglicherweise vollständig an, sodass der Puffer keine weiteren Protokollseiten empfangen kann. Wenn die Datenbanken im SYNC- oder NEARSYNC-Modus arbeiten und die Primärdatenbank für ihren Protokollpuffer ein weiteres Mal eine Flushoperation ausführt, werden die Daten wahrscheinlich in der Netzpipeline gepuffert, die aus der primären Maschine, dem Netz und der Bereitschaftsdatenbank besteht. Da die Bereitschaftsdatenbank keinen freien Puffer zum Empfang der Daten hat, kann sie keine Bestätigung senden, sodass die Primärdatenbank blockiert wird, solange sie auf die Bestätigung der Bereitschaftsdatenbank wartet.

Im ASYNC-Modus fährt die Primärdatenbank mit dem Senden von Protokollseiten fort, bis sich die Pipeline gänzlich füllt und sie keine weiteren Protokollseiten senden kann. Eine solche Bedingung wird als *Überlastung* bezeichnet. Eine Überlastung wird durch das Monitorelement **hadr_connect_status** gemeldet. In den Modi

SYNC und NEARSYNC kann die Pipeline in der Regel eine einzelne Flushoperation auffangen, sodass keine Überlastung auftritt. Die Primärdatenbank bleibt jedoch blockiert, solange sie auf eine Bestätigung von der Bereitschaftsdatenbank für die Flushoperation wartet.

Eine Überlastung kann darüber hinaus auftreten, wenn die Bereitschaftsdatenbank Protokollsätze anwendet, deren Anwendung zeitaufwendig ist, wie zum Beispiel Protokollsätze für eine Datenbank- oder Tabellenreorganisation.

Im Modus SUPERASYNC vergrößert sich die Protokollabstimmungsdiskrepanz zwischen der Primär- und der Bereitschaftsdatenbank möglicherweise stetig, da die Commitoperationen für Transaktionen in der Primärdatenbank nicht von der relativen Langsamkeit des HADR-Netz- oder des HADR-Bereitschaftsservers betroffen sind. Es ist wichtig, die Protokollabstimmungsdiskrepanz zu überwachen, da es sich um eine indirekte Kennzahl der potenziellen Anzahl von Transaktionen handelt, die möglicherweise im Fall eines echten Störfalls auf dem Primärsystem verloren gehen könnten. Bei Szenarios zur Recovery nach einem Katastrophenfall würden sämtliche Transaktionen, die bei der Protokollabstimmungsdiskrepanz festgeschrieben wurden, der Bereitschaftsdatenbank nicht zur Verfügung stehen. Überwachen Sie die Protokollabstimmungsdiskrepanz daher mit dem Monitorelement **hadr_log_gap**; falls die Protokollabstimmungsdiskrepanz nicht zulässig ist, überprüfen Sie die Netzunterbrechungen oder die jeweilige Geschwindigkeit des HADR-Bereitschaftsservers und ergreifen Sie zur Reduzierung der Protokollabstimmungsdiskrepanz korrigierende Maßnahmen.

Empfehlungen für das Minimieren der Netzüberlastung:

- Die Leistungsfähigkeit der Bereitschaftsdatenbank sollte ausreichen, um die protokollierten Operationen der Datenbank ebenso schnell nachvollziehen zu können, wie sie in der Primärdatenbank generiert werden. Daher wird für die Primär- und die Bereitschaftsdatenbank eine identische Hardwareausstattung empfohlen.
- Sie sollten die Größe des Protokollempfangspuffers der Bereitschaftsdatenbank mithilfe der Registry-Variablen **DB2_HADR_BUF_SIZE** optimieren.

  Ein größerer Puffer kann helfen, die Wahrscheinlichkeit einer Überlastung zu verringern, obwohl er sehr wahrscheinlich nicht alle Ursachen einer Überlastung beseitigen kann. Standardmäßig beträgt die Größe des Protokollempfangspuffers der Bereitschaftsdatenbank das Zweifache der Größe des Protokollschreibpuffers der Primärdatenbank. Der Datenbankkonfigurationsparameter **logbufsz** definiert die Größe des Protokollschreibpuffers der Primärdatenbank.

  Sie können ermitteln, ob die Größe des Protokollempfangspuffers der Bereitschaftsdatenbank unangemessen ist, indem Sie den Befehl **db2pd** mit der Option `-hadr` verwenden. Wenn der Wert für den Parameter `StandByRcvBufUsed`, der den Prozentsatz des verwendeten Protokollempfangspuffers der Bereitschaftsdatenbank anzeigt, fast 100 beträgt, sollten Sie die Registry-Variable **DB2_HADR_BUF_SIZE** erhöhen.
- Es wird empfohlen, die Registry-Variable **DB2_HADR_PEER_WAIT_LIMIT** zu setzen, mit der eine Blockierung der Protokollierung in der Primärdatenbank (zurückzuführen auf eine langsame oder blockierte Bereitschaftsdatenbank) verhindert werden kann.

  Durch das Setzen der Registry-Variablen **DB2_HADR_PEER_WAIT_LIMIT** wird für die HADR-Primärdatenbank der Peerstatus aufgehoben, wenn die Protokollierung in der Primärdatenbank für die angegebene Anzahl von Sekunden blockiert wurde, weil das Protokoll in die Bereitschaftsdatenbank repliziert wird. Wenn dieser Grenzwert erreicht wird, unterbricht die Primärdatenbank

die Verbindung zur Bereitschaftsdatenbank. Wenn das Peerfenster inaktiviert ist, wird die Primärdatenbank in den Unterbrechungsstatus versetzt, und die Protokollierung wird wieder aufgenommen. Wenn das Peerfenster aktiviert ist, wird die Primärdatenbank in den Status 'Unterbrochener Peer' versetzt, in dem die Protokollierung weiterhin blockiert ist. Der Status 'Unterbrochener Peer' wird für die Primärdatenbank aufgehoben, sobald die Verbindung erneut hergestellt wird oder das Peerfenster abgelaufen ist. Die Protokollierung wird wieder aufgenommen, sobald der Status 'Unterbrochener Peer' für die Primärdatenbank aufgehoben ist.

Anmerkung: Falls Sie **DB2_HADR_PEER_WAIT_LIMIT** festlegen, verwenden Sie mindestens den Wert 10, um das Auslösen von Fehlalarmen zu verhindern.

Die Berücksichtigung der Statusänderung für das Peerfenster beim Aufheben des Peerstatus stellt die Peerfenstersemantik für eine sichere Übernahme in allen Fällen sicher. Bei einem Ausfall der Primärdatenbank während der Statusänderung gilt weiterhin der normale Peerfensterschutz (sichere Übernahme von der Bereitschaftsdatenbank, während sie sich im Status 'Unterbrochener Peer' befindet).

- In den meisten Systemen wird die Protokollierungsfunktion nicht bis zur Kapazitätsgrenze ausgenutzt. Selbst im SYNC-Modus lässt sich vielleicht keine Verlangsamung in der Primärdatenbank beobachten. Wenn die Protokollierkapazität bei aktiviertem HADR zum Beispiel 40 Mb pro Sekunde beträgt, das System vor der Aktivierung von HADR jedoch nur mit 30 Mb arbeitete, bemerken Sie möglicherweise keinen Unterschied in der Gesamtsystemleistung.
- Um den Catch-up-Prozess zu beschleunigen, können Sie eine gemeinsam genutzte Protokollarchivierungseinheit verwenden. Wenn die gemeinsam genutzte Einheit jedoch eine serielle Einheit ist (z. B. ein Bandlaufwerk), kann durch die gemischten Lese- und Schreibvorgänge die Leistung sowohl auf der Primär- als auch auf der Bereitschaftsdatenbank verringert werden.
- Wenn Sie die Funktion für Leseoperationen in der Bereitschaftsdatenbank nutzen möchten, muss die Bereitschaftsdatenbank über die Ressourcen verfügen, die für die zusätzlichen Verarbeitungsprozesse erforderlich sind.
- Wenn Sie die Funktion für Leseoperationen in der Bereitschaftsdatenbank nutzen möchten, konfigurieren Sie die Pufferpools der Primärdatenbank. Diese Informationen werden dann mithilfe von Protokollen an die Primärdatenbank übertragen.
- Wenn Sie die Funktion für Leseoperationen in der Bereitschaftsdatenbank nutzen möchten, optimieren Sie die Konfigurationsparameter **pckcachesz**, **catalogcache_sz**, **applheapsz** und **sortheap** in der Bereitschaftsdatenbank.

Auf der folgenden Website finden Sie Informationen zu den neuesten Updates zum Thema 'Verbesserung der HADR-Leistung': http://www.ibm.com/developerworks/wikis/display/data/HADR_tune.

Cluster-Manager und HADR (High Availability Disaster Recovery)

Sie können DB2-HADR-Datenbanken auf den Knoten eines Clusters implementieren und einen Cluster-Manager verwenden, um die Verfügbarkeit Ihrer Datenbanklösung zu verbessern.

Sie können die Primärdatenbank und die Bereitschaftsdatenbank entweder von demselben oder von unterschiedlichen Cluster-Managern verwalten lassen.

Konfigurieren eines HADR-Paares, bei dem die Primär- und die Bereitschaftsdatenbank von demselben Cluster-Manager verwaltet werden

Diese Konfiguration eignet sich am besten für Umgebungen, in denen sich die Primärdatenbank und die Bereitschaftsdatenbank am selben Standort befinden und die schnellstmögliche Funktionsübernahme erforderlich ist. Das Verwenden von HADR bietet diesen Umgebungen den Vorteil, dass die DBMS-Verfügbarkeit gewährleistet wird statt Recovery nach einem Systemabsturz oder andere Recoverymethoden zu verwenden.

Mit dem Cluster-Manager können Sie auf schnelle Weise Probleme aufspüren und eine Übernahmeoperation einleiten. Da HADR separaten Speicher für das DBMS benötigt, sollte der Cluster-Manager mit separater Datenträgersteuerung konfiguriert werden. Diese Konfiguration verhindert, dass der Cluster-Manager auf das Ausführen einer Funktionsübernahme auf dem Datenträger wartet, bevor er das DBMS auf dem Bereitschaftssystem verwendet. Sie können die automatische Clientweiterleitung verwenden, um Clientanwendungen an die neue Primärdatenbank umzuleiten.

Konfigurieren eines HADR-Paares, bei dem die Primär- und die Bereitschaftsdatenbank von unterschiedlichen Cluster-Managern verwaltet werden

Diese Konfiguration eignet sich am besten für Umgebungen, in denen sich die Primärdatenbank und die Bereitschaftsdatenbank an verschiedenen Standorten befinden und im Fall eines vollständigen Ausfalls eines Standorts hohe Verfügbarkeit für die Recovery nach einem Katastrophenfall erforderlich ist. Sie können diese Konfiguration auf verschiedene Weise implementieren. Wenn eine Primärdatenbank oder eine Bereitschaftsdatenbank Teil eines Clusters ist, sind zwei Funktionsübernahmeszenarios möglich.

- Bei einem teilweisen Ausfall eines Standorts können Sie eine Clusterfunktionsübernahme ausführen, sofern ein Knoten verfügbar blieb, den das DBMS für die Funktionsübernahme verwenden kann. In diesem Fall wird die Funktionsübernahme für IP-Adresse und Datenträger mit dem Cluster-Manager ausgeführt, und HADR ist nicht betroffen.
- Bei einem vollständigen Ausfall eines Standorts, an dem sich die Primärdatenbank befindet, können Sie mit HADR die DBMS-Verfügbarkeit gewährleisten, indem Sie eine Übernahmeoperation einleiten. Bei einem vollständigen Ausfall eines Standorts, an dem sich die Bereitschaftsdatenbank befindet, können Sie eine Reparatur des Standorts ausführen oder die Bereitschaftsdatenbank an einen anderen Standort versetzen.

Initialisieren einer Bereitschaftsdatenbank

Eine der Strategien für eine Datenbanklösung mit hoher Verfügbarkeit besteht darin, eine primäre Datenbank für die Verarbeitung von Benutzeranforderungen zu betreiben, sowie eine sekundäre bzw. Bereitschaftsdatenbank, die die Datenbankoperationen für die Primärdatenbank übernehmen kann, falls diese ausfällt.

Für das Initialisieren der Bereitschaftsdatenbank ist das Kopieren der Primärdatenbank in die Bereitschaftsdatenbank erforderlich.

Vorgehensweise

Die Bereitschaftsmaschine kann auf verschiedene Weisen initialisiert werden. Beispiel:

- Verwenden Sie die Plattenspiegelung, um die Primärdatenbank zu kopieren, und verwenden Sie die DB2-Datenbankunterstützung für ausgesetzte E/A, um die Spiegeldatenbank zu teilen und damit die zweite Datenbank zu erstellen.
- Erstellen Sie ein Backup-Image der Primärdatenbank und führen Sie für dieses Image eine Recovery in der Bereitschaftsdatenbank aus.
- Verwenden Sie die SQL Replication, um Daten aus der Primärdatenbank zu erfassen und sie in der Bereitschaftsdatenbank anzuwenden.

Nächste Schritte

Nach dem Initialisieren der Bereitschaftsdatenbank müssen Sie Ihre Datenbanklösung so konfigurieren, dass die Primärdatenbank und die Bereitschaftsdatenbank synchronisiert werden, damit die Bereitschaftsdatenbank die Operationen für die Primärdatenbank übernehmen kann, falls diese ausfällt.

Verwenden einer geteilten Spiegeldatenbank als Bereitschaftsdatenbank

Verwenden Sie die folgende Prozedur, um eine geteilte Spiegeldatenbank einer Datenbank zu erstellen, die als Bereitschaftsdatenbank außerhalb einer DB2 pureScale-Umgebung verwendet werden soll.

Wenn in der Primärdatenbank ein Fehler auftritt und nicht mehr auf sie zugegriffen werden kann, können Sie die Bereitschaftsdatenbank die Rolle der Primärdatenbank übernehmen lassen.

Informationen zu diesem Vorgang

Wenn die primäre Datenbank für die Protokollarchivierung konfiguriert wurde, wird dieselbe Protokollarchivierungskonfiguration von der Bereitschaftsdatenbank gemeinsam genutzt. Wenn das Ziel der Protokollarchivierung von der Bereitschaftsdatenbank aus zugänglich ist, ruft die Bereitschaftsdatenbank während Operationen zur aktualisierenden Recovery aus diesem Ziel automatisch Protokolldateien ab. Sobald jedoch die Datenbank den Status 'Aktualisierende Recovery anstehend' verlassen hat, versucht die Bereitschaftsdatenbank Protokolldateien an derselben Position zu archivieren, die auch die Primärdatenbank verwendet hat. Während die Bereitschaftsdatenbank zunächst eine andere Protokollkette als die Primärdatenbank verwendet, könnte die Primärdatenbank mit der Zeit denselben Wert für die Protokollkette verwenden wie die Bereitschaftsdatenbank. Dies könnte dazu führen, dass die Primärdatenbank Protokolldateien über die Protokolldateien archiviert, die von der Bereitschaftsdatenbank archiviert wurden oder umgekehrt, sodass möglicherweise die Wiederherstellbarkeit beider Datenbanken betroffen ist. Zum Vermeiden von Problemen mit der Wiederherstellbarkeit sollten Sie daher das Ziel für die Protokollarchivierung für die Bereitschaftsdatenbank ändern, sodass es sich von dem Ziel der Primärdatenbank unterscheidet.

Vorgehensweise

Gehen Sie wie folgt vor, um eine geteilte Spiegeldatenbank als Bereitschaftsdatenbank zu verwenden:

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zur Primärdatenbank her:

```
db2 connect to datenbankname
```

2. Setzen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl aus:

   ```
   db2 set write suspend for database
   ```

   Anmerkung: Führen Sie keine weiteren Dienstprogramme oder Tools aus, während sich die Datenbank im ausgesetzten Status befindet. Das Erstellen einer Datenbankkopie sollte die einzige Aktivität sein. Sie können optional die Anweisung FLUSH BUFFERPOOLS ALL verwenden, bevor Sie den Befehl **SET WRITE** SUSPEND absetzen, um die Wiederherstellungszeit der Bereitschaftsdatenbank zu minimieren.

3. Erstellen Sie aus der Primärdatenbank mithilfe der entsprechenden Befehle auf Betriebssystem- bzw. Speicherebene mindestens eine geteilte Spiegeldatenbank.

   Anmerkung:

   - Stellen Sie sicher, dass Sie das gesamte Datenbankverzeichnis einschließlich des Datenträgerverzeichnisses kopieren. Sie müssen auch das Protokollverzeichnis und alle Containerverzeichnisse kopieren, die außerhalb des Datenbankverzeichnisses vorhanden sind. Eine Zusammenstellung dieser Informationen finden Sie in der Verwaltungssicht DBPATHS, in der alle Dateien und Verzeichnisse der Datenbank angezeigt werden, die geteilt werden müssen.
   - Wenn Sie EXCLUDE LOGS im Befehl **SET WRITE** angegeben haben, dürfen Sie die Protokolldateien nicht in die Kopie einbeziehen.

4. Nehmen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl wieder auf:

   ```
   db2 set write resume for database
   ```

5. Katalogisieren Sie die gespiegelte Datenbank auf dem sekundären System.

   Anmerkung: Eine gespiegelte Datenbank kann standardmäßig nicht in demselben System vorhanden sein wie die Primärdatenbank. Sie muss sich auf einem sekundären System mit der gleichen Verzeichnisstruktur befinden und denselben Instanznamen als primäre Datenbank verwenden. Wenn die gespiegelte Datenbank auf demselben System wie die Primärdatenbank vorhanden sein muss, können Sie das Dienstprogramm **db2relocatedb** oder die Option RELOCATE USING des Befehls **db2inidb** für diese Aufgabe verwenden.

6. Starten Sie die Datenbankinstanz auf dem sekundären System mit folgendem Befehl:

   ```
   db2start
   ```

7. Initialisieren Sie die gespiegelte Datenbank auf dem sekundären System, indem Sie sie in den Status für anstehende aktualisierende Recovery versetzen. Verwenden Sie dazu folgenden Befehl:

   ```
   db2inidb <aliasname-der-datenbank> as standby
   ```

   Falls erforderlich, geben Sie die Option RELOCATE USING des Befehls **db2inidb** an, um die Bereitschaftsdatenbank zu verlagern:

   ```
   db2inidb <aliasname-der-datenbank> as standby relocate using relocatedbcfg.txt
   ```

   Dabei ist relocatedbcfg.txt die Datei, die die zum Verlagern der Datenbank erforderlichen Informationen enthält.

Anmerkung: Sie können ein vollständiges Datenbankbackup mit geteilter Spiegeldatenbank durchführen, wenn Sie über DMS-Tabellenbereiche oder Tabellenbereiche des dynamischen Speichers verfügen. Das Durchführen eines Backups mit geteilter Spiegeldatenbank verringert den Systemaufwand für ein Backup der Produktionsdatenbank. Solche Backups werden als Online-Backups betrachtet und enthalten unvollständige Transaktionen; Sie können jedoch keine Protokolldateien aus der Bereitschaftsdatenbank einschließen. Wenn ein solches Backup wiederhergestellt wird, müssen Sie eine aktualisierende Recovery mindestens bis zum Ende des Backups durchführen, bevor Sie einen Befehl **ROLLFORWARD** mit der Option STOP absetzen können. Da das Backup keine Protokolldateien enthält, müssen die Protokolldateien aus der Primärdatenbank verfügbar sein, die zum Zeitpunkt, als der Befehl **SET WRITE** SUSPEND abgesetzt wurde, verwendet wurden, oder die aktualisierende Recovery kann das Ende des Backups nicht erreichen.

8. Stellen Sie die archivierten Protokolldateien aus der Primärdatenbank der Bereitschaftsdatenbank zur Verfügung, indem Sie die Parameter für die Protokollarchivierung auf der Bereitschaftsdatenbank konfigurieren oder indem Sie Protokolle an die Bereitschaftsdatenbank senden.
9. Stellen Sie die Datenbank bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt aktualisierend wieder her.
10. Rufen Sie weiter Protokolldateien ab, und stellen Sie die Datenbank anhand der Protokolle aktualisierend wieder her, entweder bis zum Ende der Protokolle oder bis zu dem für die Bereitschaftsdatenbank erforderlichen Zeitpunkt.
11. Versetzen Sie die Bereitschaftsdatenbank in den Online-Status, indem Sie den Befehl **ROLLFORWARD** mit der Option STOP absetzen.

Anmerkung:

- Die Protokolle der Primärdatenbank können nicht mehr auf die gespiegelte Datenbank angewendet werden, nachdem für sie der Status für anstehende aktualisierende Recovery aufgehoben wurde.
- Wenn die primäre Datenbank für die Protokollarchivierung konfiguriert wurde, wird dieselbe Protokollarchivierungskonfiguration von der Bereitschaftsdatenbank gemeinsam genutzt. Wenn das Ziel der Protokollarchivierung von der Bereitschaftsdatenbank aus zugänglich ist, ruft die Bereitschaftsdatenbank während Operationen zur aktualisierenden Recovery aus diesem Ziel automatisch Protokolldateien ab. Sobald jedoch die Datenbank den Status 'Aktualisierende Recovery anstehend' verlassen hat, versucht die Bereitschaftsdatenbank, Protokolldateien an derselben Position zu archivieren, die auch die Primärdatenbank verwendet hat. Obwohl die Bereitschaftsdatenbank zunächst eine andere Protokollkette als die Primärdatenbank verwendet, könnte die Primärdatenbank mit der Zeit denselben Wert für die Protokollkette verwenden wie die Bereitschaftsdatenbank. Dies kann dazu führen, dass die Primärdatenbank Protokolldateien über die Protokolldateien archiviert, die von der Bereitschaftsdatenbank archiviert wurden, oder umgekehrt. Dies kann Auswirkungen auf die Wiederherstellbarkeit beider Datenbanken haben. Zum Vermeiden dieser Probleme sollten Sie das Ziel der Protokollarchivierung für die Bereitschaftsdatenbank ändern, so dass es sich von dem Ziel der Primärdatenbank unterscheidet.

Verwenden einer geteilten Spiegeldatenbank als Bereitschaftsdatenbank in einer DB2 pureScale-Umgebung

Verwenden Sie die folgende Prozedur, um eine geteilte Spiegeldatenbank einer Datenbank zu erstellen, die als Bereitschaftsdatenbank in einer DB2 pureScale-Umgebung verwendet werden soll. Wenn in der Primärdatenbank ein Fehler auftritt und

nicht mehr auf sie zugegriffen werden kann, können Sie die Bereitschaftsdatenbank die Rolle der Primärdatenbank übernehmen lassen.

Informationen zu diesem Vorgang

Wenn die primäre Datenbank für die Protokollarchivierung konfiguriert wurde, wird dieselbe Protokollarchivierungskonfiguration von der Bereitschaftsdatenbank gemeinsam genutzt. Wenn das Ziel der Protokollarchivierung von der Bereitschaftsdatenbank aus zugänglich ist, ruft die Bereitschaftsdatenbank während Operationen zur aktualisierenden Recovery aus diesem Ziel automatisch Protokolldateien ab. Sobald jedoch die Datenbank den Status 'Aktualisierende Recovery anstehend' verlassen hat, versucht die Bereitschaftsdatenbank Protokolldateien an derselben Position zu archivieren, die auch die Primärdatenbank verwendet hat. Während die Bereitschaftsdatenbank zunächst eine andere Protokollkette als die Primärdatenbank verwendet, könnte die Primärdatenbank mit der Zeit denselben Wert für die Protokollkette verwenden wie die Bereitschaftsdatenbank. Dies könnte dazu führen, dass die Primärdatenbank Protokolldateien über die Protokolldateien archiviert, die von der Bereitschaftsdatenbank archiviert wurden oder umgekehrt, sodass möglicherweise die Wiederherstellbarkeit beider Datenbanken betroffen ist. Zum Vermeiden von Problemen mit der Wiederherstellbarkeit sollten Sie daher das Ziel für die Protokollarchivierung für die Bereitschaftsdatenbank ändern, sodass es sich von dem Ziel der Primärdatenbank unterscheidet.

Vorgehensweise

Gehen Sie wie folgt vor, um eine geteilte Spiegeldatenbank als Bereitschaftsdatenbank zu verwenden:

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zur Primärdatenbank her:
   ```
   db2 connect to <datenbankname>
   ```
2. Konfigurieren Sie das General Parallel File System (GPFS) auf dem sekundären Cluster durch Extrahieren und Importieren der Einstellungen des primären Clusters. Führen Sie auf dem primären Cluster den folgenden GPFS-Befehl aus:
   ```
   mmfsctl <dateisystem> syncFSconfig -n <datei-des-fernen-knotens>
   ```

   Dabei steht <*datei-des-fernen-knotens*> für die Liste der Hosts im sekundären Cluster.
3. Listen Sie die Cluster-Manager-Domäne mit folgendem Befehl auf:
   ```
   db2cluster -cm -list -domain
   ```
4. Stoppen Sie den Cluster-Manager auf jedem Host im Cluster mit folgendem Befehl:
   ```
   db2cluster
   -cm -stop -host <host> -force
   ```

   Anmerkung: Sie müssen den Host, von dem aus Sie diesen Befehl absetzen, als Letztes beenden.
5. Stoppen Sie den GPFS-Cluster auf dem sekundären System mit dem folgenden Befehl:
   ```
   db2cluster -cfs -stop -all
   ```
6. Setzen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl aus:
   ```
   db2 set write suspend for database
   ```

Anmerkung: Führen Sie keine weiteren Dienstprogramme oder Tools aus, während sich die Datenbank im ausgesetzten Status befindet. Das Erstellen einer Datenbankkopie sollte die einzige Aktivität sein. Optional können Sie vor Eingabe des Befehls **SET WRITE SUSPEND** alle Pufferpools löschen, um das Recoveryfenster zu minimieren. Hierfür können Sie die Anweisung FLUSH BUFFERPOOLS ALL verwenden.

7. Ermitteln Sie mithilfe des folgenden Befehls, welche Dateisysteme ausgesetzt und kopiert werden müssen:

   ```
   db2cluster -cfs -list -filesystem
   ```

8. Setzen Sie mit folgendem Befehl alle GPFS-Dateisysteme aus, die Daten oder Protokolldaten enthalten:

   ```
   /usr/lpp/mmfs/bin/mmfsctl <dateisystem> suspend
   ```

   Dabei steht <*dateisystem*> für ein Dateisystem, das Daten oder Protokolldaten enthält.

   Anmerkung: Während die GPFS-Dateisysteme ausgesetzt sind, sind sowohl Lese- als auch Schreiboperationen blockiert. Führen Sie in diesem Zeitraum ausschließlich die Operationen für die geteilte Spiegeldatenbank aus, um den Zeitraum zu minimieren, in dem Leseoperationen blockiert sind.

9. Erstellen Sie aus der Primärdatenbank mithilfe der entsprechenden Befehle auf Betriebssystem- bzw. Speicherebene mindestens eine geteilte Spiegeldatenbank.

   Anmerkung:

   - Stellen Sie sicher, dass Sie das gesamte Datenbankverzeichnis einschließlich des Datenträgerverzeichnisses kopieren. Sie müssen auch das Protokollverzeichnis und alle Containerverzeichnisse kopieren, die außerhalb des Datenbankverzeichnisses vorhanden sind. Eine Zusammenstellung dieser Informationen finden Sie in der Verwaltungssicht DBPATHS, in der alle Dateien und Verzeichnisse der Datenbank angezeigt werden, die geteilt werden müssen.
   - Wenn Sie `EXCLUDE LOGS` im Befehl **SET WRITE** angegeben haben, dürfen Sie die Protokolldateien nicht in die Kopie einbeziehen.

10. Setzen Sie den Betrieb der ausgesetzten GPFS-Dateisysteme fort. Verwenden Sie dazu den folgenden Befehl für jedes ausgesetzte Dateisystem:

    ```
    /usr/lpp/mmfs/bin/mmfsctl <dateisystem> resume
    ```

    Dabei steht *dateisystem* für ein ausgesetztes Dateisystem, das Daten oder Protokolldaten enthält.

11. Nehmen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl wieder auf:

    ```
    db2 set write resume for database
    ```

12. Starten Sie den GPFS-Cluster auf dem sekundären System mit dem folgenden Befehl:

    ```
     db2cluster -cfs -start -all
    ```

13. Starten Sie den Cluster-Manager mit folgendem Befehl:

    ```
    db2cluster -cm -start -domain <domäne>
    ```

14. Katalogisieren Sie die gespiegelte Datenbank auf dem sekundären System.

    Anmerkung: Eine gespiegelte Datenbank kann standardmäßig nicht in demselben System vorhanden sein wie die Primärdatenbank. Sie muss sich auf ei-

nem sekundären System mit der gleichen Verzeichnisstruktur befinden und denselben Instanznamen als primäre Datenbank verwenden. Wenn die gespiegelte Datenbank auf demselben System wie die Primärdatenbank vorhanden sein muss, können Sie das Dienstprogramm **db2relocatedb** oder die Option **RELOCATE USING** des Befehls **db2inidb** für diese Aufgabe verwenden.

15. Starten Sie die Datenbankinstanz auf dem sekundären System mit folgendem Befehl:

    ```
    db2start
    ```

16. Initialisieren Sie die Datenbank auf dem sekundären System, indem Sie sie in den Status für anstehende aktualisierende Recovery versetzen:

    ```
    db2inidb <aliasname-der-datenbank> as standby
    ```

    Falls erforderlich, geben Sie die Option RELOCATE USING des Befehls **db2inidb** an, um die Datenbank zu verlagern:

    ```
    db2inidb aliasname-der-datenbank as standby relocate using relocatedbcfg.txt
    ```

    Dabei ist relocatedbcfg.txt die Datei, die die zum Verlagern der Datenbank erforderlichen Informationen erhält.

    Anmerkung: Sie können ein vollständiges Datenbankbackup mit geteilter Spiegeldatenbank durchführen, wenn Sie über DMS-Tabellenbereiche oder Tabellenbereiche des dynamischen Speichers verfügen. Das Durchführen eines Backups mit geteilter Spiegeldatenbank verringert den Systemaufwand für ein Backup der Produktionsdatenbank. Solche Backups werden als Online-Backups betrachtet und enthalten unvollständige Transaktionen; Sie können jedoch keine Protokolldateien aus der Bereitschaftsdatenbank einschließen. Wenn ein solches Backup wiederhergestellt wird, müssen Sie eine aktualisierende Recovery mindestens bis zum Ende des Backups durchführen, bevor Sie einen Befehl **ROLLFORWARD** STOP absetzen können. Da das Backup keine Protokolldateien enthält, müssen die Protokolldateien aus der Primärdatenbank verfügbar sein, die zum Zeitpunkt, als der Befehl **SET WRITE** SUSPEND abgesetzt wurde, verwendet wurden, oder die aktualisierende Recovery kann das Ende des Backups nicht erreichen.

17. Stellen Sie die archivierten Protokolldateien aus der Primärdatenbank der Bereitschaftsdatenbank zur Verfügung, indem Sie die Parameter für die Protokollarchivierung auf der Bereitschaftsdatenbank konfigurieren oder indem Sie Protokolle an die Bereitschaftsdatenbank senden.

18. Stellen Sie die Datenbank bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt aktualisierend wieder her.

    Anmerkung: Bei der Durchführung von aktualisierenden Recoverys treten möglicherweise Fehler auf (SQL1273). Diese Fehler sind erwartet, sofern einige der Protokolldateien zum Zeitpunkt der Teilung der Datenbank nicht aus dem Primärsystem kopiert wurden oder sofern ein Member Protokolldateien schneller als die anderen Member generiert hat. Der Fehler SQL1273 wird in einigen Fällen zurückgegeben, wenn die aktualisierende Recovery stoppen muss, um die Datenkonsistenz zu bewahren, da der Inhalt der Protokolldateien auf dem Inhalt von nicht verfügbaren Protokolldateien anderer Member beruht. Wenn die Bereitschaftsdatenbank so konfiguriert ist, dass sie von der Primärdatenbank archivierte Protokolldateien abruft, können Sie abwarten, bis das primäre System die erforderliche Protokolldatei archiviert oder Sie können den Befehl **ARCHIVE LOG** auf dem primären System absetzen, um die Archivierung der Protokolldatei zu erzwingen. Andernfalls müssen Sie die erforderlichen Protokolldateien an die Bereitschaftsdatenbank senden. Nachdem die erforderliche Protokolldatei auf der Bereitschaftsdatenbank verfügbar ist, kann

die aktualisierende Recovery mit dem Lesen der Protokolle fortfahren, obwohl ein weiteres Mal der Fehler SQL1273 auftreten kann, falls immer noch einige Member Protokolldateien schneller als andere Member erstellen. Weitere Informationen finden Sie im Abschnitt „Recovery nach einem Katastrophenfall und Hochverfügbarkeit durch Protokollübertragung in einer DB2 pureScale-Umgebung"des Themas „Backup- und Restoreoperationen in einer DB2 pureScale-Umgebung" im Information Center.

19. Setzen Sie die aktualisierende Recovery für die Protokolle fort, bis Sie das Ende der Protokolle oder den für die Bereitschaftsdatenbank erforderlichen Zeitpunkt erreichen (falls erforderlich).
20. Versetzen Sie die Bereitschaftsdatenbank in den Online-Status, indem Sie den Befehl **ROLLFORWARD DATABASE** mit der Option STOP absetzen.

Anmerkung:

- Die Protokolle der primären Datenbank können nicht mehr auf die gespiegelte Datenbank angewandt werden, nachdem für sie der Status für anstehende aktualisierende Recovery aufgehoben wurde.
- Wenn die Primärdatenbank für die Protokollarchivierung konfiguriert wurde, wird dieselbe Protokollarchivierungskonfiguration von der Bereitschaftsdatenbank gemeinsam genutzt. Wenn das Ziel der Protokollarchivierung von der Bereitschaftsdatenbank aus zugänglich ist, ruft die Bereitschaftsdatenbank während Operationen zur aktualisierenden Recovery aus diesem Ziel automatisch Protokolldateien ab. Sobald jedoch die Datenbank den Status 'Aktualisierende Recovery anstehend' verlassen hat, versucht die Bereitschaftsdatenbank, Protokolldateien an derselben Position zu archivieren, die auch die Primärdatenbank verwendet hat. Obwohl die Bereitschaftsdatenbank zunächst eine andere Protokollkette als die Primärdatenbank verwendet, könnte die Primärdatenbank mit der Zeit denselben Wert für die Protokollkette verwenden wie die Bereitschaftsdatenbank. Dies kann dazu führen, dass die Primärdatenbank Protokolldateien über die Protokolldateien archiviert, die von der Bereitschaftsdatenbank archiviert wurden, oder umgekehrt. Dies kann Auswirkungen auf die Wiederherstellbarkeit beider Datenbanken haben. Zum Vermeiden dieser Probleme sollten Sie das Ziel der Protokollarchivierung für die Bereitschaftsdatenbank ändern, so dass es sich von dem Ziel der Primärdatenbank unterscheidet.

HADR-Synchronisationsmodus (High Availability Disaster Recovery)

Der HADR-Synchronisationsmodus legt den Schutzgrad fest, den Ihre DB2-HADR-Datenbanklösung vor Transaktionsverlusten haben soll. Der Synchronisationsmodus legt fest, wann der primäre Datenbankserver eine Transaktion als abgeschlossen betrachtet. Dies basiert auf dem Status der Protokollierung auf der Bereitschaftsdatenbank.

Je strikter der Wert des Konfigurationsparameters für den Synchronisationsmodus, desto mehr Schutz weist Ihre Datenbanklösung vor Verlusten von Transaktionsdaten auf, aber desto langsamer wird auch die Transaktionsverarbeitungsleistung. Sie müssen also die Schutzerfordernisse im Hinblick auf Transaktionsverluste gegen die Leistungserfordernisse abwägen.

In Abb. 3 auf Seite 68 sehen Sie die verfügbaren DB2-HADR-Synchronisationsmodi; außerdem wird dargestellt, wann Transaktionen auf der Grundlage des ausgewählten Synchronisationsmodus als 'festgeschrieben' betrachtet werden:

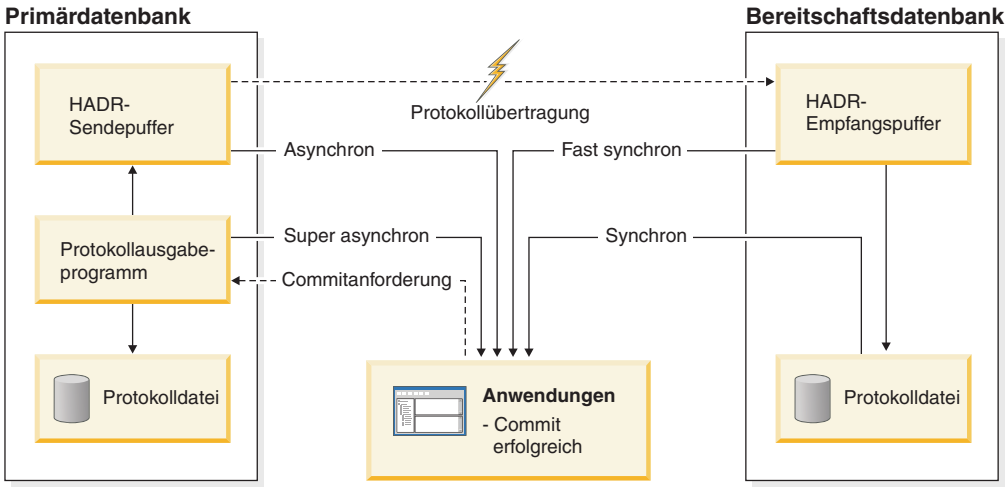


Abbildung 3. Synchronisationsmodi für HADR (High Availability and Disaster Recovery)

Im Modus für mehrere Bereitschaftsdatenbanken muss die Einstellung für **hadr_syncmode** in der Primärdatenbank nicht unbedingt mit der entsprechenden Einstellung der Bereitschaftsdatenbanken übereinstimmen. Die Einstellung für **hadr_syncmode**, die auf einer Bereitschaftsdatenbank angegeben ist, wird als ihr *konfigurierter Synchronisationsmodus* betrachtet. Diese Einstellung ist ausschließlich dann relevant, wenn die Bereitschaftsdatenbank die Rolle einer Primärdatenbank übernimmt. Stattdessen wird der Bereitschaftsdatenbank ein *effektiver Synchronisationsmodus* zugeordnet. Für alle Nebenbereitschaftsdatenbanken lautet der effektive Synchronisationsmodus immer SUPERASYNC. Für die Hauptbereitschaftsdatenbank entspricht der effektive Synchronisationsmodus der Einstellung für **hadr_syncmode** der Primärdatenbank. Der effektive Synchronisationsmodus der Bereitschaftsdatenbank ist der Wert, der von jeder Überwachungsschnittstelle angezeigt wird.

Verwenden Sie den Datenbankkonfigurationsparameter **hadr_syncmode**, um den Synchronisationsmodus festzulegen. Die folgenden Werte sind gültig:

SYNC (synchron)

Dieser Modus bietet den größten Schutz vor Verlust von Transaktionsdaten, benötigt jedoch von allen vier Modi die längste Transaktionsantwortzeit.

In diesem Modus wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokolle in Protokolldateien in der Primärdatenbank geschrieben wurden und wenn die Primärdatenbank von der Bereitschaftsdatenbank eine Empfangsbestätigung erhalten hat, dass die Protokolle auch in Protokolldateien in der Bereitschaftsdatenbank geschrieben wurden. So wird sichergestellt, dass die Protokolldaten an beiden Standorten gespeichert werden.

Wenn die Bereitschaftsdatenbank ausfällt, bevor sie die Protokollsätze nachvollziehen kann, kann sie bei ihrem nächsten Start die Protokollsätze aus ihren lokalen Protokolldateien abrufen und nachvollziehen. Wenn die Primärdatenbank ausfällt, wird durch eine Funktionsübernahme durch die Bereitschaftsdatenbank sichergestellt, dass alle in der Primärdatenbank festgeschriebenen Transaktionen auch in der Bereitschaftsdatenbank festgeschrieben wurden. Wenn der Client nach der Funktionsübernahmeoperation eine Verbindung zur neuen Primärdatenbank herstellt, können in der neuen Primärdatenbank Transaktionen festgeschrieben sein, die an die An-

wendung auf der ursprünglichen Primärdatenbank nie als festgeschrieben gemeldet wurden. Dies geschieht, wenn die Primärdatenbank ausfällt, bevor sie eine Empfangsbestätigungsnachricht von der Bereitschaftsdatenbank verarbeiten konnte. Clientanwendungen sollten unter Umständen die Datenbank abfragen, um zu ermitteln, ob solche Transaktionen vorhanden sind.

Wenn die Primärdatenbank keine Verbindung mehr zur Bereitschaftsdatenbank hat, ist die weitere Vorgehensweise von der Konfiguration des Datenbankkonfigurationsparameters **hadr_peer_window** abhängig. Wenn **hadr_peer_window** auf einen Zeitwert ungleich null gesetzt ist, wechselt die Primärdatenbank nach dem Verlust der Verbindung mit der Bereitschaftsdatenbank in den Status 'Unterbrochener Peer' und wartet weiterhin auf eine Bestätigung von der Bereitschaftsdatenbank, bevor Transaktionen festgeschrieben werden. Wenn der Datenbankkonfigurationsparameter **hadr_peer_window** auf null gesetzt ist, wird nicht länger davon ausgegangen, dass sich die Primär- und die Bereitschaftsdatenbank im Peerstatus befinden. Transaktionen werden nicht länger zurückgehalten, um auf eine Bestätigung von der Bereitschaftsdatenbank zu warten. Wird die Funktionsübernahmeoperation ausgeführt, obwohl sich die Datenbanken nicht im Peerstatus oder im Status 'Unterbrochener Peer' befinden, gibt es keine Gewährleistung, dass alle in der Primärdatenbank festgeschriebenen Transaktionen auch in der Bereitschaftsdatenbank vorhanden sind.

Fällt die Primärdatenbank aus, während sich die Datenbanken im Peerstatus bzw. im Status 'Unterbrochener Peer' befinden, kann sie nach einer Funktionsübernahmeoperation dem HADR-Paar als Bereitschaftsdatenbank hinzugefügt werden. Da eine Transaktion erst dann als festgeschrieben betrachtet wird, wenn die Primärdatenbank eine Empfangsbestätigung von der Bereitschaftsdatenbank erhält, dass auch die Protokolle in Protokolldateien in der Bereitschaftsdatenbank geschrieben wurden, stimmt die Protokollfolge der Primärdatenbank mit der Protokollfolge der Bereitschaftsdatenbank überein. Die ursprüngliche Primärdatenbank (die nun als Bereitschaftsdatenbank fungiert) muss lediglich die neuen Protokollsätze nachvollziehen, die seit der Funktionsübernahmeoperation in der neuen Primärdatenbank generiert wurden.

Wenn sich die Primärdatenbank zum Zeitpunkt ihres Ausfalls nicht im Peerstatus befand, können Unterschiede zwischen ihrer Protokollfolge und der Protokollfolge der Bereitschaftsdatenbank bestehen. Wird eine Funktionsübernahmeoperation erforderlich, können diese Unterschiede dadurch entstehen, dass die Bereitschaftsdatenbank nach der Funktionsübernahme eine eigene Protokollfolge startet. Da einige Operationen nicht rückgängig gemacht werden können (z. B. das Löschen einer Tabelle), ist es nicht möglich, die Primärdatenbank auf den Zeitpunkt zurückzusetzen, zu dem die neue Protokollfolge erstellt wurde. Wenn die Protokollfolgen nicht übereinstimmen und Sie den Befehl **START HADR** mit dem Parameter **AS STANDBY** in der ursprünglichen Primärdatenbank ausführen, empfangen Sie eine Nachricht, dass der Befehl erfolgreich war. Allerdings wird diese Nachricht ausgegeben, bevor die Reintegration versucht wird. Wenn die Reintegration fehlschlägt, werden Nachrichten über die Prüfung des Paares an das Verwaltungsprotokoll und das Diagnoseprotokoll in der Primärdatenbank und der Bereitschaftsdaten ausgegeben. Die reintegrierte Bereitschaftsdatenbank bleibt Bereitschaftsdatenbank, jedoch weist die Primärdatenbank die Bereitschaftsdatenbank bei der Prüfung des Paares zurück, sodass die Bereitschaftsdatenbank herunterfährt. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar erneut erfolgreich hinzugefügt wurde, können Sie die Da-

tenbank zurücksetzen, indem Sie den Befehl **TAKEOVER HADR** ohne den Parameter **BY FORCE** absetzen. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar nicht wieder hinzugefügt werden kann, können Sie sie reinitialisieren, indem Sie ein Backup-Image der neuen Primärdatenbank wiederherstellen.

NEARSYNC (fast synchron)
Dieser Modus hat eine kürzere Transaktionsantwortzeit als der Modus SYNC, er bietet aber auch einen etwas geringeren Schutz vor Transaktionsverlust.

In diesem Modus wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokolle in Protokolldateien in der Primärdatenbank geschrieben wurden und wenn die Primärdatenbank von der Bereitschaftsdatenbank eine Empfangsbestätigung erhalten hat, dass die Protokolle auch in den Hauptspeicher des Bereitschaftssystems geschrieben wurden. Zu Datenverlust kann es nur dann kommen, wenn beide Standorte gleichzeitig ausfallen und wenn der Zielstandort nicht alle empfangenen Protokolldaten an nicht flüchtigen Speicher übertragen hat.

Wenn die Bereitschaftsdatenbank ausfällt, bevor sie die Protokollsätze aus dem Speicher auf Platte kopieren kann, gehen diese Protokollsätze in der Bereitschaftsdatenbank verloren. Normalerweise kann die Bereitschaftsdatenbank bei ihrem nächsten Start die fehlenden Protokollsätze aus der Primärdatenbank abrufen. Wenn jedoch in der Primärdatenbank ein Fehler auftritt oder das Netz das Abrufen der Protokollsätze unmöglich macht, werden die Protokollsätze ebenso wie die ihnen zugeordneten Transaktionen nie in der Bereitschaftsdatenbank angezeigt.

Gehen Transaktionen verloren, ist die neue Primärdatenbank nach einer Funktionsübernahme nicht mit der ursprünglichen Primärdatenbank identisch. Clientanwendungen sollten diese Transaktionen gegebenenfalls erneut übergeben, um den Anwendungsstatus zu aktualisieren.

Fällt die Primärdatenbank aus, während sich die Primärdatenbank und die Bereitschaftsdatenbank im Peerstatus befinden, ist es möglich, dass die ursprüngliche Primärdatenbank dem HADR-Paar nach einer Funktionsübernahmeoperation nur dann als Bereitschaftsdatenbank hinzugefügt werden kann, wenn sie mit einer Operation zum vollständigen Restore reinitialisiert wurde. Wenn die Funktionsübernahme auch verlorene Protokollsätze umfasst (bei einem Ausfall beider Datenbanken), stimmen die Protokollfolge der Primärdatenbank und der Bereitschaftsdatenbank nicht überein, sodass Versuche, die ursprüngliche Primärdatenbank als Bereitschaftsdatenbank erneut zu starten, fehlschlagen, wenn nicht zuerst eine Restoreoperation durchgeführt wird. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar erneut erfolgreich hinzugefügt wurde, können Sie die Datenbank zurücksetzen, indem Sie den Befehl **TAKEOVER HADR** ohne den Parameter **BY FORCE** absetzen. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar nicht wieder hinzugefügt werden kann, können Sie sie reinitialisieren, indem Sie ein Backup-Image der neuen Primärdatenbank wiederherstellen.

ASYNC (asynchron)
Im Vergleich zu den Modi SYNC und NEARSYNC hat der Modus ASYNC zwar kürzere Transaktionsantwortzeiten, kann aber größere Transaktionsverluste zur Folge haben, wenn die Primärdatenbank ausfällt.

Im Modus ASYNC wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokollsätze in die Protokolldateien in der Pri-

märdatenbank geschrieben und an die TCP-Schicht der Hostmaschine des primären Systems übergeben wurden. Da das primäre System nicht auf eine Empfangsbestätigung vom Bereitschaftssystem wartet, können Transaktionen bereits als festgeschrieben betrachtet werden, während sie noch an die Bereitschaftsdatenbank weitergeleitet werden.

Bei einem Ausfall der Hostmaschine der Primärdatenbank, des Netzes oder der Bereitschaftsdatenbank können Protokollsätze, die zu diesem Zeitpunkt gerade weitergeleitet werden, verloren gehen. Steht die Primärdatenbank zur Verfügung, können die fehlenden Protokollsätze erneut an die Bereitschaftsdatenbank übertragen werden, wenn die Verbindung zwischen den beiden Datenbanken wiederhergestellt wird. Wird jedoch eine Funktionsübernahme erforderlich, während Protokollsätze fehlen, erreichen diese Protokollsätze nie die Bereitschaftsdatenbank, sodass die entsprechenden Transaktionen bei der Funktionsübernahme verloren gehen.

Gehen Transaktionen verloren, ist die neue Primärdatenbank nach einer Funktionsübernahme nicht mit der ursprünglichen Primärdatenbank identisch. Clientanwendungen sollten diese Transaktionen gegebenenfalls erneut übergeben, um den Anwendungsstatus zu aktualisieren.

Fällt die Primärdatenbank aus, während sich die Primärdatenbank und die Bereitschaftsdatenbank im Peerstatus befinden, ist es möglich, dass die ursprüngliche Primärdatenbank dem HADR-Paar nach einer Funktionsübernahmeoperation nur dann als Bereitschaftsdatenbank hinzugefügt werden kann, wenn sie einer vollständigen Restoreoperation reinitialisiert wurde. Wenn die Funktionsübernahme auch verlorene Protokollsätze umfasst, stimmen die Protokollfolge der Primärdatenbank und der Bereitschaftsdatenbank nicht überein, sodass Versuche, die ursprüngliche Primärdatenbank als Bereitschaftsdatenbank erneut zu starten, fehlschlagen, wenn nicht zuerst eine Restoreoperation ausgeführt wird. Da beim Auftreten einer Funktionsübernahme in asynchronem Modus ein Verlust von Protokollsätzen wahrscheinlicher ist, ist auch die Wahrscheinlichkeit größer, dass die Primärdatenbank dem HADR-Paar nicht wieder hinzugefügt werden kann. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar erneut erfolgreich hinzugefügt wurde, können Sie die Datenbank zurücksetzen, indem Sie den Befehl **TAKEOVER HADR** ohne den Parameter **BY FORCE** absetzen. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar nicht wieder hinzugefügt werden kann, können Sie sie reinitialisieren, indem Sie ein Backup-Image der neuen Primärdatenbank wiederherstellen.

SUPERASYNC (super asynchron)

Dieser Modus hat die kürzeste Transaktionsantwortzeit; bei diesem Modus sind aber auch Transaktionsverluste am Wahrscheinlichsten, wenn das primäre System ausfällt. Dieser Modus ist nützlich, wenn Sie möchten, dass Transaktionen niemals geblockt werden; es kommt auch niemals zu ausgedehnten Antwortzeiten aufgrund von Netzunterbrechungen oder Überlastung.

In diesem Modus kann das HADR-Paar niemals den Status 'Peer' oder 'Unterbrochener Peer' aufweisen. Das Schreiben von Protokollen wird als erfolgreich betrachtet, sobald die Protokollsätze in Protokolldateien in der Primärdatenbank geschrieben wurden. Da die Primärdatenbank nicht auf eine Bestätigung der Bereitschaftsdatenbank wartet, werden Transaktionen als 'festgeschrieben' betrachtet, ganz gleich, welchen Replikationsstatus diese Transaktion aufweist.

Bei einem Ausfall der Hostmaschine der Primärdatenbank, des Netzes oder der Bereitschaftsdatenbank können Protokollsätze, die zu diesem Zeitpunkt

gerade weitergeleitet werden, verloren gehen. Steht die Primärdatenbank zur Verfügung, können die fehlenden Protokollsätze erneut an die Bereitschaftsdatenbank übertragen werden, wenn die Verbindung zwischen den beiden Datenbanken wiederhergestellt wird. Wird jedoch eine Funktionsübernahme erforderlich, während Protokollsätze fehlen, erreichen diese Protokollsätze nie die Bereitschaftsdatenbank, sodass die entsprechenden Transaktionen bei der Funktionsübernahme verloren gehen.

Gehen Transaktionen verloren, ist die neue Primärdatenbank nach einer Funktionsübernahme nicht mit der ursprünglichen Primärdatenbank identisch. Clientanwendungen sollten diese Transaktionen gegebenenfalls erneut übergeben, um den Anwendungsstatus zu aktualisieren.

Es vergrößert sich die Protokollabstimmungsdiskrepanz zwischen der Primär- und der Bereitschaftsdatenbank möglicherweise stetig, da die Commitoperationen für Transaktionen in der Primärdatenbank nicht von der relativen Langsamkeit des HADR-Netz- oder des HADR-Bereitschaftsservers betroffen sind. Es ist wichtig, die Protokollabstimmungsdiskrepanz zu überwachen, da es sich um eine indirekte Kennzahl der potenziellen Anzahl von Transaktionen handelt, die möglicherweise im Fall eines echten Störfalls auf dem Primärsystem verloren gehen könnten. Bei Szenarios zur Recovery nach einem Katastrophenfall würden sämtliche Transaktionen, die bei der Protokollabstimmungsdiskrepanz festgeschrieben wurden, der Bereitschaftsdatenbank nicht zur Verfügung stehen. Überwachen Sie die Protokollabstimmungsdiskrepanz daher mit dem Monitorelement **hadr_log_gap**; falls die Protokollabstimmungsdiskrepanz nicht zulässig ist, überprüfen Sie die Netzunterbrechungen oder die jeweilige Geschwindigkeit des Bereitschaftsdatenbankknotens und ergreifen Sie zur Reduzierung der Protokollabstimmungsdiskrepanz korrigierende Maßnahmen.

Fällt die Primärdatenbank aus, ist es möglich, dass die ursprüngliche Primärdatenbank dem HADR-Paar nur dann wieder als Bereitschaftsdatenbank hinzugefügt werden kann, wenn sie mit einer vollständigen Restoreoperation reinitialisiert wurde. Wenn die Funktionsübernahme auch verlorene Protokollsätze umfasst, stimmen die Protokollfolge der Primärdatenbank und der Bereitschaftsdatenbank nicht überein, sodass Versuche, die ursprüngliche Primärdatenbank als Bereitschaftsdatenbank erneut zu starten, fehlschlagen, wenn nicht zuerst eine Restoreoperation ausgeführt wird. Da beim Auftreten einer Funktionsübernahme im Modus SUPERASYNC (super asynchron) ein Verlust von Protokollsätzen wahrscheinlicher ist, ist auch die Wahrscheinlichkeit größer, dass die Primärdatenbank dem HADR-Paar nicht wieder hinzugefügt werden kann. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar erneut erfolgreich hinzugefügt wurde, können Sie die Datenbank zurücksetzen, indem Sie den Befehl **TAKEOVER HADR** ohne den Parameter **BY FORCE** absetzen. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar nicht wieder hinzugefügt werden kann, können Sie sie reinitialisieren, indem Sie ein Backup-Image der neuen Primärdatenbank wiederherstellen.

Unterstützung für High Availability Disaster Recovery (HADR)

Um eine optimale Leistung mit der DB2 High Availability Disaster Recovery (HADR) zu erreichen, müssen Sie Systemanforderungen und Funktionseinschränkungen berücksichtigen, wenn Sie Ihre Datenbanklösung mit hoher Verfügbarkeit entwerfen.

Systemvoraussetzungen für High Availability Disaster Recovery (HADR)

Um eine optimale Leistung der High Availability Disaster Recovery (HADR) zu erreichen, sollten Sie sicherstellen, dass Ihr System die folgenden Voraussetzungen im Hinblick auf die Hardware, Betriebssysteme und das DB2-Datenbanksystem erfüllt.

Empfehlung: Verwenden Sie aus Leistungsgründen für die Systeme, auf denen sich die Primärdatenbank und die Bereitschaftsdatenbank befinden, die gleiche Hardware und Software. Wenn das System, auf dem sich die Bereitschaftsdatenbank befindet, über weniger Ressourcen verfügt, als das System, auf dem sich die Primärdatenbank befindet, ist es möglich, dass die Bereitschaftsdatenbank die von der Primärdatenbank generierte Transaktionslast nicht bewältigen kann. Dies kann dazu führen, dass die Bereitschaftsdatenbank nicht auf dem aktuellsten Stand bleibt bzw. dass die Leistung der Primärdatenbank verringert wird. Im Fall einer Funktionsübernahme sollte die neue Primärdatenbank über die erforderlichen Ressourcen verfügen, um die Clientanwendungen adäquat bedienen zu können.

Wenn Sie die Funktion für Leseoperationen in der Bereitschaftsdatenbank aktivieren und die Bereitschaftsdatenbank für einen Teil der schreibgeschützten Workloads verwenden, müssen Sie sicherstellen, dass die Bereitschaftsdatenbank über ausreichende Ressourcen verfügt. Eine aktive Bereitschaftsdatenbank erfordert zusätzlichen Speicher und Tabellenbereiche für temporäre Tabellen, um Transaktionen, Sitzungen und neue Threads sowie Abfragen zu unterstützen, die Sortier- und Joinoperationen beinhalten.

Hardware- und Betriebssystemvoraussetzungen

Empfehlung: Verwenden Sie für die Primärdatenbank und die Bereitschaftsdatenbank identische Hostcomputer. Das heißt, die Computer sollten denselben Hersteller und dieselbe Architektur haben.

Das Betriebssystem für die Primärdatenbank und die Bereitschaftsdatenbank sollte dieselbe Version einschließlich der Programmkorrekturen haben. Diese Vorgabe können Sie während eines schrittweisen Upgrades vorübergehend außer Acht lassen, Sie sollten dabei jedoch äußerst vorsichtig vorgehen.

Zwischen den HADR-Hostmaschinen muss eine TCP/IP-Schnittstelle zur Verfügung stehen, und es wird ein Hochgeschwindigkeitsnetz mit hoher Kapazität empfohlen.

Voraussetzungen für DB2-Datenbanken

Die Datenbanksystemversionen für die Primär- und die Bereitschaftsdatenbank müssen identisch sein; es muss sich beispielsweise für beide entweder um Version 8 oder um Version 9 handeln. Bei schrittweisen Updates kann die Modifikationsstufe (z. B. die Fixpackstufe) des Datenbanksystems für die Bereitschaftsdatenbank kurzzeitig zum Testen der neuen Stufe neuer als die der Primärdatenbank sein. Allerdings sollte diese Konfiguration nicht über einen längeren Zeitraum verwendet werden. Die Primär- und die Bereitschaftsdatenbank stellen keine Verbindung zueinander her, wenn die Modifikationsstufe des Datenbanksystems für eine Primärdatenbank neuer als die für die Bereitschaftsdatenbank ist. Damit die Funktion für Leseoperationen in der Bereitschaftsdatenbank genutzt werden kann, muss für Primär- und Bereitschaftsdatenbank Version 9.7 Fixpack 1 installiert sein.

Die DB2-Datenbanksoftware muss für die Primärdatenbank und die Bereitschaftsdatenbank über dieselbe Bitgröße verfügen (32 oder 64 Bit). Tabellenbereiche und ihre Container müssen in der Primär- und der Bereitschaftsdatenbank identisch sein. Andere Eigenschaften, die ebenfalls identisch sein müssen, sind der Tabellenbereichstyp (DMS oder SMS), die Tabellenbereichsgröße, der Containerpfad und der Dateityp des Containers (Roheinheit oder Dateisystem). Der den Protokolldateien zugewiesene Speicherbereich sollte außerdem in der Primär- und der Bereitschaftsdatenbank gleich groß sein.

Wenn Sie für die Primärdatenbank eine Tabellenbereichsanweisung ausgeben, wie z. B. CREATE TABLESPACE, ALTER TABLESPACE oder DROP TABLESPACE, wird sie auch auf die Bereitschaftsdatenbank angewendet. Sie müssen sicherstellen, dass die einbezogenen Einheiten in beiden Datenbanken definiert sind, bevor Sie die Tabellenbereichsanweisung in der Primärdatenbank absetzen.

Die Primär- und die Bereitschaftsdatenbank müssen jedoch nicht denselben Datenbankpfad aufweisen. Werden relative Containerpfade verwendet, wird derselbe relative Pfad möglicherweise unterschiedlichen absoluten Containerpfaden in der Primär- und der Bereitschaftsdatenbank zugeordnet.

Speichergruppen werden von HADR vollständig unterstützt, einschließlich Replikation der Anweisungen CREATE STOGROUP, ALTER STOGROUP und DROP STOGROUP. Ähnlich wie bei Tabellenbereichscontainern müssen die Speicherpfade sowohl in der Primärdatenbank als auch in der Bereitschaftsdatenbank vorhanden sein.

Die Primär- und die Bereitschaftsdatenbank müssen denselben Datenbanknamen besitzen. Dies bedeutet, dass sie sich in unterschiedlichen Instanzen befinden müssen.

Ein umgeleiteter Restore wird nicht unterstützt. Das heißt, HADR unterstützt keine Umleitung von Tabellenbereichscontainern. Änderungen am Datenbankverzeichnis und am Protokollverzeichnis werden jedoch unterstützt. Tabellenbereichscontainer, die in relativen Pfaden erstellt wurden, lassen sich in Pfaden relativ zum neuen Datenbankverzeichnis wiederherstellen.

Pufferpoolvoraussetzungen

Da Pufferpooloperationen auch auf die Bereitschaftsdatenbank angewendet werden, ist es wichtig, dass die Primärdatenbank und die Bereitschaftsdatenbank dieselbe Speicherkapazität haben. Wenn Sie mit der Funktion für Leseoperationen in der Bereitschaftsdatenbank arbeiten, müssen Sie den Pufferpool in der Primärdatenbank so konfigurieren, dass die aktive Bereitschaftsdatenbank das Anwenden von Protokollen (Log Replay) und die Ausführung von Anwendungen, die Lesezugriff erfordern.

Installations- und Speichervoraussetzungen für HADR (High Availability Disaster Recovery)

Um eine optimale Leistung mit der High Availability Disaster Recovery (HADR) zu erreichen, sollten Sie sicherstellen, dass Ihr System die folgenden Voraussetzungen im Hinblick auf die Installation und das Speichern erfüllt.

Installationsvoraussetzungen

Für HADR sollten die Instanzpfade in der Primär- und der Bereitschaftsdatenbank übereinstimmen. Eine Verwendung unterschiedlicher Instanzpfade kann in eini-

gen Fällen Probleme verursachen, zum Beispiel wenn eine gespeicherte SQL-Prozedur eine benutzerdefinierte Funktion (UDF) aufruft und ein identischer Pfad zum UDF-Objektcode auf beiden Servern, das heißt dem primären Server und dem Bereitschaftsserver erwartet wird.

Speicheranforderungen

Speichergruppen werden von HADR vollständig unterstützt, einschließlich Replikation der Anweisungen CREATE STOGROUP, ALTER STOGROUP und DROP STOGROUP. Ähnlich wie bei Tabellenbereichscontainern muss der Speicherpfad sowohl in der Primärdatenbank als auch in der Bereitschaftsdatenbank vorhanden sein. Symbolische Links können zur Erstellung identischer Pfade verwendet werden. Die Primärdatenbank und die Bereitschaftsdatenbank können sich auf demselben Computer befinden. Obwohl ihre Datenbankspeicher mit demselben Pfad beginnen, geraten sie nicht in Konflikt, weil die tatsächlich verwendeten Verzeichnisse die Instanznamen in ihren Pfaden enthalten (da die Primär- und die Bereitschaftsdatenbank ja den gleichen Datenbanknamen besitzen müssen, müssen sie sich in verschiedenen Instanzen befinden). Der Speicherpfad wird folgendermaßen formuliert: `speicherpfadname/instanzname/dbpartitionsname/dbname/tabbereichsname/containername`.

Tabellenbereiche und ihre Container müssen in der Primär- und der Bereitschaftsdatenbank identisch sein. Andere Eigenschaften, die ebenfalls identisch sein müssen, sind der Tabellenbereichstyp (DMS oder SMS), die Tabellenbereichsgröße, der Containerpfad und der Dateityp des Containers (Roheinheit oder Dateisystem). Speichergruppen und ihre Speicherpfade müssen identisch sein. Die schließt die Pfadnamen sowie die Menge des jeweils verfügbaren Speicherplatzes ein, der für jede Speichergruppe reserviert ist. Der den Protokolldateien zugewiesene Speicherbereich sollte außerdem in der Primär- und der Bereitschaftsdatenbank gleich groß sein.

Wenn Sie für die Primärdatenbank eine Tabellenbereichsanweisung ausgeben, wie z. B. CREATE TABLESPACE, ALTER TABLESPACE oder DROP TABLESPACE, wird sie auch auf die Bereitschaftsdatenbank angewendet. Stellen Sie sicher, dass die hierfür benötigten Einheiten in beiden Datenbanken eingerichtet sind, bevor Sie die Tabellenbereichsanweisung in der Primärdatenbank eingeben.

Wenn die Tabellenbereichskonfiguration in der Primär- und der Bereitschaftsdatenbank nicht identisch ist, können bei der Anwendung der Protokolle in der Bereitschaftsdatenbank Fehler wie KEIN SPEICHERPLATZ VERFÜGBAR oder TABELLENBEREICHSCONTAINER NICHT GEFUNDEN auftreten. Außerdem werden zu CREATE STOGROUP oder ALTER STOGROUP gehörige Protokollsätze nicht wiederholt, wenn die Speicherpfadkonfiguration der Speichergruppen auf der Primärdatenbank und der Bereitschaftsdatenbank nicht identisch ist. Dies könnte dazu führen, dass für die vorhandenen Speicherpfade bereits frühzeitig kein Speicherplatz mehr auf dem Bereitschaftssystem verfügbar ist und Tabellenbereiche mit dynamischem Speicher nicht vergrößert werden können. Tritt eine dieser Situationen auf, wird der betreffende Tabellenbereich in den Status *aktualisierende Recovery anstehend* gesetzt und bei der folgenden Anwendung der Protokolle ignoriert. Wenn eine Übernahmeoperation stattfindet, steht dieser Tabellenbereich den Anwendungen nicht zur Verfügung.

Wird dieses Problem auf dem Bereitschaftssystem vor einer Übernahme festgestellt, besteht die Lösung darin, die Bereitschaftsdatenbank erneut zu erstellen und dabei die Speicherprobleme zu beheben. Folgende Schritte sind erforderlich:

- Inaktivieren der Bereitschaftsdatenbank.

- Löschen der Bereitschaftsdatenbank.
- Sicherstellen, dass die erforderlichen Dateisysteme vorhanden sind und ausreichend freien Speicherplatz enthalten, damit die folgenden Restore- und aktualisierenden Recoveryoperationen ausgeführt werden können.
- Restore der Datenbank auf dem Bereitschaftssystem mithilfe eines aktuellen Backups der Primärdatenbank (oder Reinitialisieren mithilfe einer geteilten Spiegeldatenbank oder Flash-Kopie und dem Befehl **db2inidb**). Speicherpfade von Speichergruppen dürfen während des Restores nicht neu definiert werden. Darüber hinaus dürfen Tabellenbereichscontainer beim Restore nicht umgeleitet werden.
- Erneutes Starten von HADR auf dem Bereitschaftssystem.

Wenn das Problem in der Bereitschaftsdatenbank jedoch erst nach der Übernahme festgestellt wird (oder wenn entschieden wurde, die Speicherprobleme bis zu diesem Zeitpunkt nicht zu beheben), ist die Lösung abhängig von der Art des aufgetretenen Problems.

Wenn für die Datenbank der dynamische Speicher aktiviert ist und in den Speicherpfaden, die der Bereitschaftsdatenbank zugeordnet sind, kein Speicherplatz zur Verfügung steht, führen Sie die folgenden Schritte aus:

1. Stellen Sie Speicherplatz in den Speicherpfaden zur Verfügung, indem Sie die Dateisysteme erweitern, oder indem Sie unnötige Dateien, bei denen es sich nicht um DB2-Dateien handelt, in ihnen entfernen.
2. Führen Sie eine ROLLFORWARD-Operation für den Tabellenbereich bis zum Ende der Protokolle durch.

Falls das Hinzufügen oder Erweitern von Containern als Teil der Anwendung der Protokolle auf die Bereitschaftsdatenbank nicht durchgeführt werden konnte: Wenn die erforderlichen Backup-Images und Protokolldateiarchive verfügbar sind, können Sie den Tabellenbereich möglicherweise wiederherstellen, indem Sie zunächst die Anweisung SET TABLESPACE CONTAINERS mit der Option IGNORE ROLLFORWARD CONTAINER OPERATIONS und anschließend den Befehl **ROLLFORWARD** eingeben.

Die Primär- und die Bereitschaftsdatenbank müssen jedoch nicht denselben Datenbankpfad aufweisen. Werden relative Containerpfade verwendet, wird derselbe relative Pfad möglicherweise unterschiedlichen absoluten Containerpfaden in der Primär- und der Bereitschaftsdatenbank zugeordnet. Wenn sich also die Primär- und die Bereitschaftsdatenbank auf demselben Computer befinden, müssen alle Tabellenbereichscontainer mit relativen Pfaden definiert werden, sodass sie verschiedenen Pfaden für die Primärdatenbank und die Bereitschaftsdatenbank zugeordnet werden.

HADR- und NAT-Unterstützung

NAT (Network Address Translation, Netzadressumsetzung) wird in HADR-Umgebungen unterstützt und normalerweise für Firewalls und Sicherheitszwecke verwendet, da NAT die reale Adresse des Servers verdeckt.

In einer HADR-Konfiguration wird ein Abgleich der lokalen und fernen Hostkonfigurationen auf den Primär- und Bereitschaftsknoten durchgeführt, um sicherzustellen, dass diese korrekt sind. In einer NAT-Umgebung hat ein Host eine bestimmte IP-Adresse, über die er sich selbst identifiziert, während er für andere Hosts über eine andere IP-Adresse identifiziert wird. Dieses Verhalten hat zur Folge, dass der HADR-Hostabgleich nur dann erfolgreich ist, wenn die Registrierdatenbankvariable **DB2_HADR_NO_IP_CHECK** auf ON gesetzt ist. Bei Verwendung dieser

Einstellung wird der Hostabgleich umgangen, sodass die Primär- und die Bereitschaftsdatenbank in der Lage sind, eine Verbindung in einer NAT-Umgebung herzustellen.

Wenn keine NAT-Umgebung verwendet wird, legen Sie für die Registrierdatenbankvariable **DB2_HADR_NO_IP_CHECK** die Standardeinstellung OFF fest. Eine Inaktivierung des Abgleichs schwächt die Prüfung Ihrer Konfiguration durch HADR.

Hinweise zum Modus für mehrere HADR-Bereitschaftsdatenbanken

Wenn in einer NAT-Umgebung eine Konfiguration mit mehreren Bereitschaftsdatenbanken verwendet wird, müssen die Einstellungen für **hadr_local_host** und **hadr_local_svc** der einzelnen Bereitschaftsdatenbanken in **hadr_target_list** der Primärdatenbank aufgelistet sein, damit die Primärdatenbank Verbindungen von diesen Bereitschaftsdatenbanken akzeptieren kann.

Im Modus für mehrere Bereitschaftsdatenbanken prüft die Bereitschaftsdatenbank normalerweise beim Start, ob sich die Einstellungen für **hadr_remote_host** und **hadr_remote_svc** in **hadr_target_list** befinden. Dadurch wird sichergestellt, dass die alte Primärdatenbank bei einem Rollenwechsel zur neuen Bereitschaftsdatenbank werden kann. In NAT-Szenarios ist diese Prüfung nur dann erfolgreich, wenn die Registrierdatenbankvariable **DB2_HADR_NO_IP_CHECK** auf ON gesetzt ist. Da diese Prüfung umgangen wird, überprüft die Bereitschaftsdatenbank erst nach Herstellung der Verbindung zur Primärdatenbank, ob die Einstellungen für **hadr_local_host** und **hadr_local_svc** der Primärdatenbank in **hadr_target_list** der Bereitschaftsdatenbank vorhanden sind. Dabei wird sichergestellt, dass der Rollenwechsel bei diesem Paar vorgenommen werden kann.

Anmerkung: Wenn die Registrierdatenbankvariable **DB2_HADR_NO_IP_CHECK** auf ON gesetzt ist, werden **hadr_remote_host** und **hadr_remote_svc** nicht automatisch aktualisiert.

In einer Konfiguration mit mehreren Bereitschaftsdatenbanken sollte **DB2_HADR_NO_IP_CHECK** für alle Datenbanken definiert werden, die möglicherweise über eine NAT-Grenze hinweg Verbindungen zu anderen Datenbanken herstellen. Wenn eine Datenbank auf keinen Fall eine NAT-Grenze überschreitet, um eine Verbindung zu einer anderen Datenbank herzustellen (d. h. es wurde kein entsprechender Link konfiguriert), sollte die Registrierdatenbankvariable für diese Datenbank nicht definiert werden. **DB2_HADR_NO_IP_CHECK** verhindert, dass eine Bereitschaftsdatenbank nach einer Übernahme die neue Primärdatenbank automatisch erkennt. In diesem Fall muss die Bereitschaftsdatenbank manuell rekonfiguriert werden, damit sie eine Verbindung zur neuen Primärdatenbank herstellen kann.

Einschränkungen für High Availability Disaster Recovery (HADR)

Um eine optimale Leistung mit der High Availability Disaster Recovery (HADR) zu erreichen ist es wichtig, die HADR-Einschränkungen bereits in Analyse und Entwurf Ihrer hoch verfügbaren DB2-Datenbanklösung zu berücksichtigen.

Die folgende Liste enthält eine Zusammenfassung der Einschränkungen für HADR:

- HADR wird nicht in einer Umgebung mit partitionierten Datenbanken unterstützt.
- HADR wird in DB2 pureScale-Umgebungen nicht unterstützt.

- Die Primär- und die Bereitschaftsdatenbanken müssen über dieselbe Betriebssystemversion und dieselbe Version des DB2-Datenbanksystems verfügen, bis auf einen kurzen Zeitraum, während dessen ein schrittweiser Upgrade ausgeführt wird.
- Die DB2-Datenbanksystemsoftware muss auf der Primärdatenbank und der Bereitschaftsdatenbank dieselbe Bitgröße haben (32 oder 64 Bit).
- Clients können keine Verbindung zur Bereitschaftsdatenbank herstellen, es sei denn, Sie haben die die Funktion für Leseoperationen in der Bereitschaftsdatenbank aktiviert. Diese Funktion gibt Clients die Möglichkeit, eine Verbindung zur aktiven Bereitschaftsdatenbank herzustellen und Abfragen auszuführen, die Lesezugriff erfordern.
- Wenn die Funktion für Leseoperationen in der Bereitschaftsdatenbank aktiviert ist, können keine Operationen durchgeführt werden, die einen Protokollsatz in die Bereitschaftsdatenbank schreiben. Nur Leseclients können eine Verbindung zur aktiven Bereitschaftsdatenbank herstellen.
- Wenn die Funktion für Leseoperationen in der Bereitschaftsdatenbank aktiviert ist, sind Schreiboperationen in der Bereitschaftsdatenbank nicht zulässig, die den Datenbankinhalt ändern würden. Jeder asynchrone Thread, wie z. B. die Erfassung statistischer Echtzeitdaten, der automatische Rebuild von Indizes und Dienstprogramme, die versuchen, Datenbankobjekte zu ändern, werden nicht unterstützt. Die Erfassung statistischer Echtzeitdaten und der automatische Rebuild von Indizes dürfen nicht in der Bereitschaftsdatenbank ausgeführt werden.
- Protokolldateien werden nur von der Primärdatenbank archiviert.
- Der Speichermanager für automatische Leistungsoptimierung (STMM) kann jeweils nur für die Primärdatenbank ausgeführt werden. Nach dem Starten der Primärdatenbank oder dem Konvertieren einer Bereitschaftsdatenbank in eine Primärdatenbank durch Übernahme wird die STMM-EDU (Self Tuning Memory Manager Engine-Dispatchable-Unit) möglicherweise erst bei der ersten ankommenden Clientverbindungsanforderung gestartet.
- Backup-Operationen in der Bereitschaftsdatenbank werden nicht unterstützt.
- Nicht protokollierte Operationen (z. B. wie Änderungen an Datenbankkonfigurationsparametern und der Datei des Recoveryprotokolls) sowie LOB-Spalten mit der Option `NOT LOGGED` werden nicht in die Bereitschaftsdatenbank repliziert.
- Ladeoperationen mit der Option **`COPY`** `NO` werden nicht unterstützt.
- HADR unterstützt die Verwendung einer unformatierten Ein-/Ausgabe (direkter Plattenzugriff) für Datenbankprotokolldateien nicht. Wenn HADR durch den Befehl **`START HADR`** gestartet wird oder die Datenbank mit konfiguriertem HADR aktiviert (erneut gestartet) wird und Protokolle auf Roheinheiten festgestellt werden, schlägt der entsprechende Befehl fehl.
- Ein Server mit föderierten Datenbanken bietet keine vollständige Unterstützung von HADR. Bei einem einphasigen Commit kann eine HADR-Datenbank als Server mit föderierten Datenbanken (Transaktionsmanager) oder als Datenquelle (Ressourcenmanager) fungieren, was eine Clientweiterleitungskonfiguration erforderlich macht. Bei einem zweiphasigen Commit kann eine HADR-Datenbank nur als Datenquelle fungieren; aufgrund von Einschränkungen in Bezug auf die Datenkonsistenz ist eine Funktionsweise der HADR-Datenbank als Server mit föderierten Datenbanken hier nicht möglich.
- HADR unterstützt keine Endlosprotokollierung.
- Die Systemuhr der HADR-Primärdatenbank muss mit der Systemuhr der HADR-Bereitschaftsdatenbank synchronisiert werden.

Terminieren von Verwaltungs- und Wartungsaktivitäten für hohe Verfügbarkeit

Ihre DB2-Datenbanklösung erfordert regelmäßige Verwaltung und Wartung. Dazu gehören Aktivitäten wie Software- oder Hardware-Upgrades, Optimierung der Datenbankleistung, Datenbankbackups, Statistikerfassung und Überwachung zu Geschäftszwecken. Sie müssen die Beeinträchtigung der Verfügbarkeit Ihrer Datenbanklösung durch diese Verwaltungs- und Wartungsaktivitäten minimieren.

Vorbereitende Schritte

Bevor Sie Ihre Verwaltungs- und Wartungsaktivitäten terminieren können, müssen Sie feststellen, welche dieser Aktivitäten für Ihre Datenbanklösung ausgeführt werden müssen.

Vorgehensweise

Führen Sie die folgenden Schritte aus, um die Verwaltung und Wartung zu terminieren:

1. Identifizieren Sie Zeiträume mit niedriger Datenbankaktivität.

   Verwaltungs- und Wartungsaktivitäten werden am besten für Zeiten mit geringer Beanspruchung der Datenbank terminiert (d. h. Zeiträume, in denen am wenigsten Benutzeranwendungen Anforderungen an das Datenbanksystem senden). Je nach Typ der von Ihnen erstellten Geschäftsanwendung kann es sogar Zeiträume geben, in denen überhaupt keine Benutzeranwendung auf das Datenbanksystem zugreift.
2. Kategorisieren Sie die Verwaltungs- und Wartungsaktivitäten nach folgenden Kriterien:
   - Verwaltung und Wartung können automatisiert werden
   - Die Datenbanklösung muss offline geschaltet werden, während Sie die Verwaltung/Wartung durchführen
   - Die Verwaltung/Wartung kann durchgeführt werden, während die Datenbank online geschaltet ist
3. Konfigurieren Sie für die Verwaltungsaktivitäten, die automatisiert werden können, die automatische Verwaltung mithilfe einer der folgenden drei Methoden:
   - Verwenden Sie den Konfigurationsparameter **auto_maint**.
   - Verwenden Sie die gespeicherte Systemprozedur AUTOMAINT_SET_POLICY oder AUTOMAINT_SET_POLICYFILE
4. Wenn es für eine der auszuführenden Wartungs- oder Verwaltungsaktivitäten erforderlich ist, den Datenbankserver offline zu schalten, terminieren Sie diese Offlineverwaltungsaktivitäten für Zeiträume mit geringer Beanspruchung der Datenbank.
5. Bei Wartungs- und Verwaltungsaktivitäten, die ausgeführt werden können, während der Datenbankserver online geschaltet ist:
   - Ermitteln Sie, wie stark die Verfügbarkeit Ihrer Datenbank durch das Ausführen der Onlineverwaltungsaktivitäten beeinträchtigt würde.
   - Terminieren Sie diese Onlineverwaltungsaktivitäten so, dass die Beeinträchtigung der Verfügbarkeit Ihres Datenbanksystems durch das Ausführen der Verwaltungsaktivitäten minimiert wird.

Beispiel: Terminieren Sie Onlineverwaltungsaktivitäten für Zeiten mit geringer Datenbankbeanspruchung, und verwenden Sie Mechanismen zum Drosseln bzw. zum Ausgleichen des Umfangs an Systemressourcen, die von den Verwaltungsaktivitäten beansprucht werden.

Konfigurieren einer Richtlinie für automatische Verwaltung mit SYSPROC.AUTOMAINT_SET_POLICY oder SYSPROC.AUTOMAINT_SET_POLICYFILE

Sie können die gespeicherten Systemprozeduren AUTOMAINT_SET_POLICY und AUTOMAINT_SET_POLICYFILE verwenden, um Ihre automatische Verwaltungsrichtlinie für eine Datenbank zu konfigurieren.

Vorgehensweise

Führen Sie die folgenden Schritte aus, um eine Richtlinie für automatische Verwaltung für eine Datenbank zu konfigurieren:

1. Stellen Sie eine Verbindung zur Datenbank her
2. Rufen Sie AUTOMAINT_SET_POLICY oder AUTOMAINT_SET_POLICYFILE auf
   - Die für AUTOMAINT_SET_POLICY erforderlichen Parameter sind die Folgenden:
     a. Der Verwaltungstyp. Er gibt den Typ der zu konfigurierenden automatischen Verwaltungsaktivität an.
     b. Verweis auf ein großes Binärobjekt (BLOB). Dieses Objekt gibt die Richtlinie für automatische Verwaltung im XML-Format an.
   - Die für AUTOMAINT_SET_POLICYFILE erforderlichen Parameter sind die Folgenden:
     a. Der Verwaltungstyp. Er gibt den Typ der zu konfigurierenden automatischen Verwaltungsaktivität an.
     b. Der Name einer XML-Datei. Diese Datei gibt die Richtlinie für automatische Verwaltung an.

   Gültige Werte für den Verwaltungstyp sind:
   - AUTO_BACKUP - automatisches Backup
   - AUTO_REORG - automatische Reorganisation von Tabellen und Indizes
   - AUTO_RUNSTATS - automatische **RUNSTATS**-Operationen für Tabellen
   - MAINTENANCE_WINDOW - Verwaltungsfenster

Nächste Schritte

Sie können die gespeicherten Systemprozeduren AUTOMAINT_GET_POLICY und AUTOMAINT_GET_POLICYFILE verwenden, um die für eine Datenbank konfigurierte automatische Verwaltungsrichtlinie abzurufen.

XML-Richtlinienspezifikation für automatische Verwaltung für AUTOMAINT_SET_POLICY oder AUTOMAINT_SET_POLICYFILE - Beispiel

Unabhängig davon, ob Sie für die Spezifikation Ihrer Richtlinie für die automatische Verwaltung AUTOMAINT_SET_POLICY oder AUTOMAINT_SET_POLICYFILE verwenden, muss dies im XML-Format geschehen. Es sind Beispieldateien vorhanden, die veranschaulichen, wie Sie Ihre Richtlinie für die automatische Verwaltung in XML angeben müssen. Auf Linux- und UNIX-Betriebssystemen be-

finden sich die Beispieldateien im Verzeichnis SQLLIB/samples/automaintcfg. Bei Windows-Betriebssystemen befinden sich die Beispieldateien im Verzeichnis SQLLIB\samples\automaintcfg.

Der zweite Parameter, den Sie an die gespeicherte Systemprozedur AUTOMAINT_SET_POLICY übergeben, ist ein XML enthaltendes großes Binärobjekt (BLOB), das Ihre gewünschte Richtlinie für die automatische Verwaltung angibt. Der zweite Parameter, den Sie an die gespeicherte Systemprozedur AUTOMAINT_SET_POLICYFILE übergeben, ist der Name einer XML-Datei, die Ihre gewünschte Richtlinie für die automatische Verwaltung angibt. Die XML-Elemente, die im großen Binärobjekt gültig sind, welches Sie an AUTOMAINT_SET_POLICY übergeben, sind dieselben Elemente, die in der XML-Datei gültig sind, die Sie an AUTOMAINT_SET_POLICYFILE übergeben.

In den Verzeichnissen mit den Beispielen (SQLLIB/samples/automaintcfg in Linux- und UNIX-Umgebungen bzw. SQLLIB\samples\automaintcfg in Windows-Umgebungen) gibt es vier XML-Dateien mit Beispielspezifikationen für Ihre Richtlinie für die automatische Verwaltung:

DB2MaintenanceWindowPolicySample.xml

Beschreibt die Spezifikation eines Verwaltungsfensters für die Zeitangabe, innerhalb deren der Datenbankmanager die automatische Verwaltung terminieren soll.

DB2AutoBackupPolicySample.xml

Beschreibt, wie die Spezifikation für den Datenbankmanager zum Ausführen eines automatischen Backups aussehen soll.

DB2AutoReorgPolicySample.xml

Beschreibt, wie die Spezifikation für den Datenbankmanager zum Ausführen der automatischen Tabellen- und Indexreorganisation aussehen soll.

DB2DefaultAutoRunstatsPolicySample.xml

Beschreibt, wie die Spezifikation für den Datenbankmanager zum Ausführen der automatischen **runstats**-Operationen für Tabellen aussehen soll.

Sie können Ihre eigene XML-Richtlinienspezifikation für die automatische Verwaltung erstellen, indem Sie das XML-Format aus den Beispieldateien kopieren und die XML gemäß den Anforderungen Ihres Systems modifizieren.

Konfigurieren der Optionen zur Datenbankprotokollierung

Mit den Konfigurationsparameter für die Datenbankprotokollierung können Sie Optionen zur Datenprotokollierung für Ihre Datenbank angeben, wie zum Beispiel den Typ der zu verwendenden Protokollierung, die Größe der Protokolldateien und die Position, an der die Protokolldateien gespeichert werden sollen.

Vorbereitende Schritte

Zur Konfiguration von Optionen der Datenbankprotokollierung müssen Sie über die Berechtigung SYSADM, SYSCTRL oder SYSMAINT verfügen.

Informationen zu diesem Vorgang

Sie können Optionen für die Datenbankprotokollierung über den Befehlszeilenprozessor (CLP) mit dem Befehl **UPDATE DATABASE CONFIGURATION** oder durch Aufrufen der Anwendungsprogrammierschnittstelle (API) db2CfgSet konfigurieren.

Vorgehensweise

- Gehen Sie wie folgt vor, um Optionen für die Datenbankprotokollierung über den Befehlszeilenprozessor mit dem Befehl **UPDATE DATABASE CONFIGURATION** zu konfigurieren:
  1. Geben Sie an, ob Sie eine Umlaufprotokollierung oder eine Archivprotokollierung wünschen. Wenn Sie eine Umlaufprotokollierung wünschen, müssen die Datenbankkonfigurationsparameter **LOGARCHMETH1** und **LOGARCHMETH2** auf den Wert OFF gesetzt werden. Diese Einstellung ist der Standardwert. Wenn Sie eine Archivprotokollierung nutzen möchten, müssen Sie mindestens einen dieser Datenbankkonfigurationsparameter auf einen anderen Wert als OFF setzen. Wenn Sie zum Beispiel die Archivprotokollierung verwenden und die archivierten Protokolle auf Platte speichern wollen, führen Sie den folgenden Befehl aus:

     ```
     db2 update db configuration for mydb using logarchmeth1
           disk:/u/dbuser/archived_logs
     ```

     Die archivierten Protokolle werden in einem Verzeichnis mit dem Namen '/u/dbuser/archived_logs' gespeichert.
  2. Geben Sie für die anderen Konfigurationsparameter zur Datenbankprotokollierung Werte nach Bedarf an. Folgendes sind weitere Konfigurationsparameter für die Datenbankprotokollierung:
     - **archretrydelay**
     - **blk_log_dsk_ful**
     - **failarchpath**
     - **logarchcompr1**
     - **logarchcompr2**
     - **logarchmeth1**
     - **logarchmeth2**
     - **logarchopt1**
     - **logarchopt2**
     - **logbufsz**
     - **logfilsiz**
     - **logprimary**
     - **logsecond**
     - **max_log**
     - **mirrorlogpath**
     - **newlogpath**
     - **mincommit**
     - **numarchretry**
     - **num_log_span**
     - **overflowlogpath**

     Weitere Informationen zu diesen Konfigurationsparametern für die Datenbankprotokollierung finden Sie in „Konfigurationsparameter für die Datenbankprotokollierung” auf Seite 83.
- Gehen Sie wie folgt vor, um Optionen für die Datenbankprotokollierung mit IBM Data Studio unter Verwendung des Taskassistenten für den Befehl **UPDATE DATABASE CONFIGURATION** zu konfigurieren.

Konfigurationsparameter für die Datenbankprotokollierung

Die Datenbankprotokollierung ist eins der Schlüsselelemente aller Strategien für die Hochverfügbarkeit. Sie können Datenbankprotokolle verwenden, um Transaktionsinformationen aufzuzeichnen, primäre und sekundäre Datenbanken (Bereitschaftsdatenbanken) zu synchronisieren und eine sekundäre Datenbank aktualisierend wiederherzustellen, die die Funktion einer fehlgeschlagenen Primärdatenbank übernommen hat.

Sie müssen eine Reihe von Datenbankkonfigurationsparametern festlegen, um diese Datenbankprotokollierungsaktivitäten zu konfigurieren.

Wiederholungsintervall für Protokollarchivierung (archretrydelay)
Gibt die Zeit in Sekunden an, die jeweils zwischen einem fehlgeschlagenen Versuch, Protokolldateien zu archivieren, und dem nächsten Versuch gewartet werden soll. Der Standardwert ist 20.

Bei voller Protokollplatte blockieren (blk_log_dsk_ful)

Dieser Konfigurationsparameter kann so festgelegt werden, dass keine Fehler aufgrund erschöpfter Festplattenkapazität generiert werden, wenn der DB2-Datenbankmanager keine neue Protokolldatei im Pfad für aktive Protokolldateien erstellen kann. Stattdessen wird der DB2-Datenbankmanager alle fünf Minuten versuchen, die Protokolldatei zu erstellen, bis diese Datei erfolgreich erstellt wurde. Nach jedem Versuch schreibt der DB2-Datenbankmanager eine Nachricht in das Protokoll mit den Benachrichtigungen für die Systemverwaltung. Sie können nur durch Überwachen des Protokolls mit den Benachrichtigungen für die Systemverwaltung feststellen, ob Ihre Anwendung deshalb blockiert ist, weil für Protokolldateien kein Plattenplatz mehr vorhanden ist. Bis zur erfolgreichen Erstellung der Protokolldatei kann keine Benutzeranwendung, die eine Aktualisierung von Tabellendaten ausführen will, eine Transaktion festschreiben. Abfragen mit Lesezugriff sind hiervon möglicherweise nicht direkt betroffen. Wenn eine Abfrage jedoch auf Daten zugreifen will, die aufgrund einer Aktualisierungsanforderung gesperrt sind, oder auf eine Datenseite, die von der aktualisierenden Anwendung im Pufferpool fixiert wurde, werden Abfragen mit Lesezugriff ebenfalls blockiert.

Wenn Sie **blk_log_dsk_ful** auf YES setzen, blockieren Anwendungen, sobald der DB2-Datenbankmanager einen Fehler wegen einer vollen Protokollplatte feststellt. Sie können diesen Fehler anschließend beheben und die Anwendung kann fortgesetzt werden. Sie können auf einem vollen Datenträger Speicher freigeben, indem Sie alte Protokolldateien auf ein anderes Dateisystem verschieben, indem Sie die Größe des Dateisystems erhöhen, sodass blockierte Anwendungen beendet werden können, oder indem Sie Fehler bei der Protokollarchivierung untersuchen und beheben.

Wenn Sie **blk_log_dsk_ful** auf NO setzen, schlägt eine Transaktion, die einen Fehler wegen einer vollen Protokollplatte empfängt, fehl und wird zurückgesetzt.

Alternativpfad für Archivprotokolldatei (failarchpath)
Gibt ein alternatives Verzeichnis für die Archivprotokolldateien an, falls ein Problem mit dem normalen Archivpfad besteht (wenn auf diesen beispielsweise nicht zugegriffen werden kann oder er voll ist). Dieses Verzeichnis ist ein Bereich zur temporären Speicherung der Protokolldateien, bis die fehlgeschlagene Protokollarchivierungsmethode wieder zur Verfügung steht, woraufhin die Protokolldateien aus diesem Verzeichnis in den Pfad versetzt werden, der in der ursprünglichen Protokollarchivierung angege-

ben ist. Durch Versetzen der Protokolldateien in dieses temporäre Verzeichnis können Situationen vermieden werden, in denen der Speicherplatz im Protokollverzeichnis erschöpft ist. Für diesen Parameter muss ein vollständig qualifiziertes, vorhandenes Verzeichnis angegeben werden.

Komprimierung archivierter Protokolldateien für primäre Datenbank (logarchcompr1), Komprimierung archivierter Protokolldateien für sekundäre Datenbank (logarchcompr2)

In bestimmten Fällen wird über diese Parameter gesteuert, ob der Datenbankmanager archivierte Protokolldateien komprimiert. Durch die Komprimierung archivierter Protokolldateien können die Speicherkosten in Verbindung mit diesen Dateien gesenkt werden.

Gültige Werte für diese Parameter:

OFF Dieser Wert gibt an, dass archivierte Protokolldateien nicht komprimiert werden. Der Standardwert ist `OFF`.

ON Dieser Wert gibt an, dass archivierte Protokolldateien komprimiert werden. Bei dynamischer Festlegung werden bereits archivierte Protokolldateien nicht komprimiert.

Anmerkung:

1. Wenn der Konfigurationsparameter **logarchmeth1** auf einen anderen Wert als `DISK`, `TSM` oder `VENDOR` gesetzt ist, bleibt die Komprimierung der Protokollarchive unabhängig von der Einstellung des Konfigurationsparameters **logarchcompr1** wirkungslos.
2. Wenn der Konfigurationsparameter **logarchmeth2** auf einen anderen Wert als `DISK`, `TSM` oder `VENDOR` gesetzt ist, bleibt die Komprimierung der Protokollarchive unabhängig von der Einstellung des Konfigurationsparameters **logarchcompr2** wirkungslos.

Erste Protokollarchivierungsmethode (logarchmeth1), zweite Protokollarchivierungsmethode (logarchmeth2)

Diese Parameter weisen den Datenbankmanager an, Protokolldateien in einem anderen Pfad als dem Pfad für aktive Protokolldateien zu speichern. Wenn beide Parameter angegeben werden, wird jede Protokolldatei aus dem mit dem Konfigurationsparameter **logpath** angegebenen Pfad für aktive Protokolldateien doppelt archiviert. Dies bedeutet, dass Sie zwei identische Kopien der Archivprotokolldateien aus dem Protokollpfad an zwei verschiedenen Zielen erhalten. Wenn Sie die Protokollspiegelung mit dem Konfigurationsparameter **mirrorlogpath**) angeben, archiviert der Konfigurationsparameter **logarchmeth2** Protokolldateien aus dem Pfad für Protokollspiegelung anstatt zusätzliche Kopien der Protokolldateien im Pfad für aktive Protokolldateien zu archivieren. Dies bedeutet, dass Sie zwei separate Kopien der Archivprotokolldateien an zwei verschiedenen Zielen erhalten: eine Kopie aus dem Protokollpfad und eine Kopie aus dem Pfad für Protokollspiegelung.

Gültige Werte für diese Parameter:

OFF Dieser Wert gibt an, dass die Protokollarchivierungsmethode nicht verwendet werden soll. Wenn sowohl der Konfigurationsparameter **logarchmeth1** als auch der Konfigurationsparameter **logarchmeth2** auf `OFF` gesetzt wird, heißt dies, dass die Datenbank Umlaufprotokolle verwendet. Sie kann in diesem Fall nicht aktualisierend wiederhergestellt werden. Der Standardwert ist `OFF`.

LOGRETAIN
Gibt an, dass aktive Protokolldateien beibehalten und zu Onlinearchivprotokolldateien werden, die bei der aktualisierenden Recovery verwendet werden.

USEREXIT
Gibt an, dass die Protokollierung mit Protokollspeicherung ausgeführt wird und dass ein Benutzerexitprogramm verwendet werden muss, um die Protokolldateien zu archivieren und abzurufen. Protokolldateien werden archiviert, sobald sie voll sind. Sie werden abgerufen, wenn das Dienstprogramm für die aktualisierende Recovery sie verwenden muss, um ein Restore für eine Datenbank auszuführen.

DISK Auf diesen Wert muss ein Doppelpunkt (:) folgen und darauf der vollständig qualifizierte Name des Pfads, in dem die Protokolldateien archiviert werden sollen. Wenn Sie beispielsweise den Konfigurationsparameter **logarchmeth1** auf `DISK:/u/dbuser/archived_logs` setzen, werden die archivierten Protokolldateien unter oder in das Verzeichnis `/u/dbuser/archived_logs/`*instanzname*`/`*datenbankname*`/knoten`*xxxx*`/LOGSTREAM`*xxxx*`/C`*xxxxxxx* gestellt.

Anmerkung: Wenn Sie auf Band archivieren, können Sie zum Speichern und Abrufen der Protokolldateien das Dienstprogramm **db2tapemgr** verwenden.

TSM Wenn dieser Wert ohne weitere Konfigurationsparameter angegeben wird, werden Protokolldateien unter Verwendung der Standardmanagementklasse auf dem lokalen Tivoli Storage Manager-Server (TSM) archiviert. Folgen auf diesen Wert ein Doppelpunkt (:) und eine TSM-Managementklasse, werden die Protokolldateien unter Verwendung der angegebenen Managementklasse archiviert.

VENDOR
Gibt an, dass zur Archivierung der Protokolldateien eine Bibliothek eines anderen Lieferanten verwendet wird. Auf diesen Wert müssen ein Doppelpunkt (:) und der Name der Bibliothek folgen. Die in der Bibliothek zur Verfügung gestellten APIs müssen die Backup- und Restore-APIs für Produkte anderer Lieferanten verwenden.

Anmerkung:

1. Wenn **logarchmeth1** oder **logarchmeth2** auf einen anderen Wert als `OFF` gesetzt ist, ist die Datenbank für die aktualisierende Recovery konfiguriert.

Optionen für logarchmeth1 (logarchopt1), Optionen für logarchmeth2 (logarchopt2) Gibt eine Zeichenfolge an, die an den TSM-Server oder an APIs anderer Lieferanten übergeben wird.

Verwenden Sie für TSM-Umgebungen diesen Parameter zum Aktivieren der Datenbank, um Protokolle abzurufen, die auf einem anderen TSM-Knoten, von einem anderen TSM-Benutzer oder in TSM-Umgebungen mit Proxy-Knoten wie beispielsweise in DB2 pureScale-Umgebungen generiert wurden. Sie müssen die Zeichenfolge in einem der folgenden Formate angeben:

- Zum Abrufen von Protokollen, die auf einem anderen TSM-Knoten generiert wurden und wenn der TSM-Server nicht für die Unterstützung von Proxy-Knoten-Clients konfiguriert ist:

  "-fromnode=*knotenname*"

- Zum Abrufen von Protokollen, die von einem anderen TSM-Benutzer generiert wurden und wenn der TSM-Server nicht für die Unterstützung von Proxy-Knoten-Clients konfiguriert ist:

  "-fromowner=*eignername*"

- Zum Abrufen von Protokollen, die auf einem anderen TSM-Knoten und von einem anderen TSM-Benutzer generiert wurden und wenn der TSM-Server nicht für die Unterstützung von Proxy-Knoten-Clients konfiguriert ist:

  "-fromnode=*knotenname* -fromowner=*eignername*"

- Zum Abrufen von Protokollen, die in Konfigurationen mit Client-Proxy-Knoten generiert wurden, wie beispielsweise in DB2 pureScale-Umgebungen, in denen mehrere Member an denselben Daten arbeiten:

  "-asnodename=*proxy-knoten*"

  Dabei ist *knotenname* der Name des TSM-Knotens, auf dem die Protokolldateien ursprünglich archiviert wurden, *eignername* ist der Name des TSM-Benutzers, von dem die Protokolldateien ursprünglich archiviert wurden, und *proxy-knoten* ist der Name des gemeinsam genutzten TSM-Proxy-Zielknotens. Jedes Protokollarchivierungsoptionsfeld entspricht einer der Protokollarchivierungsmethoden: **logarchopt1** wird mit **logarchmeth1** verwendet und **logarchopt2** mit **logarchmeth2**.

Anmerkung:

- Wird die TSM-Option -asnodename verwendet, wird zum Speichern der Daten nicht der Name des Knotens (knotenname) der einzelnen Member verwendet. Die Daten werden stattdessen mithilfe des Namens des gemeinsam genutzten TSM-Zielknotens gespeichert, der von allen Membern innerhalb einer DB2 pureScale-Instanz verwendet wird.
- Die Optionen -fromnode und -fromowner sind nicht kompatibel mit der Option -asnodename und sie können nicht zusammen verwenden werden. Verwenden Sie die Option -asnodename für TSM-Konfigurationen mit Proxy-Knoten und die beiden anderen Optionen für andere Arten von TSM-Konfigurationen. Weitere Informationen hierzu finden Sie im Abschnitt „Konfigurieren eines Tivoli Storage Manager-Clients” auf Seite 473.

Protokollpuffer (logbufsz)

Mit diesem Parameter kann die Speichermenge angegeben werden, die als Puffer für Protokollsätze verwendet werden soll, bevor diese Protokollsätze auf die Festplatte geschrieben werden. Die Protokollsätze werden auf die Platte geschrieben, sobald eines der folgenden Ereignisse eintritt:

- Eine Transaktion wird festgeschrieben.
- Die Größe des Protokollpuffers reicht nicht mehr aus.
- Ein anderes internes Datenbankmanagerereignis tritt ein.

Die Erhöhung der Protokollpuffergröße kann zu einer effizienteren E/A-Aktivität in Bezug auf die Protokollierung führen, da die Protokollsätze nun in größeren Abständen und dabei in größeren Mengen auf Platte geschrieben werden. Die Recovery kann bei einem höheren Wert für die Protokollpuffergröße jedoch länger dauern. Möglicherweise können Sie auch

eine höhere Einstellung für die Protokollpuffergröße (**logbufsz**) verwenden, um die Anzahl der Lesevorgänge von der Protokollplatte zu reduzieren. (Zum Ermitteln, ob dies für Ihr System von Vorteil wäre, verwenden Sie das Monitorelement '**log_reads**, mit dem Sie überprüfen, ob das Lesen von der Protokollplatte eine bedeutende Größe darstellt).

Protokolldateigröße (logfilsiz)
Dieser Parameter gibt die Größe jeder konfigurierten Protokolldatei an, und zwar als Anzahl 4-KB-Seiten.

Es besteht eine logische Begrenzung von 1024 GB pro Protokolldatenstrom für den konfigurierbaren Gesamtspeicherplatz für aktive Protokolldateien. Diese Begrenzung ergibt sich aus dem oberen Grenzwert für jede Protokolldatei, der bei 4 GB liegt, und der maximalen Gesamtanzahl der primären und sekundären Protokolldateien, die bei 256 liegt.

Die Größe der Protokolldatei hat eine direkte Auswirkung auf die Leistung. Wenn von einem Protokoll zu einem nächsten gewechselt werden muss, ist dies mit einem Leistungsaufwand verbunden. Aus reiner Leistungsperspektive betrachtet wäre das Protokoll daher umso besser, je größer es wäre. Dieser Parameter gibt auch die Protokolldateigröße für die Archivierung an. In diesem Fall ist eine größere Protokolldatei nicht unbedingt leistungsfördernd, da sie das Fehlerrisiko erhöht bzw. bei der Protokollübertragung zu Verzögerungen führen kann. Für den Speicherbereich für aktive Protokolldateien könnte es sich eher empfehlen, eine größere Anzahl kleinerer Protokolldateien zu verwenden. Wenn beispielsweise zwei sehr große Protokolldateien verwendet werden und eine Transaktion nahe am Ende einer der Protokolldateien startet, ist nur noch die Hälfte des Speicherbereichs verfügbar.

Jedes Mal, wenn eine Datenbank inaktiviert wird (d. h. alle Verbindungen zur Datenbank werden beendet), wird die gerade verwendete Protokolldatei abgeschnitten. Wenn eine Datenbank häufig inaktiviert wird, sollte eine eher kleine Protokolldateigröße gewählt werden, da der DB2-Datenbankmanager andernfalls eine große Datei generieren würde, die abgeschnitten wird. Sie können den Befehl **ACTIVATE DATABASE** verwenden, um diesen Aufwand zu vermeiden. Dieser Befehl verhindert die automatische Datenbankinaktivierung, die eintreten würde, wenn der letzte Client die Verbindung trennt.

Angenommen, Sie haben eine Anwendung, die die Datenbank geöffnet hält, um die Verarbeitungszeit beim Öffnen von Datenbanken zu minimieren. In diesem Fall sollte der Wert für die Protokollgröße durch den Zeitraum bestimmt werden, der erforderlich ist, um Kopien von Offlinearchivprotokolldateien anzulegen.

Die Verlustminimierung von Protokolldaten ist ebenfalls ein wichtiger Aspekt beim Definieren der Protokollgröße. Die Archivierung arbeitet gleichzeitig an einer (1) gesamten Protokolldatei. Wenn Sie größere Protokolldateien konfigurieren, vergrößern Sie den Zeitraum zwischen den Archivierungen. Bei einem Defekt des Datenträgers, auf dem sich das Protokoll befindet, gehen wahrscheinlich einige Transaktionsinformationen verloren. Das Verringern der Protokolldateigröße erhöht zwar die Häufigkeit der Archivierungen, kann aber den Informationsverlust bei einem Datenträgerfehler verringern, da zu einem bestimmten Zeitpunkt im Durchschnitt weniger Protokolldaten noch nicht archiviert sind.

Maximales Protokoll pro Transaktion (max_log)
Dieser Parameter zeigt den Prozentsatz des primären Protokollspeicherbe-

reichs an, der von einer Transaktion verbraucht werden kann. Der Wert ist ein Prozentsatz des Werts, der für den Konfigurationsparameter **logprimary** angegeben ist.

Wird der Wert auf 0 gesetzt, besteht keine Begrenzung des Prozentsatzes des gesamten primären Protokollbereichs, den eine Transaktion verbrauchen kann. Verletzt eine Anwendung die Konfiguration **max_log**, wird die Anwendung gezwungen, die Verbindung zur Datenbank zu beenden. Die Transaktion wird rückgängig gemacht.

Sie können dieses Verhalten außer Kraft setzen, indem Sie die Registrierdatenbankvariable **DB2_FORCE_APP_ON_MAX_LOG** auf FALSE setzen. Dies bewirkt, dass Transaktionen, die die Konfiguration **max_log** verletzen, fehlschlagen. Die Anwendung kann weiterhin die von vorherigen Anweisungen in der UOW fertig gestellte Arbeit festschreiben oder die fertig gestellte Arbeit wird rückgängig gemacht, um die UOW zu widerrufen.

Dieser Parameter kann zusammen mit dem Konfigurationsparameter **num_log_span** nützlich sein, wenn ein unbegrenzter aktiver Protokollspeicherbereich aktiviert ist. Wenn die Endlosprotokollierung aktiv ist (d. h., **logsecond** weist den Wert -1 auf), werden Transaktionen nicht auf die Obergrenze der Anzahl von Protokolldateien (**logprimary** + **logsecond**) beschränkt. Wenn der Wert von **logprimary** erreicht wurde, beginnt der DB2-Datenbankmanager mit der Archivierung der aktiven Protokolldateien, statt die Transaktion fehlschlagen zu lassen. Dies kann zu Problemen führen, wenn zum Beispiel eine Transaktion mit langer Ausführungszeit ohne Commit zurückgelassen wurde (vielleicht aufgrund einer Anwendung mit einem logischen Fehler). In einen solchen Fall wächst der Speicherbereich für aktive Protokolldateien immer weiter an, was sich negativ auf die Leistung bei einer Recovery nach einem Systemabsturz auswirken könnte. Sie können dies vermeiden, indem Sie Werte für einen der Konfigurationsparameter **max_log** und **num_log_span** (oder beide) angeben.

Anmerkung: Die folgenden DB2-Befehle sind von der Begrenzung ausgenommen, die vom Konfigurationsparameter **max_log** festgelegt wird: **ARCHIVE LOG**, **BACKUP DATABASE**, **LOAD**, **REORG**, **RESTORE DATABASE** und **ROLLFORWARD DATABASE**.

Pfad für Protokollspiegelung (mirrorlogpath)
Zum Schutz der Protokolle im primären Protokollpfad vor Plattenausfällen oder versehentlichem Löschen können Sie angeben, dass alle Protokolle zusätzlich in einem zweiten Protokollpfad, dem Spiegelprotokollpfad, gespeichert werden. Ändern Sie dazu den Wert dieses Konfigurationsparameters, sodass er auf ein anderes Verzeichnis zeigt. Aktive Protokolldateien, die momentan im Spiegelprotokollpfad gespeichert werden, werden nicht an die neue Position versetzt, wenn die Datenbank für die aktualisierende Recovery konfiguriert ist.

Der Parameter **mirrorlogpath** wirkt sich auch auf das Verhalten der Protokollarchivierung aus, das Sie dazu verwenden können, die Ausfallsicherheit während der aktualisierenden Recovery weiter zu verbessern: Wenn sowohl **mirrorlogpath** als auch **logarchmeth2** angegeben sind, archiviert **logarchmeth2** Protokolldateien aus dem Pfad für Protokollspiegelung anstatt zusätzliche Kopien der Protokolldateien im Pfad für aktive Protokolldateien zu archivieren. Dieses Verhalten der Protokollarchivierung können Sie nutzen, um die Ausfallsicherheit zu verbessern, da eine verwendbare Archivprotokolldatei aus dem Pfad für Protokollspiegelung möglicherweise noch verfügbar ist, um eine Datenbankrecoveryoperation auch dann fortzu-

setzen, auch wenn eine primäre Protokolldatei aufgrund eines Plattenfehlers vor der Archivierung beschädigt wurde.

Da Sie den Protokollpfad ändern können, befinden sich die für die aktualisierende Recovery erforderlichen Protokolle möglicherweise in verschiedenen Verzeichnissen. Sie können den Wert dieses Konfigurationsparameters während der aktualisierenden Recovery ändern, damit Sie auf Protokolldateien aus einem anderen Pfad für Protokollspiegelung zugreifen können.

Sie müssen die Speicherposition der Protokolle verfolgen.

Die Änderungen werden erst dann angewendet, wenn sich die Datenbank wieder in einem konsistenten Status befindet. Der Datenbankstatus wird durch den Konfigurationsparameter **database_consistent** zurückgegeben.

Wenn Sie diesen Konfigurationsparameter inaktivieren wollen, setzen Sie seinen Wert auf DEFAULT.

Anmerkung:

1. Dieser Konfigurationsparameter wird nicht unterstützt, wenn es sich bei dem primären Protokollpfad um eine Roheinheit handelt.
2. Der für diesen Parameter angegebene Wert darf keine Roheinheit sein.
3. In einer DB2 pureScale-Umgebung verarbeitet das erste Member, das eine Verbindung zur Datenbank herstellt oder die Datenbank aktiviert, die Konfigurationsänderungen an diesem Protokollpfadparameter. Der DB2-Datenbankmanager bestätigt, dass der Pfad vorhanden ist und sowohl Lese- als auch Schreibzugriff für diesen Pfad vorhanden sind. Außerdem werden memberspezifische Unterverzeichnisse für die Protokolldateien erstellt. Falls eine dieser Operationen fehlschlägt, weist der DB2-Datenbankmanager den angegebenen Pfad zurück und versetzt die Datenbank mithilfe des alten Pfads in den Onlinestatus. Wird der angegebene Pfad akzeptiert, wird der neue Wert an die übrigen Member weitergegeben. Falls ein Member fehlschlägt, während es versucht, zum neuen Pfad zu wechseln, schlagen nachfolgende Versuche zum Aktivieren der Datenbank oder zum Herstellen einer Verbindung zur Datenbank fehl (SQL5099N). Alle Member müssen denselben Protokollpfad verwenden.

Neuer Protokollpfad (newlogpath)

Die Datenbankprotokolle werden zunächst in folgendem Verzeichnis erstellt: *datenbankpfad*/*instanzname*/*datenbankname*/NODE0000/LOGSTREAM0000. Sie können die Position, an der die aktiven Protokolldateien gespeichert werden (und an der zukünftige Protokolldateien gespeichert werden), ändern, indem Sie den Wert dieses Konfigurationsparameters ändern, sodass er auf ein anderes Verzeichnis oder auf eine Einheit verweist. Aktive Protokolldateien, die momentan im Verzeichnispfad der Datenbankprotokolle gespeichert werden, werden nicht an die neue Position versetzt, wenn die Datenbank für die aktualisierende Recovery konfiguriert ist.

Da Sie den Protokollpfad ändern können, befinden sich die für die aktualisierende Recovery erforderlichen Protokolle möglicherweise in verschiedenen Verzeichnissen bzw. auf verschiedenen Einheiten. Sie können diesen Konfigurationsparameter während der aktualisierenden Recovery ändern, um Zugriff auf Protokolle an mehreren Speicherpositionen zu ermöglichen.

Sie müssen die Speicherposition der Protokolle verfolgen.

Die Änderungen werden erst dann angewendet, wenn sich die Datenbank wieder in einem konsistenten Status befindet. Der Datenbankstatus wird durch den Konfigurationsparameter **database_consistent** zurückgegeben.

Anmerkung: In einer DB2 pureScale-Umgebung verarbeitet das erste Member, das eine Verbindung zur Datenbank herstellt oder die Datenbank aktiviert, die Konfigurationsänderungen an diesem Protokollpfadparameter. Der DB2-Datenbankmanager bestätigt, dass der Pfad vorhanden ist und sowohl Lese- als auch Schreibzugriff für diesen Pfad vorhanden sind. Außerdem werden memberspezifische Unterverzeichnisse für die Protokolldateien erstellt. Falls eine dieser Operationen fehlschlägt, weist der DB2-Datenbankmanager den angegebenen Pfad zurück und versetzt die Datenbank mithilfe des alten Pfads in den Onlinestatus. Wird der angegebene Pfad akzeptiert, wird der neue Wert an die übrigen Member weitergegeben. Falls ein Member fehlschlägt, während es versucht, zum neuen Pfad zu wechseln, schlagen nachfolgende Versuche zum Aktivieren der Datenbank oder zum Herstellen einer Verbindung zur Datenbank fehl (SQL5099N). Alle Member müssen denselben Protokollpfad verwenden.

Anzahl der Gruppencommits (mincommit)
Dieser Parameter ermöglicht Ihnen, das Schreiben von Protokollsätzen auf Platte zu verzögern, bis eine minimale Anzahl von Commitoperationen ausgeführt worden ist. Diese Verzögerung kann dazu beitragen, dass der Systemaufwand für den Datenbankmanager im Zusammenhang mit dem Schreiben von Protokollsätzen verringert und infolgedessen der Durchsatz erhöht wird, wenn mehrere Anwendungen für eine Datenbank ausgeführt werden und viele Commits von den Anwendungen innerhalb kurzer Zeit angefordert werden.

Diese Gruppierung von Commits wird nur ausgeführt, wenn der Wert dieses Parameters größer als eins ist und mehrere Anwendungen gleichzeitig versuchen, für ihre Transaktionen ein Commit durchzuführen. Wenn die Gruppierung von Commits durchgeführt wird, werden Commitanforderungen von Anwendungen solange zurückgehalten, bis entweder eine Sekunde vergangen oder die Anzahl der Commitanforderungen gleich dem Wert dieses Parameters ist.

Anzahl Wiederholungen der Archivierung nach Fehler (numarchretry)
Gibt an, wie oft versucht wird, die Protokolldateien mit einer konfigurierten Protokollarchivierungsmethode zu archivieren, bevor sie in dem vom Konfigurationsparameter **failarchpath** angegebenen Pfad archiviert werden. Dieser Parameter kann nur verwendet werden, wenn der Konfigurationsparameter **failarchpath** gesetzt ist. Der Standardwert ist 5.

Anzahl der aktiven Protokolle, die eine Transaktion umfassen kann (num_log_span)
Dieser Parameter zeigt die Anzahl aktiver Protokolldateien an, die eine aktive Transaktion umfassen kann. Wird der Wert auf 0 gesetzt, gibt es keine Begrenzung für die Anzahl der Protokolldateien, die von einer einzigen Transaktion umfasst werden können.

Verletzt eine Anwendung die Einstellung **num_log_span**, wird die Anwendung gezwungen, die Verbindung zur Datenbank zu beenden.

Dieser Parameter kann zusammen mit dem Konfigurationsparameter **max_log** nützlich sein, wenn ein unbegrenzter Speicherbereich für aktive Protokolle aktiviert ist. Wenn die Endlosprotokollierung aktiv ist (d. h., **logsecond** weist den Wert -1 auf), werden Transaktionen nicht auf die Obergrenze der Anzahl von Protokolldateien (**logprimary** + **logsecond**) beschränkt. Wenn der Wert von **logprimary** erreicht wurde, beginnt der DB2-

Datenbankmanager mit der Archivierung der aktiven Protokolldateien, statt die Transaktion fehlschlagen zu lassen. Dies kann zu Problemen führen, wenn zum Beispiel eine Transaktion mit langer Ausführungszeit ohne Commit zurückgelassen wurde (vielleicht aufgrund einer Anwendung mit einem logischen Fehler). In einen solchen Fall wächst der Speicherbereich für aktive Protokolldateien immer weiter an, was sich negativ auf die Leistung bei einer Recovery nach einem Systemabsturz auswirken könnte. Sie können dies vermeiden, indem Sie Werte für einen der Konfigurationsparameter **max_log** und **num_log_span** (oder beide) angeben.

Anmerkung: Die folgenden DB2-Befehle sind von der Begrenzung ausgenommen, die vom Konfigurationsparameter **num_log_span** festgelegt wird: ARCHIVE LOG, BACKUP DATABASE, LOAD, REORG, RESTORE DATABASE und ROLLFORWARD DATABASE.

Überlaufprotokollpfad (overflowlogpath)

Je nach Ihren Protokollierungsanforderungen kann dieser Parameter für verschiedene Funktionen verwendet werden. Sie können eine Position angeben, an der der DB2-Datenbankmanager nach Protokolldateien suchen soll, die für eine aktualisierende Recovery benötigt werden. Dies weist Ähnlichkeiten mit der Option OVERFLOW LOG PATH des Befehls ROLLFORWARD auf, mit dem Unterschied, dass die Option OVERFLOW LOG PATH mit jedem abgesetzten Befehl ROLLFORWARD angegeben werden muss und Sie diesen Parameter nur einmal angeben müssen. Wenn sowohl der Befehl als auch der Parameter verwendet werden, überschreibt die Option OVERFLOW LOG PATH den Konfigurationsparameter **overflowlogpath** für diese bestimmte ROLLFORWARD-Operation.

Wenn **logsecond** auf -1 gesetzt ist, können Sie ein Verzeichnis angeben, in dem der DB2-Datenbankmanager die aus dem Archiv abgerufenen aktiven Protokolldateien speichern kann. (Aktive Protokolldateien müssen für Rollback-Operationen abgerufen werden, wenn sie sich nicht mehr im Pfad für aktive Protokolldateien befinden).

Wenn für **overflowlogpath** kein Wert angegeben ist, ruft der DB2-Datenbankmanager die Protokolldateien in den Pfad für aktive Protokolldateien ab. Wenn Sie für diesen Parameter einen Wert angeben, werden dadurch zusätzliche Speicherressourcen bereitgestellt, in die der DB2-Datenbankmanager die abgerufenen Protokolldateien stellen kann. Dies bietet den Vorteil, den Ein-/Ausgabeaufwand auf verschiedene Datenträger zu verteilen und mehr Protokolldateien im Pfad für aktive Protokolldateien speichern zu können.

Wenn Sie beispielsweise die Anwendungsprogrammierschnittstelle (API) db2ReadLog zur Replikation verwenden, können Sie mit dem Parameter **overflowlogpath** eine Position angeben, an der der DB2-Datenbankmanager nach Protokolldateien suchen soll, die für diese API benötigt werden. Wenn die Protokolldatei nicht gefunden wird (weder im Pfad für aktive Protokolldateien noch im Überlaufprotokollpfad) und die Datenbank für die Protokollarchivierung konfiguriert ist, ruft der DB2-Datenbankmanager die Protokolldatei ab. Sie können mit diesem Parameter auch ein Verzeichnis angeben, in dem der DB2-Datenbankmanager die abgerufenen Protokolldateien speichern soll. Dadurch wird der Ein-/Ausgabeaufwand für den Pfad für aktive Protokolldateien reduziert, und es können mehr Protokolldateien im Pfad für aktive Protokolldateien gespeichert werden.

Die Einstellung **overflowlogpath** ist nützlich, wenn Endlosprotokollierung konfiguriert ist (d. h., wenn **logsecond** den Wert -1 aufweist). Der DB2-Da-

tenbankmanager kann aktive Protokolldateien speichern, die aus dem Archiv in diesem Pfad abgerufen wurden. (Bei der Endlosprotokollierung müssen aktive Protokolldateien für das Rollback oder Recoverys nach einem Systemabsturz möglicherweise aus dem Archiv abgerufen werden, wenn sie sich nicht mehr im Pfad für aktive Protokolldateien befinden.)

Wenn Sie für den Pfad für aktive Protokolldateien eine Roheinheit konfiguriert haben und **logsecond** auf -1 setzen oder die Anwendungsprogrammierschnittstelle **db2ReadLog** verwenden wollen, muss **overflowlogpath** konfiguriert werden.

Geben Sie eine Zeichenfolge von bis zu 242 Byte an, um den Parameter **overflowlogpath** festzulegen. Die Zeichenfolge muss auf einen Pfadnamen zeigen, bei dem es sich um einen vollständig qualifizierten Pfadnamen handeln muss, nicht um einen relativen Pfad. Der Pfadname muss ein Verzeichnis, keine Roheinheit sein.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken wird die Datenbankpartitionsnummer automatisch an den Pfad angehängt. Dies geschieht, um die Eindeutigkeit des Pfads in mehreren logischen Knotenkonfigurationen zu bewahren.

Anzahl primärer Protokolldateien (logprimary)
Dieser Parameter gibt die Anzahl der primären Protokolle der Größe **logfilsiz** an, die erstellt werden.

Eine primäre Protokolldatei belegt im leeren sowie im vollen Zustand denselben Plattenspeicherplatz. Wenn Sie also mehr Protokolle als erforderlich konfigurieren, wird unnötigerweise Plattenspeicherplatz belegt. Konfigurieren Sie dagegen zu wenig Protokolle, kann der Fall eintreten, dass Ihre Protokolle vollständig mit Daten gefüllt sind. Beim Auswählen der Anzahl der zu konfigurierenden Protokolle müssen Sie daher die für jedes Protokoll vorgesehene Größe einkalkulieren und berücksichtigen, ob Ihre Anwendung mit vollen Protokollen umgehen kann. Die Protokolldateien im Speicherbereich aktiver Protokolldateien können insgesamt maximal 256 GB belegen.

Wenn Sie für eine existierende Datenbank die aktualisierende Recovery aktivieren, müssen Sie die Anzahl der primären Protokolle auf den Wert der Summe der primären und sekundären Protokolle plus eins ändern.

Anzahl sekundärer Protokolle (logsecond)
Dieser Parameter gibt die Anzahl der sekundären Protokolldateien an, die erstellt und bei Bedarf für die Recovery verwendet werden.

Wenn die primären Protokolldateien voll sind, werden die sekundären Protokolldateien in der durch **logfilsiz** Größe definierten Größe bei Bedarf einzeln zugeordnet, und zwar im Höchstfall so viele, wie von diesem Parameter angegeben wird. Wenn der Parameter auf -1 gesetzt ist, wird die Datenbank mit unbegrenztem Speicherbereich für aktive Protokolldateien konfiguriert. Es gibt keine Begrenzung für die Größe oder Anzahl der unvollständigen Transaktionen, die in der Datenbank ausgeführt werden. Eine aktive Endlosprotokollierung ist in Umgebungen nützlich, die umfangreiche Jobs mit größerem Protokollspeicherbedarf verarbeiten müssen, als Sie normalerweise für primäre Protokolle zuordnen würden.

Anmerkung:

1. Die Protokollarchivierung muss aktiviert sein, damit **logsecond** auf -1 gesetzt werden kann.

2. Wenn dieser Parameter auf -1 gesetzt ist, wird der Zeitaufwand für die Recovery nach einem Systemabsturz möglicherweise größer, da der DB2-Datenbankmanager eventuell archivierte Protokolldateien abrufen muss.

Verringern der Protokollieraktivitäten mit dem Parameter NOT LOGGED INITIALLY

Wenn die verwendete Anwendung Arbeitstabellen aus Originaltabellen erstellt und füllt, können Sie die Arbeitstabellen erstellen und in der Anweisung CREATE TABLE den Parameter NOT LOGGED INITIALLY angeben.

Diese Option ist nützlich, wenn die Wiederherstellbarkeit dieser Arbeitstabellen keine große Rolle spielt, da sie ohne großen Aufwand aus den Originaltabellen erneut erstellt werden können. Durch die Angabe des Parameters NOT LOGGED INITIALLY wird die Protokollierungsaktivität reduziert und die Leistung verbessert.

Ein Vorteil in der Verwendung des Parameters NOT LOGGED INITIALLY besteht darin, dass etwaige in der Tabelle durchgeführte Änderungen (wie Einfüge-, Lösch-, Aktualisierungs- oder Indexerstellungsoperationen) innerhalb derselben UOW (Unit of Work), in der die Tabelle erstellt wird, nicht protokolliert werden. Dadurch werden nicht nur die Protokollierungsaktivitäten verringert, sondern es wird auch eine bessere Leistung für Ihre Anwendung erzielt. Sie können das gleiche Ergebnis für vorhandene Tabellen erreichen, indem Sie die Anweisung ALTER TABLE mit dem Parameter NOT LOGGED INITIALLY verwenden.

Anmerkung:

1. Sie können mehrere Tabellen mit dem Parameter NOT LOGGED INITIALLY in derselben UOW erstellen.
2. Änderungen an den Katalogtabellen und anderen Benutzertabellen werden weiterhin protokolliert.

Da die Änderungen der Tabelle nicht protokolliert werden, sollten Sie bei der Entscheidung, das Tabellenattribut NOT LOGGED INITIALLY zu verwenden, Folgendes beachten:

- *Alle* Änderungen an der Tabelle werden zum Commitzeitpunkt auf Platte geschrieben. Das bedeutet, dass die Commitoperation länger dauern kann.
- Wenn das Attribut NOT LOGGED INITIALLY aktiviert ist und eine nicht protokollierte Aktivität ausgeführt wird, wird beim Fehlschlagen einer Anweisung entweder die gesamte UOW rückgängig gemacht oder eine Operation ROLLBACK TO SAVEPOINT ausgeführt (SQL1476N).
- Wenn Sie HADR (High Availability Disaster Recovery) verwenden, sollten Sie das Tabellenattribut NOT LOGGED INITIALLY nicht verwenden. Tabellen, die in der Primärdatenbank mit der Option NOT LOGGED INITIALLY erstellt werden, werden nicht für die Bereitschaftsdatenbank repliziert. Zugriffsversuche auf solche Tabellen in einer aktiven Bereitschaftsdatenbank - oder nachdem die Bereitschaftsdatenbank nach einer Übernahmeoperation zur Primärdatenbank wurde - führen zur Rückgabe eines Fehlers SQL1477N.
- Diese Tabellen können Sie bei einer aktualisierenden Recovery nicht wiederherstellen. Wenn die aktualisierende Recovery auf eine Tabelle stößt, die mit der Option NOT LOGGED INITIALLY erstellt wurde, wird diese Tabelle als nicht verfügbar markiert. Nach der Recovery der Datenbank führt jeder Versuch, auf die Tabelle zuzugreifen, zur Rückgabe der Nachricht SQL1477N.

Anmerkung: Bei der Erstellung einer Tabelle werden Zeilensperren für die Katalogtabellen aktiviert, bis ein Commit durchgeführt wird. Die Inaktivierung der Protokollfunktion ist nur dann von Vorteil, wenn Sie die Tabelle in derselben UOW mit Werten füllen, in der sie erstellt wird. Daraus ergeben sich Konsequenzen für den gemeinsamen Zugriff.

Verringern der Protokollieraktivitäten mit deklarierten temporären Tabellen

Wenn Sie planen, deklarierte temporäre Tabellen als Arbeitstabellen einzusetzen, ist Folgendes zu beachten:

- Deklarierte temporäre Tabellen werden nicht in den Katalogen erstellt. Daher werden keine Sperren aktiviert.
- Für deklarierte temporäre Tabellen findet keine Protokollierung statt, auch nicht nach der ersten Commitoperation.
- Geben Sie die Option ON COMMIT PRESERVE an, um die Zeilen in der Tabelle nach einer Commitoperation zu behalten. Ansonsten werden alle Zeilen gelöscht.
- Nur die Anwendung, die die deklarierte temporäre Tabelle erstellt, kann auf diese Instanz der Tabelle zugreifen.
- Die Tabelle wird implizit gelöscht, wenn die Verbindung der Anwendung zur Datenbank beendet wird.
- Erstellte temporäre Tabellen (CGTTs) und deklarierte temporäre Tabellen (DGTTs) können nicht in einer aktiven Bereitschaftsdatenbank erstellt werden und auch ein Zugriff auf diese Tabellen in einer Bereitschaftsdatenbank ist nicht möglich.
- Betriebsfehler in einer UOW mit einer deklarierten temporären Tabelle bewirken nicht, dass die UOW vollständig rückgängig gemacht wird. Ein Fehler bei der Ausführung einer Anweisung, die den Inhalt einer deklarierten temporären Tabelle ändert, führt jedoch dazu, dass alle Zeilen in dieser Tabelle gelöscht werden. Eine Rollback-Operation der UOW (bzw. eines Sicherungspunkts) löscht alle Zeilen in deklarierten temporären Tabellen, die innerhalb dieser UOW (bzw. dieses Sicherungspunkts) geändert wurden.

Blockieren von Transaktionen bei vollem Protokollverzeichnis

Wenn der DB2-Datenbankmanager im Pfad für aktive Protokolldateien keine Protokolldatei erstellen kann, weil für diese nicht genügend Speicherplatz verfügbar ist, erhalten Sie Fehlerrückmeldungen, dass der Datenträger voll ist.

Wenn Sie dagegen den Datenbankkonfigurationsparameter **blk_log_dsk_ful** einrichten, unternimmt der DB2-Datenbankmanager wiederholte Versuche, die Protokolldatei zu erstellen, bis er schließlich erfolgreich ist, anstatt den Fehler „Datenträger voll" zurückzugeben.

Wenn Sie den Datenbankkonfigurationsparameter **blk_log_dsk_ful** einrichten, versucht der DB2-Datenbankmanager alle 5 Minuten, die Protokolldatei zu erstellen, bis er erfolgreich ist. Wenn eine Protokollarchivierungsmethode angegeben wurde, prüft der DB2-Datenbankmanager außerdem den Abschluss der Protokolldateiarchivierung. Wenn eine archivierte Protokolldatei erfolgreich archiviert wurde, kann der DB2-Datenbankmanager die inaktive Protokolldatei in den neuen Protokolldateinamen umbenennen und fortfahren. Nach jedem Versuch schreibt der DB2-Datenbankmanager eine Nachricht in das Protokoll mit den Benachrichtigungen für die Systemverwaltung. Sie können nur durch Überwachen des Protokolls mit den

Benachrichtigungen für die Systemverwaltung feststellen, ob Ihre Anwendung nur deshalb blockiert ist, weil für Protokolldateien kein Plattenplatz mehr vorhanden ist.

Bis zur erfolgreichen Erstellung der Protokolldatei kann keine Benutzeranwendung, die Tabellendaten aktualisieren will, eine Transaktion festschreiben. Auf Lesezugriff beschränkte Abfragen sind u. U. nicht direkt betroffen. Wenn jedoch eine Abfrage auf Daten zugreifen muss, die aufgrund einer Aktualisierungsanforderung gesperrt sind, oder auf eine Datenseite, die im Pufferpool von der aktualisierenden Anwendung korrigiert wird, erscheinen auch auf Lesezugriff beschränkte Abfragen blockiert.

Protokolldateiverwaltung durch Protokollarchivierung

Die Protokolldateiarchivierung des DB2-Servers gestaltet sich durch verschiedene Dateiverwaltungs- und Terminierungsprobleme der Betriebssysteme kompliziert. Beispiel: Wenn auf einer Platte ein Fehler auftritt, während der DB2-Datenbankmanager eine Warteschlange von Protokolldateien archiviert, können diese Protokolldateien (und die darin enthaltenen Transaktionsdaten) verloren gehen.

Ein ordnungsgemäßes Konfigurieren der Datenbankprotokollierung kann verhindern, dass derartige Probleme Ihre Verfügbarkeits- und Recoverystrategie untergraben.

Die folgenden allgemeinen Aspekte gelten für alle Methoden der Protokollarchivierung:

- Der Datenbankkonfigurationsparameter **logarchcompr1** gibt an, ob der Datenbankmanager die Protokolldateien in der von **logarchmeth1** angegebenen Position komprimiert. Wenn der Konfigurationsparameter **logarchmeth1** auf einen anderen Wert als DISK, TSM oder VENDOR gesetzt ist, bleibt die Komprimierung der Protokollarchive unabhängig von der Einstellung des Konfigurationsparameters **logarchcompr1** wirkungslos.
- Der Datenbankkonfigurationsparameter **logarchcompr2** gibt an, ob der Datenbankmanager die Protokolldateien in der von **logarchmeth2** angegebenen Position komprimiert. Wenn der Konfigurationsparameter **logarchmeth2** auf einen anderen Wert als DISK, TSM oder VENDOR gesetzt ist, bleibt die Komprimierung der Protokollarchive unabhängig von der Einstellung des Konfigurationsparameters **logarchcompr2** wirkungslos.
- Der Datenbankkonfigurationsparameter **logarchmeth1** veranlasst den Datenbankmanager zum Archivieren von Protokolldateien oder zum Abrufen der Protokolldateien während einer aktualisierenden Recovery von Datenbanken mithilfe der angegebenen Methode. Es erfolgt eine Anforderung zum Abrufen einer Protokolldatei, wenn das Dienstprogramm zur aktualisierenden Recovery eine Protokolldatei benötigt, die nicht im Verzeichnis für den Protokollpfad zu finden ist. Protokolldateien werden aus dem Pfad archiviert, der im Konfigurationsparameter **logpath** angegeben wird.

  Der Datenbankkonfigurationsparameter **logarchmeth2** veranlasst den Datenbankmanager zum Archivieren zusätzlicher Kopien der Protokolldateien. Wenn Sie die Protokollspiegelung konfigurieren, dann werden die Protokolldateien, die in dem im Parameter **logarchmeth2** angegebenen Pfad archiviert werden, dem Pfad für die Protokollspiegelung entnommen. Wenn Sie die Protokollspiegelung nicht konfigurieren, dann werden die Protokolldateien, die in dem im Parameter **logarchmeth2** angegebenen Pfad archiviert werden, dem aktuellen Protokollpfad entnommen.

- Sie sollten keine lokal angeschlossenen Bandlaufwerke verwenden, um Protokolldateien zu speichern, wenn Sie mit einer der folgenden Funktionen arbeiten:
  - Endlosprotokollierung
  - Online-Recovery auf Tabellenbereichsebene
  - Replikation
  - API zum asynchronen Lesen von Protokolldaten (db2ReadLog)
  - High Availability Disaster Recovery (HADR)

  Bei jedem dieser Verfahren kann eine Protokolldatei abgerufen werden, was mit den Protokollarchivierungsoperationen kollidieren kann. Ebenso können Sie lokal angeschlossene Bandlaufwerke nicht in einer DB2 pureScale-Umgebung verwenden, da das Member, das die Operation für die Protokollzusammenführung ausführt, Protokolle für die anderen Member abrufen muss.
- Wenn Sie mit der Protokollarchivierung arbeiten, versucht der Protokollmanager aktive Protokolle zu archivieren, wenn sie gefüllt sind. Wenn eine Datenbank inaktiviert wird, bevor der Protokollmanager die Archivierung als erfolgreich aufzeichnen kann, versucht der Protokollmanager in bestimmten Fällen, das Protokoll erneut zu archivieren, wenn die Datenbank aktiviert wird. Auf diese Weise kann es vorkommen, dass eine Protokolldatei mehrmals archiviert wird.
- Beim Archivieren wird eine Protokolldatei an den Protokollmanager übergeben, wenn sie voll ist, auch wenn die Protokolldatei noch aktiv ist und für die normale Verarbeitung benötigt wird. Dadurch können Kopien der Daten so schnell wie möglich aus den flüchtigen Speichern entfernt werden. Die an den Protokollmanager übergebene Protokolldatei wird so lange im Verzeichnis für den Protokollpfad gespeichert, bis sie für die normale Verarbeitung nicht mehr benötigt wird. Dann wird der Plattenspeicherplatz wiederverwendet.
- Wenn eine Protokolldatei archiviert wurde und keine offenen Transaktion enthält, löscht der DB2-Datenbankmanager die Datei nicht, sondern benennt sie in die nächste Protokolldatei um, wenn eine solche Datei benötigt wird. Diese Vorgehensweise führt zu einer Leistungsverbesserung, da die Erstellung einer neuen Protokolldatei (anstatt der Umbenennung) erfordern würde, dass alle Seiten herausgeschrieben werden müssten, um sicherzustellen, dass der erforderliche Plattenspeicherplatz oder entsprechender Speicherplatz auf anderen Einheiten verfügbar ist.
- Während der Recovery nach einem Systemabsturz, der Recovery nach dem Absturz eines Members (in einer DB2 pureScale-Umgebung) oder während eines Rollbacks während der Laufzeit ruft der DB2-Datenbankmanager keine Protokolldateien ab, es sei denn, Sie haben den Datenbankkonfigurationsparameter **logsecond** auf den Wert -1 gesetzt (und somit die Endlosprotokollierung aktiviert). In einer DB2 pureScale-Umgebung muss der Datenbankmanager möglicherweise während der Recovery nach Absturz einer Gruppe archivierte Protokolle abrufen, wenn die Endlosprotokollierung nicht aktiviert wurde.
- Das Konfigurieren der Protokollarchivierung garantiert nicht die aktualisierende Recovery bis zum Fehlerpunkt, sondern ist lediglich ein Versuch, das Fehlerfenster zu verkleinern. Während des Füllens von Protokolldateien mit Daten werden die Protokolle vom Protokollmanager asynchron archiviert. Sollte bei der Platte, auf der das Protokoll gespeichert wird, ein Fehler auftreten, bevor eine Protokolldatei voll ist, gehen die Daten in der betreffenden Protokolldatei verloren. Da die Dateien für die Archivierung in eine Warteschlange gestellt werden, kann außerdem ein Fehler auf der Platte auftreten, bevor alle Dateien kopiert werden konnten. Dadurch gehen alle Protokolldateien in der Warteschlange verloren.

  Damit Protokolldateien nicht permanent verloren gehen, wenn die Platte oder die Einheit ausfällt, auf der sich der Protokollpfad befindet, können Sie den Datenbankkonfigurationsparameter **mirrorlogpath** verwenden. Damit wird sicher-

gestellt, dass die Protokolle in einen sekundären Pfad geschrieben werden. Solange der sekundäre Pfad nicht ebenso wie die primäre Platte bzw. Einheit ausfällt, stehen die Protokolldateien für die Recovery zur Verfügung.

Wenn Sie sowohl den Konfigurationsparameter **mirrorlogpath** als auch den Konfigurationsparameter **logarchmeth2** angeben, dann archiviert der Konfigurationsparameter **logarchmeth2** die Protokolldateien aus dem Pfad für die Protokollspiegelung, anstatt zusätzliche Kopien der Protokolldateien im aktuellen Protokollpfad zu archivieren. Sie können dieses Protokollarchivierungsverhalten verwenden, um die Ausfallsicherheit während der aktualisierenden Recovery zu verbessern. Die Ursache ist, dass eine verwendbare archivierte Protokolldatei aus dem Pfad für die Protokollspiegelung möglicherweise weiterhin verfügbar ist, um eine Datenbankrecoveryoperation fortzusetzen, auch wenn eine primäre Protokolldatei aus dem aktuellen Protokollpfad aufgrund eines Plattenfehlers vor der Archivierung beschädigt wurde.

- Die konfigurierte Größe der einzelnen Protokolldateien hat eine direkte Auswirkung auf die Protokollarchivierung. Wenn die einzelnen Protokolldateien sehr groß sind, kann eine große Menge von Daten verloren gehen, wenn ein Fehler auf der Platte auftritt. Wenn Sie Ihre Datenbank zur Verwendung kleiner Protokolldateien konfigurieren, führt der Protokollmanager Protokollarchivierungen häufiger aus.

  Wenn die Daten allerdings auf eine langsamere Einheit wie ein Band übertragen werden, ist es oft sinnvoller, größere Protokolldateien zu konfigurieren, um eine zu große Warteschlange zu vermeiden. Sie sollten ebenfalls größere Protokolldateien verwenden, wenn durch das Archivieren der einzelnen Dateien jeweils zusätzlicher Systemaufwand entsteht, z. B. Zurückspulen der Bandeinheit oder Herstellen einer Verbindung zum Archivierungsmedium.
- Wenn Sie mit der Protokollarchivierung arbeiten, versucht der Protokollmanager primäre Protokolle zu archivieren, wenn sie gefüllt sind. In einigen Fällen archiviert der Protokollmanager ein Protokoll, bevor es voll ist. Dieser Fall tritt ein, wenn die Protokolldatei aufgrund der Inaktivierung der Datenbank, durch Absetzen des Befehls **ARCHIVE LOG**, am Ende eines Online-Backups oder durch Absetzen des Befehls **SET WRITE SUSPEND** abgeschnitten wird.

  Anmerkung: Zur Freigabe nicht genutzten Protokollspeichers wird die Protokolldatei abgeschnitten, bevor sie archiviert wird.
- Wenn Sie Protokolle und Backup-Images auf einem Bandlaufwerk archivieren, müssen Sie sicherstellen, dass als Ziel für die Backup-Images und die archivierten Protokolle nicht dasselbe Bandlaufwerk verwendet wird. Da während einer Backup-Operation eine Protokollarchivierung stattfinden kann, kann es zu einem Fehler kommen, wenn beide Prozesse gleichzeitig versuchen, auf dasselbe Bandlaufwerk zu schreiben.

Folgende Hinweise und Informationen sind für das Aufrufen eines Benutzerexitprogramms bzw. eines von einem anderen Anbieter gelieferten Programms zum Archivieren und Abrufen von Protokolldateien zu beachten:

- Der DB2-Datenbankmanager öffnet eine Protokolldatei im Lesemodus, wenn er ein Benutzerexitprogramm zum Archivieren der Datei startet. Unter bestimmten Betriebssystemen kann deshalb das Benutzerexitprogramm die Protokolldatei nicht löschen. Andere Betriebssysteme wie AIX ermöglichen es Prozessen, einschließlich des Benutzerexitprogramms, Protokolldateien zu löschen. Ein Benutzerexitprogramm sollte eine Protokolldatei nie nach ihrer Archivierung löschen, da die Datei weiterhin aktiv und für die Recovery nach einem Systemabsturz er-

forderlich sein könnte. Der DB2-Datenbankmanager verwaltet die Wiederverwendung des Speicherplatzes auf Datenträgern, wenn Protokolldateien archiviert werden.

- Ein Benutzerexitprogramm bzw. ein Programm eines anderen Anbieters kann eine Anforderung zum Archivieren einer nicht existierenden Datei empfangen, weil mehrere Anforderungen zum Archivieren vorhanden waren und die Datei nach der ersten erfolgreichen Archivierungsoperation gelöscht wurde. Ein Benutzerexitprogramm bzw. ein Programm eines anderen Anbieters kann auch eine Anforderung zum Abrufen einer nicht existierenden Datei empfangen, weil sie sich in einem anderen Verzeichnis befindet oder weil das Ende der Protokolle erreicht ist. In beiden Fällen sollte das Benutzerexitprogramm oder das Programm eines anderen Anbieters diese Anforderung ignorieren und einen Rückkehrcode für eine erfolgreiche Ausführung übergeben.
- Unter Windows-Betriebssystemen können Sie keinen REXX-Benutzerexit zum Archivieren von Protokollen verwenden.
- Das Benutzerexitprogramm bzw. das Programm eines anderen Anbieters sollte das Vorhandensein verschiedener gleichnamiger Protokolldateien nach einer punktuellen Recovery zulassen. Das Benutzerexitprogramm bzw. das Programm eines anderen Anbieters sollte so geschrieben sein, dass beide Protokolldateien gespeichert bleiben und dass diese Protokolldateien dem korrekten Recoverypfad zugeordnet werden.
- Wenn ein Benutzerexitprogramm bzw. ein Programm eines anderen Anbieters für zwei oder mehr Datenbanken aktiviert wird, die zur Archivierung von Protokolldateien dieselbe Bandeinheit verwenden, und für eine der Datenbanken eine aktualisierende Recovery ausgeführt wird, sollten keine anderen Datenbanken aktiv sein. Wenn eine andere Datenbank während der Operation zur aktualisierenden Recovery versucht, eine Protokolldatei zu archivieren, ist es möglich, dass eine der folgenden Situationen eintritt:
  - Die Protokolle, die für die aktualisierende Recovery erforderlich sind, können möglicherweise nicht gefunden werden.
  - Die neue Protokolldatei, die auf der Bandeinheit archiviert wird, überschreibt möglicherweise die Protokolldateien, die zuvor auf dieser Bandeinheit gespeichert wurden.

  Um das Auftreten dieser Situationen zu verhindern, können Sie einen der folgenden Schritte ausführen:
  - Sie können sicherstellen, dass keine anderen Datenbanken auf der Datenbankpartition, die das Benutzerexitprogramm aufruft, während der aktualisierenden Recovery geöffnet sind.
  - Sie können ein Benutzerexitprogramm schreiben, um diese Situation zu beheben.

Konfigurieren einer Clusterumgebung für hohe Verfügbarkeit

Eine Strategie zum Entwerfen einer hoch verfügbaren Lösung besteht im Erstellen eines Clusters von Maschinen und in der Verwendung von Cluster-Management-Software, um die Auslastung auf diesen Maschinen gleichmäßig zu verteilen.

Falls Sie den IBM DB2-Server auf einer oder auf mehreren Maschinen eines Clusters installieren, müssen Sie den Cluster-Manager so konfigurieren, dass er ordnungsgemäß auf Fehler oder Störungen reagiert, die die Datenbank(en) beeinträchtigen. Sie müssen außerdem die Datenbankmanagerinstanzen so konfigurieren, dass sie in einer Clusterumgebung ordnungsgemäß funktionieren.

Informationen zu diesem Vorgang

Die manuelle Konfiguration und Verwaltung der Datenbankinstanzen und des Cluster-Managers ist komplex, zeitintensiv und fehleranfällig. DB2 High Availability Feature stellt eine Infrastruktur bereit, die es dem Datenbankmanager ermöglicht, mit dem Cluster-Manager zu kommunizieren, wenn Konfigurationsänderungen bei Instanzen (wie beispielsweise das Stoppen einer Datenbankmanagerinstanz) Clusteränderungen erforderlich machen.

Vorgehensweise

1. Installieren Sie die Cluster-Management-Software.

   SA MP ist unter den Betriebssystemen DB2 Enterprise Server Edition, DB2 Advanced Enterprise Server Edition, DB2 Workgroup Server Edition, DB2 Connect Enterprise Edition und DB2 Connect Application Server Edition on AIX, Linux and Solaris SPARC integriert. Darüber hinaus besteht eine Integration mit DB2 Express-C Fixed Term License (FTL) und IBM DB2 High Availability Feature for Express Edition auf Linux-Betriebssystemen. Unter Windows-Betriebssystemen ist SA MP Teil des jeweiligen Produktpakets für diese DB2-Datenbankprodukte und -komponenten, es ist jedoch nicht in das DB2-Installationsprogramm integriert.
2. Konfigurieren Sie DB2-Datenbankmanagerinstanzen für den Cluster-Manager, und konfigurieren Sie den Cluster-Manager für den DB2-Server.

   DB2 High Availability Instance Configuration Utility (db2haicu) ist ein textbasiertes Dienstprogramm zur Konfiguration und Verwaltung hoch verfügbarer Datenbanken in einer Clusterumgebung.
3. Wenn sich die Anforderungen an die Datenbank im Laufe der Zeit ändern und die Datenbankkonfiguration in der Clusterumgebung modifiziert werden muss, fahren Sie damit fort, die Konfiguration der Datenbankmanagerinstanzen und die Konfiguration des Cluster-Managers zu synchronisieren.

   DB2 High Availability Feature stellt eine Infrastruktur bereit, die es dem Datenbankmanager ermöglicht, mit dem Cluster-Manager zu kommunizieren, wenn Konfigurationsänderungen bei Instanzen (wie beispielsweise das Stoppen einer Datenbankmanagerinstanz) Clusteränderungen erforderlich machen.

   Es spielt keine Rolle, ob Sie **db2haicu** mit SA MP oder einen weiteren Cluster-Manager verwenden, der die API des DB2-Cluster-Managers unterstützt - die Verwaltung der Clusterumgebung mit DB2 High Availability Feature ist einfacher als eine separate Verwaltung von Datenbankmanager- und Clusterkonfiguration.

Cluster-Manager-Integration mit DB2 High Availability Feature

DB2 High Availability Feature ermöglicht die Integration zwischen dem IBM DB2-Server und der Cluster-Management-Software.

Wenn Sie eine Datenbankmanagerinstanz in einer Clusterumgebung stoppen, müssen Sie den Cluster-Manager darüber informieren, dass die Instanz gestoppt wurde. Wenn der Cluster-Manager über das Stoppen der Instanz nicht informiert wird, versucht er möglicherweise, eine Operation wie beispielsweise eine Funktionsübernahme für die gestoppte Instanz auszuführen. DB2 High Availability Feature stellt eine Infrastruktur bereit, die es dem Datenbankmanager ermöglicht, mit dem Cluster-Manager zu kommunizieren, wenn Konfigurationsänderungen bei Instanzen (wie beispielsweise das Stoppen einer Datenbankmanagerinstanz) Clusteränderungen erforderlich machen.

DB2 High Availability Feature besteht aus den folgenden Elementen:

- IBM Tivoli System Automation for Multiplatforms (SA MP) ist als Bestandteil von DB2 High Availability Feature mit dem DB2-Server unter AIX und Linux gebündelt und in das DB2-Installationsprogramm integriert. Sie können für SA MP mithilfe des DB2-Installationsprogramms oder der Scripts **installSAM** und **uninstallSAM**, die sich auf den DB2-Server-Installationsmedien befinden, eine Installation, ein Upgrade oder eine Deinstallation durchführen.
- In einer Clusterumgebung erfordern bestimmte Konfigurations- und Verwaltungsoperationen der Datenbankmanagerinstanz entsprechende Änderungen an der Clusterkonfiguration. Mit DB2 High Availability Feature kann der Datenbankmanager automatisch Änderungen an der Konfiguration des Cluster-Managers anfordern, wenn bestimmte Konfigurations- und Verwaltungsoperationen der Datenbankmanagerinstanz ausgeführt werden. Siehe: „Automatisches Konfigurieren eines Clusters mit DB2 High Availability Feature"
- DB2 High Availability Instance Configuration Utility (db2haicu) ist ein textbasiertes Dienstprogramm zur Konfiguration und Verwaltung hoch verfügbarer Datenbanken in einer Clusterumgebung. Siehe: „DB2 High Availability Instance Configuration Utility (db2haicu)" auf Seite 110

Basiskomponente von IBM Tivoli System Automation for Multiplatforms (SA MP)

IBM Tivoli System Automation for Multiplatforms (SA MP) stellt HADR-Funktionalität für AIX, Linux, Solaris SPARC und Windows bereit.

SA MP ist unter AIX-, Linux-, und Solaris SPARC-Betriebssystemen in DB2 Enterprise Server Edition, DB2 Advanced Enterprise Server Edition, DB2 Workgroup Server Edition, DB2 Connect Enterprise Edition und DB2 Connect Application Server Edition integriert. Darüber hinaus besteht eine Integration mit Express Edition zur Verwendung mit der FTL (Fixed Term License, Lizenz mit fester Laufzeit) für DB2 Express-C sowie mit DB2 High Availability Feature.

Auf Windows-Betriebssystemen ist SA MP im Paket für alle diese DB2-Datenbankprodukte und -features enthalten, jedoch nicht in das Installationsprogramm für DB2-Datenbankprodukt integriert.

Sie können diese Kopie von SA MP dazu verwenden, die Hochverfügbarkeit Ihres DB2-Datenbanksystems zu verwalten. Mit dieser Kopie können ausschließlich DB2-Datenbanksysteme verwaltet werden, es sei denn, es wird ein Upgrade für die SA MP-Lizenz erworben.

SA MP ist der Standard-Cluster-Manager in einer IBM DB2 Server-Clusterumgebung unter AIX-, Linux- und Solaris SPARC-Betriebssystemen.

Weitere Informationen zu SA MP finden Sie in IBM Tivoli System Automation for Multiplatforms (SA MP)publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms3.1.html. Die Liste der unterstützten Betriebssysteme ist auch auf der folgenden Website verfügbar: www.ibm.com/software/tivoli/products/sys-auto-linux/platforms.html.

Automatisches Konfigurieren eines Clusters mit DB2 High Availability Feature

In einer Clusterumgebung erfordern bestimmte Konfigurations- und Verwaltungsoperationen der Datenbankmanagerinstanz entsprechende Änderungen an der Clusterkonfiguration. Mit DB2 High Availability Feature kann der Datenbankmana-

ger automatisch Änderungen an der Konfiguration des Cluster-Managers anfordern, wenn bestimmte Konfigurations- und Verwaltungsoperationen der Datenbankmanagerinstanz ausgeführt werden.

Vorbereitende Schritte

Damit der Datenbankmanager die erforderliche Clusterkonfiguration für Datenbankverwaltungstasks vornehmen kann, müssen Sie die Instanz für hohe Verfügbarkeit konfigurieren, indem Sie mit **db2haicu** eine *Clusterdomäne* für die Instanz erstellen. Weitere Informationen hierzu finden Sie in „Konfigurieren einer Clusterumgebung mit DB2 High Availability Instance Configuration Utility (db2haicu)" auf Seite 102.

Vorgehensweise

Bei Ausführung der folgenden Konfigurations- und Verwaltungsoperationen der Datenbankmanagerinstanz nimmt der Datenbankmanager automatisch die entsprechende Cluster-Manager-Konfiguration für Sie vor:

- Starten einer Datenbank mit **START DATABASE** oder **db2start**.
- Stoppen einer Datenbank mit **STOP DATABASE** oder **db2stop**.
- Erstellen einer Datenbank mit **CREATE DATABASE**.
- Hinzufügen von Speicher mit **CREATE TABLESPACE**.
- Entfernen von Speicher mit **ALTER TABLESPACE DROP** oder **DROP TABLESPACE**.
- Hinzufügen oder Entfernen von Speicherpfaden mit ALTER DATABASE.
- Löschen einer Datenbank mit **DROP TABLESPACE**.
- Wiederherstellen einer Datenbank mit **RESTORE DATABASE** oder db2Restore.
- Angeben der Tabellenbereichscontainer für einen umgeleiteten Restore mit **SET TABLESPACE CONTAINERS**.
- Aktualisierendes Wiederherstellen einer Datenbank mit **ROLLFORWARD DATABASE** oder db2Rollforward.
- Recovery einer Datenbank mit **RECOVER DATABASE** oder db2Recover.
- Erstellen von Ereignismonitoren mit **CREATE EVENT MONITOR**.
- Löschen von Ereignismonitoren mit **DROP EVENT MONITOR**.
- Erstellen und Ändern von externen Routinen unter Verwendung von:
  - CREATE PROCEDURE
  - CREATE FUNCTION
  - CREATE FUNCTION
  - CREATE METHOD
  - ALTER PROCEDURE
  - ALTER FUNCTION
  - ALTER METHOD
- Löschen von externen Routinen unter Verwendung von:
  - DROP PROCEDURE
  - DROP FUNCTION
  - DROP METHOD
- Starten von DB2 HADR-Operationen (High Availability Disaster Recovery) für eine Datenbank mit **START HADR**.
- Stoppen von HADR-Operationen für eine Datenbank mit **STOP HADR**.

- Veranlassen, dass eine HADR-Bereitschaftsdatenbank die Funktion einer HADR-Primärdatenbank übernimmt (mit **TAKEOVER HADR**).
- Festlegen des Konfigurationsparameters des Datenbankmanagers **diagpath** oder **spm_log_path**.
- Festlegen des Datenbankkonfigurationsparameters **newlogpath**, **overflowlogpath**, **mirrorlogpath**, oder **failarchpath**.
- Löschen einer Datenbankmanagerinstanz mit **db2idrop**.

Ergebnisse

Wenn der Datenbankmanager die Änderungen an der Clusterkonfiguration für die aufgelisteten Datenbankverwaltungstasks koordiniert, müssen Sie keine separaten Cluster-Manager-Operationen ausführen.

Konfigurieren einer Clusterumgebung mit DB2 High Availability Instance Configuration Utility (db2haicu)

Mit DB2 High Availability Instance Configuration Utility (db2haicu) können Sie die Datenbanken in einer Clusterumgebung konfigurieren und verwalten. Wenn Sie **db2haicu** die Konfigurationsdetails für eine Datenbankmanagerinstanz mitteilen, leitet **db2haicu** die erforderlichen Details der Clusterkonfiguration an die Cluster-Management-Software weiter.

Vorbereitende Schritte

- Vor der Verwendung von DB2 High Availability Instance Configuration Utility (db2haicu) müssen bestimmte Tasks ausgeführt werden. Weitere Informationen hierzu finden Sie in „DB2 High Availability Instance Configuration Utility (db2haicu) - Voraussetzungen" auf Seite 143.

Informationen zu diesem Vorgang

Sie können **db2haicu** im interaktiven Modus ausführen oder eine XML-Eingabedatei verwenden:

Interaktiver Modus
Wenn Sie DB2 High Availability Instance Configuration Utility (db2haicu) mit dem Befehl **db2haicu** aufrufen und keine XML-Eingabedatei mit dem Parameter **-f** angeben, wird das Dienstprogramm im interaktiven Modus ausgeführt. Im interaktiven Modus von **db2haicu** werden Informationen in einem textbasierten Format angezeigt und abgefragt. Weitere Informationen hierzu finden Sie in „Ausführen von DB2 High Availability Instance Configuration Utility (db2haicu) im interaktiven Modus" auf Seite 113.

Stapelmodus mit einer XML-Eingabedatei
Sie können den Parameter `-f` *`name-der-eingabedatei`* mit dem Befehl **db2haicu** verwenden, um DB2 High Availability Instance Configuration Utility (db2haicu) mit einer XML-Eingabedatei auszuführen, in der die Konfigurationsdetails angegeben sind. Die Ausführung von **db2haicu** mit einer XML-Eingabedatei ist nützlich, wenn Sie bestimmte Konfigurationstasks mehrmals ausführen müssen, z. B. bei der Konfiguration mehrerer Datenbankpartitionen für hohe Verfügbarkeit. Weitere Informationen hierzu finden Sie in „Ausführen von DB2 High Availability Instance Configuration Utility (db2haicu) mit einer XML-Eingabedatei" auf Seite 114.

Ein detailliertes Szenario mit beiden Methoden für die Einrichtung eines HADR-Paares unter Verwendung von **db2haicu** enthält das Dokument „Automated Clus-

ter Controlled HADR (High Availability Disaster Recovery) Configuration Setup using the IBM DB2 High Availability Instance Configuration Utility (db2haicu)".

Einschränkungen

Für die Verwendung von DB2 High Availability Instance Configuration Utility (db2haicu) gelten bestimmte Einschränkungen. Weitere Informationen hierzu finden Sie in „DB2 High Availability Instance Configuration Utility (db2haicu) - Einschränkungen" auf Seite 147.

Vorgehensweise

Führen Sie für die einzelnen Datenbankmanagerinstanzen die folgenden Schritte aus:

1. Erstellen Sie eine neue Clusterdomäne.

   Wenn Sie DB2 High Availability Instance Configuration Utility (db2haicu) zum ersten Mal für eine Datenbankmanagerinstanz ausführen, erstellt **db2haicu** ein Modell des Clusters, das als *Clusterdomäne* bezeichnet wird. Weitere Informationen hierzu finden Sie in „Erstellen einer Clusterdomäne mit DB2 High Availability Instance Configuration Utility (db2haicu)" auf Seite 145.
2. Fahren Sie fort, die Konfiguration der Clusterdomäne weiter einzugrenzen, und führen Sie Verwaltungs- und Wartungsaufgaben für die Clusterdomäne aus.

   Wenn Sie das Clusterdomänenmodell der Clusterumgebung mit **db2haicu** modifizieren, gibt der Datenbankmanager die entsprechenden Änderungen an die Datenbankmanagerinstanz und die Clusterkonfiguration weiter. Weitere Informationen hierzu finden Sie in „Verwalten einer Clusterdomäne mit DB2 High Availability Instance Configuration Utility (db2haicu)" auf Seite 145.

Nächste Schritte

DB2 High Availability Instance Configuration Utility (db2haicu) verfügt über kein separates Diagnoseprotokoll. Fehler bei **db2haicu** können mit dem Diagnoseprotokoll des Datenbankmanagers, der Protokolldatei **db2diag.log** und dem Tool **db2pd** untersucht und diagnostiziert werden. Weitere Informationen hierzu finden Sie in „Fehlerbehebung bei DB2 High Availability Instance Configuration Utility (db2haicu)" auf Seite 147.

Clusterdomäne

Eine Clusterdomäne ist ein Modell, das Informationen zu Clusterelementen wie Datenbanken, Mountpunkten und Richtlinien für Funktionsübernahme enthält. Clusterdomänen werden mit DB2 High Availability Instance Configuration Utility (db2haicu) erstellt.

db2haicu (DB2 High Availability Instance Configuration Utility) verwendet die Informationen in der Clusterdomäne, um Konfigurations- und Verwaltungstasks für die Clusterverwaltung zu ermöglichen. Darüber hinaus verwendet der Datenbankmanager als Bestandteil von DB2 High Availability Feature die Informationen in der Clusterdomäne, um automatisierte Clusterverwaltungstasks auszuführen.

Wenn der Clusterdomäne ein Clusterelement hinzugefügt wird, dann wird dieses Element bei allen nachfolgenden **db2haicu**-Konfigurationsoperationen bzw. automatischen Operationen für die Clusterverwaltung berücksichtigt, die als Bestandteil von DB2 High Availability Feature vom Datenbankmanager ausgeführt werden. Wenn Sie ein Clusterelement aus der Clusterdomäne entfernen, wird dieses Element bei **db2haicu**-Operationen bzw. automatischen Operationen für die Cluster-

verwaltung des Datenbankmanagers nicht mehr berücksichtigt. **db2haicu** und der Datenbankmanager können nur Operationen für Clusterelemente mit dem Clustermanager koordinieren, die in der von Ihnen mit **db2haicu** erstellten Clusterdomäne enthalten sind.

Sie können mit **db2haicu** folgende Clusterdomänenelemente erstellen und konfigurieren:

- Computer oder Maschinen (im Zusammenhang mit Clusterdomänen *Clusterdomänenknoten* genannt)
- Netzschnittstellenkarten (in **db2haicu** *Netzschnittstellen*, *Schnittstellen*, *Netzadapter* oder *Adapter* genannt)
- IP-Adressen
- Datenbanken, inklusive von Datenbankpaaren mit einer HADR-Primärdatenbank und einer HADR-Bereitschaftsdatenbank
- Datenbankpartitionen
- Mountpunkte und -pfade, inklusive von Pfaden, die im Falle einer Störung nicht für eine Funktionsübernahme relevant sind
- Richtlinien für Funktionsübernahme
- Quorumeinheiten

Cluster-Management-Software:

Cluster-Management-Software maximiert die Workload, die ein Cluster aus Computern ausführen kann. Ein Cluster-Manager verteilt die Workload möglichst gleichmäßig, um Engpässe zu vermeiden, überwacht den Status der Clusterelemente und führt eine Funktionsübernahme durch, wenn ein Element ausfallen sollte.

Darüber hinaus kann ein Cluster-Manager Systemadministratoren die Ausführung von Verwaltungstasks für Elemente des Clusters erleichtern (z. B. durch das Umleiten von Verarbeitungsprozessen im Falle von Wartungsarbeiten am Computer).

Elemente eines Clusters

Der Cluster-Manager kann nur reibungslos ausgeführt werden, wenn er über die erforderlichen Angaben zu den Elementen des Clusters sowie zu den Beziehungen zwischen den Clusterelementen verfügt.

Der Cluster-Manager muss z. B. über Informationen zu folgenden Clusterelementen verfügen:

- Physische oder virtuelle Computer, Maschinen oder Einheiten im Cluster (im Zusammenhang mit Clustern *Clusterknoten* genannt)
- Die Clusterknoten verbindende Netze
- Netzschnittstellenkarten, die die Clusterknoten mit den Netzen verbinden
- IP-Adressen der Clusterknoten
- Virtuelle IP-Adressen oder Service-IP-Adressen

In Bezug auf die Beziehungen muss der Cluster-Manager z. B. über folgende Informationen verfügen:

- Clusterknotenpaare mit derselben Software, die für eine gegenseitige Funktionsübernahme verfügbar sind
- Netze mit denselben Eigenschaften, die für eine gegenseitige Funktionsübernahme verfügbar sind

- Clusterknoten, dem zurzeit eine virtuelle IP-Adresse zugeordnet ist

Hinzufügen oder Ändern von Clusterelementen

Die für die Clusterelemente und die Beziehungen zwischen diesen Elementen erforderlichen Angaben erhält der Cluster-Manager, indem die Elemente von einem Systemadministrator für den Cluster-Manager registriert werden. Nimmt ein Systemadministrator Änderungen an den Elementen eines Clusters vor, müssen diese Änderungen vom Administrator an den Cluster-Manager weitergegeben werden. Cluster-Manager verfügen über Schnittstellen, die für diese Tasks hilfreich sind.

Die Clusterverwaltung ist komplex, da es eine Vielzahl möglicher Clusterelemente gibt. Administratoren müssen über umfassende Kenntnisse in Bezug auf die Hardware und die Betriebssysteme der Clusterknoten, Netzprotokolle und -konfigurationen sowie die auf den Clusterknoten installierte Software verfügen (z. B. Datenbanksoftware). Die Registrierung der Clusterelemente mit der Cluster-Management-Software und das Aktualisieren des Cluster-Managers nach einer Systemänderung kann komplex und zeitaufwendig sein.

Hinzufügen und Ändern von Clusterelementen mit `db2haicu`

In einer DB2-Datenbanklösung können Sie mit DB2 High Availability Instance Configuration Utility (db2haicu) Elemente für den Cluster-Manager registrieren und den Cluster-Manager nach Verwaltungsänderungen am Cluster aktualisieren. **`db2haicu`** vereinfacht diese Tasks, da Sie kein Experte in Bezug auf die Besonderheiten der Hardware, der Betriebssysteme und der Schnittstelle des Cluster-Managers sein müssen, um diese Tasks auszuführen, sobald Sie mit dem Modell vertraut sind, das **`db2haicu`** verwendet, um die Clusterelemente und die Beziehungen zwischen diesen Elementen einzubinden.

Ressourcen und Ressourcengruppen:

Bei einer *Ressource* handelt es sich um ein Clusterelement (z. B. einen Clusterknoten, eine Datenbank, einen Mountpunkt oder eine Netzschnittstellenkarte), das mit dem Clustermanager registriert wurde. Elemente, die nicht mit dem Clustermanager registriert wurden, sind dem Clustermanager nicht bekannt und werden bei Operationen des Clustermanagers nicht berücksichtigt. Bei einer *Ressourcengruppe* handelt es sich um eine logische Sammlung von Ressourcen. Das Konstrukt der Ressourcengruppen ist sehr effizient, weil dieses Konstrukt die Definition von Beziehungen und Einschränkungen für Ressourcengruppen ermöglicht und so komplexe Verwaltungstasks für die in den Gruppen enthaltenen Ressourcen vereinfacht.

Wenn Ressourcen vom Clustermanager in Gruppen zusammengefasst werden, kann der Clustermanager diese Ressourcen anschließend gemeinsam bearbeiten. Angenommen z. B. die beiden Datenbanken `database-1` und `database-2` gehören zu der Ressourcengruppe `resource-group-A`. Führt der Clustermanager eine Startoperation für `resource-group-A` durch, werden `database-1` und `database-2` durch diese einzelne Operation des Clustermanagers gestartet.

Einschränkungen

- Ressourcengruppen können kein *Ersatzsystem* enthalten, und ein Ersatzsystem kann keine Ressourcengruppe enthalten (Ein *Ersatzsystem* besteht aus einer Gruppe von Ressourcen, die jeweils dieselbe Funktionalität bereitstellen und für eine Funktionsübernahme verwendet werden können.).
- Ressourcen können nur jeweils einer Ressourcengruppe angehören.

- Ressourcen können nicht in einer Ressourcengruppe und einem Ersatzsystem enthalten sein.
- Ressourcengruppen können weitere Ressourcengruppen enthalten. Maximale Verschachtelungsebene: 50.
- Es können bis zu 100 Ressourcen zu einer Ressourcengruppe zusammengefasst werden.

Quorumeinheiten:

Eine *Quorumeinheit* erleichtert es dem Cluster-Manager Entscheidungen in Bezug auf die Clusterverwaltung zu treffen, wenn der Standardentscheidungsprozess des Cluster-Managers nicht zu einem eindeutigen Ergebnis führt.

Wenn ein Cluster-Manager zwischen mehreren möglichen Aktionen wählen muss, zählt der Cluster-Manager die Clusterdomänenknoten, die die einzelnen möglichen Aktionen unterstützen, und wählt anschließend die Aktion aus, die von der Mehrheit der Clusterdomänenknoten unterstützt wird. Werden mehrere Aktionen von derselben Anzahl von Clusterdomänenknoten unterstützt, greift der Cluster-Manager bei seiner Entscheidung auf eine Quorumeinheit zurück.

db2haicu unterstützt die in der folgenden Tabelle aufgeführten Quorumeinheiten.

*Tabelle 2. Von **db2haicu** unterstützte Typen von Quorumeinheiten*

Quorumeinheit	Beschreibung
network	Eine Netzquorumeinheit des Typs 'network' ist eine IP-Adresse, zu der jeder einzelne Clusterdomänenknoten zu jeder beliebigen Zeit eine Verbindung herstellen kann.

Netze in einer Clusterdomäne:

Für die Konfiguration netzspezifischer Clusterdomänenelemente können Sie mit DB2 High Availability Instance Configuration Utility (db2haicu) ein *physisches Netz* zur Clusterdomäne hinzufügen. Ein physisches Netz besteht aus Netzschnittstellenkarten, IP-Adressen und Teilnetzmasken.

Netzschnittstellenkarten

Eine *Netzschnittstellenkarte* (NIC, Network Interface Card) ist eine Hardwarekomponente, die einen Computer (auch *Clusterknoten* genannt) mit einem Netz verbindet. Netzschnittstellenkarten werden teilweise auch kurz *Schnittstellen*, *Netzadapter* oder *Adapter* genannt. Wenn Sie mit **db2haicu** ein physisches Netz zu Ihrer Clusterdomäne hinzufügen, müssen Sie mindestens eine Netzschnittstellenkarte sowie den Hostnamen des Computers, zu dem die Netzschnittstellenkarte gehört, den Namen der Netzschnittstellenkarte auf dem Hostcomputer sowie die IP-Adresse der Netzschnittstellenkarte angeben.

IP-Adressen

Eine *IP-Adresse* (IP, Internet Protocol) ist eine innerhalb eines Netzes eindeutige Adresse. Bei IP Version 4 besteht eine IP-Adresse aus 32 Bit und wird normalerweise in Dezimalschreibweise mit einem Punkt als Trennzeichen angegeben (z. B. `129.30.180.16`). Eine IP-Adresse setzt sich aus einem netzspezifischen Teil und einem hostspezifischen Teil zusammen.

db2haicu unterstützt IP-Version 6 nicht.

Teilnetzmasken

Ein Netz kann mithilfe von *Teilnetzmasken* in mehrere logische Teilnetze unterteilt werden. Eine Teilnetzmaske ermöglicht es, einige Bits des hostspezifischen Teils der IP-Adresse auf den netzspezifischen Teil zu verschieben. Wenn Sie mit **db2haicu** eine IP-Adresse in Ihrer Clusterdomäne hinzufügen, müssen Sie in manchen Fällen die Teilnetzmaske für die IP-Adresse angeben. Sie müssen z. B. die Teilnetzmaske der IP-Adresse der jeweiligen Netzschnittstellenkarte angeben, wenn Sie eine Netzschnittstellenkarte mit **db2haicu** hinzufügen.

Netzersatzsysteme

Ein *Ersatzsystem* besteht aus einer Gruppe von Ressourcen, die jeweils dieselbe Funktionalität bereitstellen und für eine Funktionsübernahme verwendet werden können. Wenn Sie ein Netz mit **db2haicu** erstellen, können die Netzschnittstellenkarten in diesem Netz für eine gegenseitige Funktionsübernahme verwendet werden. Ein derartiges Netz wird auch als *Netzersatzsystem* bezeichnet.

Netzprotokolle

Wenn Sie mit **db2haicu** ein Netz zu Ihrer Clusterdomäne hinzufügen, müssen Sie den Typ des verwendeten Netzprotokolls angeben. Zurzeit wird nur das Netzprotokoll TCP/IP unterstützt.

Hinweis

Ein mit **db2haicu** konfiguriertes Netz ist nur für einen VIP-Failover erforderlich. Netzadapter, die sich in verschiedenen Teilnetzen (bzw. in verschiedenen virtuellen LANs) befinden, können nicht demselben Netz hinzugefügt werden, da für einen VIP-Failover ein allgemeines virtuelles LAN erforderlich ist.

Richtlinien für Funktionsübernahme in Clusterdomänen:

Eine *Richtlinie für Funktionsübernahme* gibt an, wie ein Cluster-Manager sich verhalten soll, wenn ein Clusterelement, wie z. B. eine Netzschnittstellenkarte oder ein Datenbankserver, ausfällt. Im Allgemeinen wird ein Cluster-Manager die Workload in diesem Fall von dem ausgefallenen Element auf ein alternatives Element übertragen, das zuvor für den Cluster-Manager als geeigneter Ersatz für das ausgefallene Element definiert wurde. Diese Übertragung der Workload von einem ausgefallenen Element auf ein anderes Element wird als *Funktionsübernahme* bezeichnet.

Funktionsübernahmerichtlinie 'Umlauf'

Die *Funktionsübernahmerichtlinie 'Umlauf'* (RoundRobin) sieht vor, dass der Datenbankmanager bei einem Ausfall eines einzigen Computers (auch *Clusterdomänenknoten* oder kurz *Knoten* genannt) in der Clusterdomäne die Workload des ausgefallenen Clusterdomänenknotens auf einem beliebigen anderen Knoten in der Clusterdomäne erneut startet.

Funktionsübernahmerichtlinie 'Gegenseitige Übernahme'

Für die Konfiguration der *Funktionsübernahmerichtlinie 'Gegenseitige Übernahme'* (Mutual) müssen Sie zwei Computer (auch *Clusterdomänenknoten* oder einfach *Knoten* genannt) in der Clusterdomäne als ein Systempaar zuordnen. Tritt bei einem dieser beiden Knoten ein Ausfall auf, wird die Workload der Datenbankpartitionen des ausgefallenen Knotens von dem anderen Knoten übernommen. Eine ge-

genseitige Übernahme ist nur möglich, wenn Sie über mehrere Datenbankpartitionen verfügen.

Funktionsübernahmerichtlinie 'M+N'

Wenn Sie die *Funktionsübernahmerichtlinie 'M+N'* verwenden, wird bei einem Ausfall eines Computers in der Clusterdomäne (auch *Clusterdomänenknoten* oder kurz *Knoten* genannt) die Workload der Datenbankpartitionen auf dem ausgefallenen Knoten von einem beliebigen anderen Knoten in der Clusterdomäne übernommen. Wenn die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten aktiviert ist, wird der letzte fehlgeschlagene Knoten zum Bereitschaftsknoten, sobald dieser fehlgeschlagene Knoten wieder in den Online-Status gesetzt wird. Die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten für die Funktionsübernahmerichtlinie M+N wird nur unterstützt, wenn M=1. Eine M+N-Übernahme ist nur möglich, wenn Sie über mehrere Datenbankpartitionen verfügen.

Funktionsübernahmerichtlinie 'Lokaler Neustart'

Wenn Sie die *Funktionsübernahmerichtlinie 'Lokaler Neustart'* (LocalRestart) verwenden, startet der Datenbankmanager bei einem Ausfall auf einem Computer (auch *Clusterdomänenknoten* oder kurz *Knoten* genannt) in der Clusterdomäne die (lokal) vorhandene Datenbank auf dem Knoten, der ausgefallen ist, neu.

Funktionsübernahmerichtlinie 'HADR'

Wenn Sie die *Funktionsübernahmerichtlinie 'HADR'* (HADRFailover) konfigurieren, ermöglichen Sie der DB2 HADR-Funktion (High Availability Disaster Recovery) die Verwaltung der Funktionsübernahme. Fällt eine HADR-Primärdatenbank aus, versetzt der Datenbankmanager die Workload von der ausgefallenen Datenbank auf eine HADR-Bereitschaftsdatenbank.

Funktionsübernahmerichtlinie 'Benutzerdefiniert'

Wenn Sie die *Funktionsübernahmerichtlinie 'Benutzerdefiniert'* (Custom) konfigurieren, erstellen Sie eine Liste von Computern (auch *Clusterdomänenknoten* oder kurz *Knoten* genannt) in der Clusterdomäne, die der Datenbankmanager für eine Funktionsübernahme nutzen kann. Fällt ein Knoten in der Clusterdomäne aus, versetzt der Datenbankmanager die Workload vom ausgefallenen Knoten auf einen in der von Ihnen angegebenen Liste enthaltenen Knoten.

Verwenden der HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten (Roving High-Availability Failover; RHAF) in Umgebungen mit partitionierten Datenbanken:

Wenn Sie die Richtlinie für Funktionsübernahmen 'M+N' mit 'N' aktiven Knoten und einem Bereitschaftsknoten verwenden, können Sie die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten aktivieren.

Vorbereitende Schritte

Für jeden Knoten im Cluster muss die Unterstützung für die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten aktiviert oder inaktiviert sein.

In Umgebungen mit partitionierten Datenbanken, bei denen die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten nicht aktiviert ist, ist der vorgesehene Bereitschaftsknoten in der Regel der der einzige Knoten mit Zugriff auf alle Platten und Datenträgergruppen einschließlich der Dateisysteme in diesen Daten-

trägergruppen. Stellen Sie in diesen Umgebungen sicher, dass für die externen LUN-Speicherzuordnungen sowie die SAN-Zonen im Cluster alle Platten der Datenbankinstanz sichtbar sind. Sie müssen außerdem überprüfen, ob alle Datenträgergruppen, die vom Cluster gesteuert werden, in alle Clusterknoten importiert wurden. Inaktivieren Sie nach dem Import der Datenträgergruppen das Attribut 'auto-varyon' für Datenträgergruppen sowie das Attribut 'auto-mount' der Dateisysteme in allen aktiven Clusterknoten.

Wenn Sie die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten verwenden möchten, müssen Sie sie erneut aktivieren und hierfür nach der Anwendung eines neuen Fixpacks diese Schritte ausführen.

Informationen zu diesem Vorgang

Wenn Sie die Richtlinie für Funktionsübernahmen *M+N* mit 'N' aktiven Knoten und einem Bereitschaftsknoten verwenden, kommt es beim Ausfall eines aktiven Knotens zu einer Funktionsübernahmeoperation. Der Bereitschaftsknoten beginnt dann mit dem Hosting der Ressourcen des ausgefallenen Knotens. Wenn der ausgefallene Knoten den Online-Status wiedererlangt, muss die Clusterumgebung in der Regel wieder in den Offline-Status versetzt werden, damit der Knoten, der ursprünglich als Bereitschaftsknoten ausgewählt wurde, wieder zum Bereitschaftsknoten wird. Sie können die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten so konfigurieren, dass der letzte ausgefallene Knoten im Cluster zum Bereitschaftsknoten wird, ohne dass zusätzliche Zurücksetzungsoperationen notwendig wären.

Vorgehensweise

Gehen Sie wie folgt vor, um die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten zu aktivieren:

1. Stellen Sie sicher, dass keine Funktionsübernahmeoperationen in Bearbeitung sind.
2. Erstellen Sie eine Backup-Kopie des Scripts `db2V10_start.ksh`, das sich im Verzeichnis `sqllib\samples\tsa` befindet.
3. Bearbeiten Sie das Script `db2V10_start.ksh`. Suchen Sie nach der folgenden Zeile:

   ```
   ROVING_STANDBY_ENABLED=false
   ```

   Nehmen Sie die folgenden Änderungen daran vor:

   ```
   ROVING_STANDBY_ENABLED=true
   ```
4. Speichern Sie Ihre Änderungen.

Ergebnisse

Die Änderung wird bei der nächsten Funktionsübernahmeoperation wirksam.

Nächste Schritte

Wenn Sie die Unterstützung für die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten inaktivieren möchten, müssen Sie für jeden Knoten die folgenden Schritte ausführen:

1. Stellen Sie sicher, dass keine Funktionsübernahmeoperationen in Bearbeitung sind.

2. Bearbeiten Sie das Script db2V10_start.ksh. Suchen Sie nach der folgenden Zeile:

   ```
   ROVING_STANDBY_ENABLED=true
   ```

   Nehmen Sie die folgenden Änderungen daran vor:

   ```
   ROVING_STANDBY_ENABLED=false
   ```

3. Speichern Sie Ihre Änderungen. Die Änderung wird bei der nächsten Funktionsübernahmeoperation wirksam.

Mountpunkte in einer Clusterdomäne:

Nach dem Anhängen eines Dateisystems können Sie den jeweiligen Mountpunkt mit DB2 High Availability Instance Configuration Utility (db2haicu) in der Clusterdomäne hinzufügen.

Mountpunkte

Bei UNIX-, Linux- und AIX-Betriebssystemen werden Dateisysteme über einen *Mountvorgang* angehängt, um sie dem Betriebssystem zur Verfügung zu stellen. Während des Mountvorgangs führt das Betriebssystems Tasks wie das Lesen der Index- oder Navigationsdatenstrukturen durch und ordnet dem angehängten Dateisystem einen Verzeichnispfad zu. Dieser zugeordnete Verzeichnispfad, über den Sie auf das angehängte Dateisystem zugreifen können, wird als *Mountpunkt* bezeichnet.

Nicht kritische Mountpunkte oder -pfade

Der Cluster enthält möglicherweise Mountpunkte oder -pfade, für die im Falle einer Störung keine Funktionsübernahme erforderlich ist. Mit **db2haicu** können Sie eine Liste dieser Mountpunkte oder - pfade zu Ihrer Clusterdomäne hinzufügen. Mountpunkte oder -pfade in dieser Liste werden vom Cluster-Manager nicht in Funktionsübernahmen einbezogen.

Angenommen z. B. ein Festplattenlaufwerk ist über /mnt/driveA an den Computer node1 im Cluster angehängt. Wenn Sie entscheiden, dass /mnt/driveA von kritischer Bedeutung ist und verfügbar sein muss, führt der Cluster-Manager im Falle einer Störung eine Funktionsübernahme durch, damit /mnt/driveA verfügbar bleibt, wenn node1 ausfällt. Wenn Sie jedoch entscheiden, dass /mnt/driveA nicht unbedingt zur Verfügung stehen muss, wenn node1 ausfällt, können Sie dem Cluster-Manager angeben, dass /mnt/driveA nicht von kritischer Bedeutung ist, indem Sie /mnt/driveA in die Liste der nicht kritischen Pfade aufnehmen. Ist /mnt/driveA in Bezug auf Funktionsübernahmen als nicht kritisch eingestuft, führt dies dazu, dass das betreffende Laufwerk möglicherweise nicht verfügbar ist, wenn node1 ausfällt.

DB2 High Availability Instance Configuration Utility (db2haicu)

DB2 High Availability Instance Configuration Utility (db2haicu) ist ein textbasiertes Dienstprogramm zur Konfiguration und Verwaltung hoch verfügbarer Datenbanken in einer Clusterumgebung.

db2haicu stellt Abfragen an das System, um Informationen zur Datenbankinstanz, zur Clusterumgebung und zum Cluster-Manager zu sammeln. Zur Angabe weiterer Informationen über Parameter für den Aufruf **db2haicu**, eine Eingabedatei oder während der Laufzeit können Sie die Eingabeaufforderungen von **db2haicu** verwenden.

Syntax

```
db2haicu [ -f name-der-XML-eingabedatei ]
         [ -disable ]
         [ -delete [ dbpartitionnum liste-der-datenbankpartitionen |
                     hadrdb datenbankname ] ]
```

Parameter

Bei der Übergabe von Parametern an den Befehl **db2haicu** muss die Groß-/Kleinschreibung beachtet werden; die Eingabe muss in Kleinbuchstaben erfolgen.

-f *name-der-XML-eingabedatei*
: Unter Verwendung des Parameters **-f** können Sie Details zur Clusterdomäne in der XML-Eingabedatei *name-der-XML-eingabedatei* angeben.Weitere Informationen hierzu finden Sie in „Ausführen von DB2 High Availability Instance Configuration Utility (db2haicu) mit einer XML-Eingabedatei" auf Seite 114.

-disable
: Eine Datenbankmanagerinstanz ist *für hohe Verfügbarkeit konfiguriert*, sobald Sie mithilfe von **db2haicu** eine Clusterdomäne für diese Instanz erstellt haben. Wenn eine Datenbankmanagerinstanz für hohe Verfügbarkeit konfiguriert wurde und bestimmte Verwaltungsoperationen des Datenbankmanagers entsprechende Änderungen an der Clusterkonfiguration erforderlich machen, teilt der Datenbankmanager dem Cluster-Manager diese Änderungen mit. Wenn der Datenbankmanager diese Clusterverwaltungstasks mit dem Cluster-Manager für Sie koordiniert, müssen Sie keine separate Cluster-Manager-Operation für diese Verwaltungstasks ausführen. Diese Integration zwischen Datenbankmanager und Cluster-Manager ist eine Funktion von DB2 High Availability Feature.

  Mit dem Parameter **-disable** können Sie angeben, dass eine Datenbankmanagerinstanz nicht mehr für hohe Verfügbarkeit konfiguriert sein soll. Wenn die Datenbankmanagerinstanz nicht mehr für hohe Verfügbarkeit konfiguriert ist, nimmt der Datenbankmanager keine Koordination mit dem Cluster-Manager vor, wenn Verwaltungsoperationen des Datenbankmanagers entsprechende Änderungen an der Clusterkonfiguration erforderlich machen.

  Führen Sie **db2haicu** erneut aus, um eine Datenbankmanagerinstanz für hohe Verfügbarkeit zu rekonfigurieren.

-delete
: Mit dem Parameter **-delete** können Sie Ressourcengruppen für die aktuelle Datenbankmanagerinstanz löschen.

  Wenn Sie weder den Parameter **dbpartitionnum** noch den Parameter **hadrdb** verwenden, entfernt **db2haicu** alle Ressourcengruppen, die der aktuellen Datenbankmanagerinstanz zugeordnet sind.

  dbpartitionnum *liste-der-datenbankpartitionen*
  : Mit dem Parameter **dbpartitionnum** können Sie Ressourcengruppen löschen, die den in *liste_der_datenbankpartitionen* aufgelisteten Datenbankpartitionen zugeordnet sind. *liste_der_datenbankpartitionen* ist eine durch Kommas getrennte Liste mit Nummern, die die Datenbankpartitionen identifizieren.

  hadrdb *datenbankname*
  : Mit dem Parameter **hadrdb** können Sie Ressourcengruppen löschen, die der HADR-Datenbank *datenbankname* zugeordnet sind.

Wenn in der Clusterdomäne keine Ressourcengruppen mehr vorhanden sind, nachdem **db2haicu** die Ressourcengruppen entfernt hat, entfernt **db2haicu** auch die Clusterdomäne.

Die Ausführung von **db2haicu** mit dem Parameter **-delete** hat zur Folge, dass die aktuelle Datenbankmanagerinstanz nicht mehr für hohe Verfügbarkeit konfiguriert ist. Wenn die Datenbankmanagerinstanz nicht mehr für hohe Verfügbarkeit konfiguriert ist, nimmt der Datenbankmanager keine Koordination mit dem Cluster-Manager vor, wenn Verwaltungsoperationen des Datenbankmanagers entsprechende Änderungen an der Clusterkonfiguration erforderlich machen.

Führen Sie **db2haicu** erneut aus, um eine Datenbankmanagerinstanz für hohe Verfügbarkeit zu rekonfigurieren.

DB2 High Availability Instance Configuration Utility (db2haicu) - Startmodus:

Wenn Sie DB2 High Availability Instance Configuration Utility (db2haicu) zum ersten Mal für eine Datenbankmanagerinstanz ausführen, wird **db2haicu** im Startmodus ausgeführt.

Bei der Ausführung von **db2haicu** überprüft **db2haicu** die Datenbankmanagerinstanz und die Systemkonfiguration und sucht nach einer vorhandenen *Clusterdomäne*. Eine Clusterdomäne ist ein Modell, das Informationen zu Clusterelementen wie Datenbanken, Mountpunkten und Richtlinien für Funktionsübernahme enthält. Clusterdomänen werden mit DB2 High Availability Instance Configuration Utility (db2haicu) erstellt.

Wenn Sie **db2haicu** für eine Datenbankmanagerinstanz ausführen und für diese Instanz noch keine Clusterdomäne erstellt und konfiguriert wurde, beginnt **db2haicu** umgehend mit der Erstellung und Konfiguration einer neuen Clusterdomäne. Bei der Erstellung einer neuen Clusterdomäne werden Sie von **db2haicu** zur Eingabe von Informationen wie beispielsweise einem Namen für die neue Clusterdomäne oder dem Hostnamen der aktuellen Maschine aufgefordert.

Wenn Sie eine Clusterdomäne erstellen, diese jedoch nicht vollständig konfigurieren, setzt **db2haicu** bei der nächsten Ausführung die Konfiguration der Clusterdomäne fort.

Nachdem Sie eine Clusterdomäne für eine Datenbankmanagerinstanz erstellt und konfiguriert haben, wird **db2haicu** im Wartungsmodus ausgeführt.

DB2 High Availability Instance Configuration Utility (db2haicu) - Wartungsmodus:

Wenn Sie DB2 High Availability Instance Configuration Utility (db2haicu) ausführen und für die aktuelle Datenbankmanagerinstanz bereits eine Clusterdomäne erstellt wurde, wird **db2haicu** im Wartungsmodus ausgeführt.

Wenn **db2haicu** im Wartungsmodus ausgeführt wird, zeigt **db2haicu** eine Liste der Konfigurations- und Verwaltungstasks an, die Sie ausführen können.

Zu den Wartungsaufgaben von **db2haicu** gehört das Hinzufügen von Datenbank- oder Clusterelementen wie beispielsweise Datenbanken oder Clusterknoten zur Clusterdomäne sowie das Entfernen von Elementen aus der Clusterdomäne. Ferner

gehört zu den Wartungsaufgaben von **db2haicu** das Modifizieren der Details von Clusterdomänenelementen wie der Richtlinie für Funktionsübernahme für die Datenbankmanagerinstanz.

Bei der Ausführung von **db2haicu** im Wartungsmodus zeigt **db2haicu** eine Liste der Operationen an, die Sie für die Clusterdomäne ausführen können:

- Fügen Sie Clusterknoten hinzu, oder entfernen Sie sie. (Die Maschine wird durch den Hostnamen identifiziert.)
- Fügen Sie eine Netzschnittstelle (Netzschnittstellenkarte) hinzu, oder entfernen Sie sie.
- Fügen Sie DB2-Datenbankpartitionen hinzu, oder entfernen Sie sie. (Nur in Umgebungen mit partitionierten Datenbanken.)
- Fügen Sie DB2-HADR-Datenbanken hinzu oder entfernen Sie sie.
- Fügen Sie eine Hochverfügbarkeitsdatenbank hinzu, oder entfernen Sie sie.
- Fügen Sie einen Mountpunkt hinzu, oder entfernen Sie ihn.
- Fügen Sie eine IP-Adresse hinzu, oder entfernen Sie sie.
- Fügen Sie einen nicht kritischen Pfad hinzu, oder entfernen Sie ihn.
- Versetzen Sie DB2-Datenbankpartitionen und HADR-Datenbanken für die planmäßige Wartung.
- Ändern Sie die Richtlinie für Funktionsübernahme für diese Instanz.
- Erstellen Sie eine neue Quorumeinheit für die Domäne.
- Löschen Sie die Domäne.

Ausführen von DB2 High Availability Instance Configuration Utility (db2haicu) im interaktiven Modus:

Wenn Sie DB2 High Availability Instance Configuration Utility (db2haicu) mit dem Befehl **db2haicu** aufrufen und keine XML-Eingabedatei mit dem Parameter **-f** angeben, wird das Dienstprogramm im interaktiven Modus ausgeführt. Im interaktiven Modus von **db2haicu** werden Informationen in einem textbasierten Format angezeigt und abgefragt.

Vorbereitende Schritte

- Vor der Verwendung von DB2 High Availability Instance Configuration Utility (db2haicu) müssen bestimmte Tasks ausgeführt werden. Weitere Informationen hierzu finden Sie in „DB2 High Availability Instance Configuration Utility (db2haicu) - Voraussetzungen” auf Seite 143.

Informationen zu diesem Vorgang

Bei der Ausführung von **db2haicu** im interaktiven Modus werden Informationen und Fragen im Textformat auf dem Bildschirm angezeigt. Sie können die von **db2haicu** angeforderten Informationen in einer Eingabeaufforderung eingeben, die sich am unteren Rand des Bildschirms befindet.

Vorgehensweise

Rufen Sie zur Ausführung von **db2haicu** im interaktiven Modus den Befehl **db2haicu** ohne den Parameter `-f` *name-der-eingabedatei* auf.

Nächste Schritte

DB2 High Availability Instance Configuration Utility (db2haicu) verfügt über kein separates Diagnoseprotokoll. Fehler bei **db2haicu** können mit dem Diagnoseprotokoll des Datenbankmanagers, der Protokolldatei **db2diag.log** und dem Tool **db2pd** untersucht und diagnostiziert werden. Weitere Informationen hierzu finden Sie in „Fehlerbehebung bei DB2 High Availability Instance Configuration Utility (db2haicu)" auf Seite 147.

Ausführen von DB2 High Availability Instance Configuration Utility (db2haicu) mit einer XML-Eingabedatei:

Sie können den Parameter `-f` *`name-der-eingabedatei`* mit dem Befehl **db2haicu** verwenden, um DB2 High Availability Instance Configuration Utility (db2haicu) mit einer XML-Eingabedatei auszuführen, in der die Konfigurationsdetails angegeben sind. Die Ausführung von **db2haicu** mit einer XML-Eingabedatei ist nützlich, wenn Sie bestimmte Konfigurationstasks mehrmals ausführen müssen, z. B. bei der Konfiguration mehrerer Datenbankpartitionen für hohe Verfügbarkeit.

Vorbereitende Schritte

- Vor der Verwendung von DB2 High Availability Instance Configuration Utility (db2haicu) müssen bestimmte Tasks ausgeführt werden. Weitere Informationen hierzu finden Sie in „DB2 High Availability Instance Configuration Utility (db2haicu) - Voraussetzungen" auf Seite 143.

Informationen zu diesem Vorgang

Im Verzeichnis `sqllib` befindet sich im Unterverzeichnis `samples` eine Gruppe von XML-Beispieleingabedateien, die Sie modifizieren und mit **db2haicu** zum Konfigurieren der Clusterumgebung verwenden können. Weitere Informationen hierzu finden Sie in „XML-Beispieleingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu)" auf Seite 134.

Ein detailliertes Szenario für die Einrichtung eines HADR-Paares unter Verwendung von **db2haicu** mit einer XML-Beispieleingabedatei enthält das Dokument „Automated Cluster Controlled HADR (High Availability Disaster Recovery) Configuration Setup using the IBM DB2 High Availability Instance Configuration Utility (db2haicu)".

Vorgehensweise

1. Erstellen Sie eine XML-Eingabedatei.Sie verwenden dieselbe XML-Datei, falls Sie Datenbankpartitionen oder - in einer HADR-Konfiguration - die Primärdatenbank und die Bereitschaftsdatenbank konfigurieren.
2. Rufen Sie **db2haicu** mit dem Parameter `-f` *`name-der-eingabedatei`* auf. Gehen Sie bei einer HADR-Konfiguration folgendermaßen vor:
   a. Melden Sie sich bei der Bereitschaftsdatenbankinstanz an und setzen Sie den Befehl ab.
   b. Melden Sie sich nach der Beendigung von **db2haicu** bei der Primärdatenbankinstanz an und setzen Sie den Befehl ab.

Nächste Schritte

DB2 High Availability Instance Configuration Utility (db2haicu) verfügt über kein separates Diagnoseprotokoll. Fehler bei **db2haicu** können mit dem Diagnoseprotokoll des Datenbankmanagers, der Protokolldatei **db2diag.log** und dem Tool **db2pd**

untersucht und diagnostiziert werden. Weitere Informationen hierzu finden Sie in „Fehlerbehebung bei DB2 High Availability Instance Configuration Utility (db2haicu)” auf Seite 147.

XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Die DB2 High Availability Instance Configuration Utility (db2haicu)-XML-Schemadefinition (XSD) für Eingabedateien definiert die Clusterdomänenobjekte, die Sie in einer **db2haicu**-XML-Eingabedatei angeben können. Diese **db2haicu**-XSD befindet sich in der Datei `db2ha.xsd` im Verzeichnis `sqllib/samples/ha/xml`.

DB2ClusterType

Das Stammelement der **db2haicu**-XSD ist das Element `DB2Cluster`, das den Typ `DB2ClusterType` aufweist. Eine **db2haicu**-XML-Eingabedatei muss mit dem Element `DB2Cluster` beginnen.

„XML-Schemadefinition ”
„Subelemente”
„Attribute” auf Seite 117
„Hinweise zur Verwendung” auf Seite 117

XML-Schemadefinition

```
<xs:complexType name='DB2ClusterType'>
  <xs:sequence>
    <xs:element name='DB2ClusterTemplate'
                type='DB2ClusterTemplateType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='ClusterDomain'
                type='ClusterDomainType'
                maxOccurs='unbounded'/>
    <xs:element name='FailoverPolicy'
                type='FailoverPolicyType'
                minOccurs='0' />
    <xs:element name='DB2PartitionSet'
                type='DB2PartitionSetType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='HADRDBSet'
                type='HADRDBType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='HADBSet'
                type='HADBType'
                minOccurs='0'
                maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='clusterManagerName' type='xs:string' use='optional'/>
</xs:complexType>
```

Subelemente

DB2ClusterTemplate

Typ: `DB2ClusterTemplateType`

Hinweise zur Verwendung:
Fügen Sie das Element `DB2ClusterTemplateType` nicht in der **db2haicu**-XML-Eingabedatei ein. Das Element `DB2ClusterTemplateType` ist für zukünftige Releases reserviert.

ClusterDomain

Typ: `ClusterDomainType`

Das Element `ClusterDomainType` enthält Spezifikationen zu den Maschinen oder Computern (auch *Clusterdomänenknoten* genannt) in der Clusterdomäne, zu den *Netzersatzsystemen* (Gruppen von Netzen mit gegenseitiger Funktionsübernahme) und zur *Quorumeinheit* (Zuteilungsroutinenmechanismus).

Regeln zur Häufigkeit:
Sie müssen mindestens ein Element `ClusterDomain` in das Element `DB2ClusterType` einfügen.

FailoverPolicy

Typ: `FailoverPolicyType`

Das Element `FailoverPolicyType` gibt die *Richtlinie für Funktionsübernahme* an, die der Cluster-Manager in der Clusterdomäne anwenden soll.

Regeln zur Häufigkeit:
Sie können kein oder ein Element `FailoverPolicy` in das Element `DB2ClusterType` einfügen.

DB2PartitionSet

Typ: `DB2PartitionSetType`

Das Element `DB2PartitionSetType` enthält Informationen zu Datenbankpartitionen. Eine Verwendung des Elements `DB2PartitionSetType` ist nur in Umgebungen mit partitionierten Datenbanken möglich.

Regeln zur Häufigkeit:
Sie können entsprechend der **db2haicu**-XML-Schemadefinition kein oder beliebig viele Elemente `DB2PartitionSet` in das Element `DB2ClusterType` einfügen.

HADRDBSet

Typ: `HADRDBType`

Das Element `HADRDBType` enthält eine Liste mit Datenbankpaaren, die aus einer HADR-Primärdatenbank (High Availability Disaster Recovery) und einer HADR-Bereitschaftsdatenbank bestehen.

Regeln zur Häufigkeit:
Sie können entsprechend der **db2haicu**-XML-Schemadefinition kein oder beliebig viele Elemente `HADRDBSet` in das Element `DB2ClusterType` einfügen.

Hinweise zur Verwendung:

- Sie dürfen das Element `HADRDBSet` nicht in einer Umgebung mit partitionierten Datenbanken einfügen.
- Wenn Sie das Element `HADRDBSet` einfügen, müssen Sie als Richtlinie für Übernahme `HADRFailover` im Element `FailoverPolicy` angeben.

HADBSet

Typ: `HADBType`

Das Element HADBType enthält eine Liste der Datenbanken, die zu der Clusterdomäne gehören und eine hohe Verfügbarkeit aufweisen sollen.

Regeln zur Häufigkeit:
Sie können entsprechend der **db2haicu**-XML-Schemadefinition kein oder beliebig viele Elemente HADBSet in das Element DB2ClusterType einfügen.

Attribute

clusterManagerName (optional)
Das Attribut clusterManagerName gibt den Cluster-Manager an.

Gültige Werte für dieses Attribut enthält die folgende Tabelle:

Tabelle 3. Gültige Werte für das Attribut clusterManager

Wert für clusterManagerName	Cluster-Managerprodukt
TSA	IBM Tivoli System Automation for Multiplatforms (SA MP)

Hinweise zur Verwendung

In einer Umgebung mit Einzelpartitionsdatenbank wird in der Regel nur eine einzige Clusterdomäne pro Datenbankmanagerinstanz erstellt.

Für eine Umgebung mit partitionierten Datenbanken ist z. B. die folgende Konfiguration möglich:

- Definieren Sie das Element FailoverPolicy mit Mutual.
- Verwenden Sie im Subelement DB2Partition von DB2PartitionSet das Element MutualPair, um zwei Clusterdomänenknoten anzugeben, die in einer einzigen Clusterdomäne enthalten sind.

ClusterDomainType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element ClusterDomainType enthält Spezifikationen zu den Maschinen oder Computern (auch *Clusterdomänenknoten* genannt) in der Clusterdomäne, zu den *Netzersatzsystemen* (Gruppen von Netzen mit gegenseitiger Funktionsübernahme) und zur *Quorumeinheit* (Zuteilungsroutinenmechanismus).

„Superelemente"
„XML-Schemadefinition "
„Subelemente" auf Seite 118
„Attribute" auf Seite 118

Superelemente

Die folgenden Elementtypen enthalten ClusterDomainType-Subelemente:

- DB2ClusterType

XML-Schemadefinition

```
<xs:complexType name='ClusterDomainType'>
  <xs:sequence>
    <xs:element name='Quorum'
                type='QuorumType'
```

```
                minOccurs='0' />
    <xs:element name='PhysicalNetwork'
                type='PhysicalNetworkType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='ClusterNode'
                type='ClusterNodeType'
                maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='domainName'      type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

Quorum

Typ: QuorumType

Das Element QuorumType gibt die *Quorumeinheit* für die Clusterdomäne an.

Regeln zur Häufigkeit:
Sie können kein oder ein Element Quorum in das Element ClusterDomainType einfügen.

PhysicalNetwork

Typ: PhysicalNetworkType

Das Element PhysicalNetworkType enthält Netzschnittstellenkarten, die für eine gegenseitige Funktionsübernahme verfügbar sind. Dies wird auch als *Netzersatzsystem* bezeichnet.

Regeln zur Häufigkeit:
Sie können kein oder beliebig viele Elemente PhysicalNetwork in das Element ClusterDomainType einfügen.

ClusterNode

Typ: ClusterNodeType

Das Element ClusterNodeType enthält Informationen zu einem bestimmten Computer oder einer bestimmten Maschine (auch *Clusterdomänenknoten* genannt) des Clusters.

Regeln zur Häufigkeit:
Sie müssen mindestens ein Element ClusterNode im Element ClusterDomainType angeben.

Hinweise zur Verwendung
IBM Tivoli System Automation for Multiplatforms (SA MP) unterstützt bis zu 32 Clusterdomänenknoten. Wenn es sich beim Clustermanager um SA MP handelt, können Sie bis zu 32 Elemente ClusterNode in das Element ClusterDomainType einfügen.

Attribute

domainName (erforderlich)

Geben Sie einen eindeutigen Namen für das Element ClusterDomainType an.

Bei der Verwendung von RSCT (Reliable Scalable Cluster Technology) zur Clusterverwaltung gelten für 'domainName' die folgenden Einschränkungen:

- domainName darf nur die Zeichen A bis Z, a bis z, die Ziffern 0 bis 9, Punkt (.) und Unterstreichungszeichen (_) enthalten.
- domainName darf nicht "IW" lauten.

Das folgende Beispiel zeigt ein ClusterDomainType-Element:

```
<ClusterDomain domainName="hadr_linux_domain">
 <Quorum quorumDeviceProtocol="network" quorumDeviceName="9.26.4.5"/>
  <PhysicalNetwork physicalNetworkName="db2_public_network_0"
   physicalNetworkProtocol="ip">
   <Interface interfaceName="eth0" clusterNodeName="linux01">
    <IPAddress baseAddress="9.26.124.30" subnetMask="255.255.255.0"
     networkName="db2_public_network_0"/>
   </Interface>
   <Interface interfaceName="eth0" clusterNodeName="linux02">
    <IPAddress baseAddress="9.26.124.31" subnetMask="255.255.255.0"
     networkName="db2_public_network_0"/>
   </Interface>
  </PhysicalNetwork>
 <ClusterNode clusterNodeName="linux01"/>
 <ClusterNode clusterNodeName="linux02"/>
</ClusterDomain>
```

QuorumType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element QuorumType gibt die *Quorumeinheit* für die Clusterdomäne an.

„Superelemente"
„XML-Schemadefinition "
„Subelemente"
„Attribute"

Superelemente

Die folgenden Elementtypen enthalten QuorumType-Subelement:

- ClusterDomainType

XML-Schemadefinition

```
<xs:complexType name='QuorumType'>
  <xs:attribute name='quorumDeviceProtocol'
                type='QuorumDeviceProtocolType'
                use='required'/>
  <xs:attribute name='quorumDeviceName'
                type='xs:string'
                use='required'/>
</xs:complexType>
```

Subelemente

Keine.

Attribute

quorumDeviceProtocol (erforderlich)

quorumDeviceProtocol gibt den Typ des zu verwendenden Quorums an.

Eine *Quorumeinheit* erleichtert es dem Cluster-Manager Entscheidungen in Bezug auf die Clusterverwaltung zu treffen, wenn der Standardentscheidungsprozess des Cluster-Managers nicht zu einem eindeutigen Ergebnis führt.

Der Typ des Attributs quorumDeviceProtocol ist QuorumDeviceProtocolType.

Die XML-Schemadefinition für das Element QuorumDeviceProtocolType sieht wie folgt aus:

```
<xs:simpleType name='QuorumDeviceProtocolType'>
  <xs:restriction base='xs:string'>
    <xs:enumeration value='disk'/>
    <xs:enumeration value='scsi'/>
    <xs:enumeration value='network'/>
    <xs:enumeration value='eckd'/>
    <xs:enumeration value='mns'/>
  </xs:restriction>
</xs:simpleType>
```

Zurzeit werden die in der folgenden Tabelle enthaltenen Werte für dieses Attribut unterstützt:

Tabelle 4. Gültige Werte für das Attribut quorumDeviceProtocol

Wert für **quorumDeviceProtocol**	**Bedeutung**
network	Eine Netzquorumeinheit des Typs 'network' ist eine IP-Adresse, zu der jeder einzelne Clusterdomänenknoten zu jeder beliebigen Zeit eine Verbindung herstellen kann.

quorumDeviceName (erforderlich)

Der Wert für quorumDeviceName richtet sich nach dem in quorumDeviceProtocol angegebenen Quorumeinheitentyp.

Gültige Werte für dieses Attribut enthält die folgende Tabelle:

Tabelle 5. Gültige Werte für das Attribut quorumDeviceName

Wert für **quorumDeviceProtocol**	**Gültige Werte für quorumDeviceName**
network	Eine Zeichenfolge mit einer korrekt formatierten IP-Adresse. Beispiel: 12.126.4.5 Die angegebene IP-Adresse ist nur für eine Netzquorumeinheit gültig, wenn jeder einzelne Clusterdomänenknoten auf die betreffende IP-Adresse zugreifen kann (z. B. mithilfe des Dienstprogramms ping).

PhysicalNetworkType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element PhysicalNetworkType enthält Netzschnittstellenkarten, die für eine gegenseitige Funktionsübernahme verfügbar sind. Dies wird auch als *Netzersatzsystem* bezeichnet.

„Superelemente"
„XML-Schemadefinition " auf Seite 121
„Subelemente " auf Seite 121
„Attribute" auf Seite 121

Superelemente

Die folgenden Elementtypen enthalten PhysicalNetworkType-Subelemente:

- ClusterDomainType

XML-Schemadefinition

```
<xs:complexType name='PhysicalNetworkType'>
  <xs:sequence>
    <xs:element name='Interface'
                type='InterfaceType'
                minOccurs='1'
                maxOccurs='unbounded'/>
    <xs:element name='LogicalSubnet'
                type='IPAddressType'
                minOccurs='0'
                maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='physicalNetworkName'
                type='xs:string'
                use='required'/>
  <xs:attribute name='physicalNetworkProtocol'
                type='PhysicalNetworkProtocolType'
                use='required'/>
</xs:complexType>
```

Subelemente

Interface

Typ: `InterfaceType`

Das Element `InterfaceType` besteht aus einer IP-Adresse, dem Namen eines Computers oder einer Maschine im Netz (auch *Clusterdomänenknoten* genannt) und dem Namen einer *Netzschnittstellenkarte* (NIC) des Clusterdomänenknotens.

Regeln zur Häufigkeit:
Sie müssen mindestens ein Element `Interface` im Element `PhysicalNetworkType` angeben.

LogicalSubnet

Typ: `IPAddressType`

Das Element `IPAddressType` enthält alle Einzelangaben zu einer IP-Adresse, z. B. die *Basisadresse*, die *Teilnetzmaske* und den Namen des Netzes, zu dem die IP-Adresse gehört.

Regeln zur Häufigkeit:
Sie können kein oder beliebig viele Elemente `LogicalSubnet` im Element `PhysicalNetworkType` angeben.

Attribute

physicalNetworkName (erforderlich)
Geben Sie einen eindeutigen Namen des zugehörigen physischen Netzes für `physicalNetworkName` für jedes einzelne Element `PhysicalNetworkType` an.

physicalNetworkProtocol (erforderlich)
Der Typ des Attributs `physicalNetworkProtocol` ist `PhysicalNetworkProtocolType`.

Die XML-Schemadefinition für das Element `PhysicalNetworkProtocolType` sieht wie folgt aus:

```
<xs:simpleType name='PhysicalNetworkProtocolType'>
  <xs:restriction base='xs:string'>
    <xs:enumeration value='ip'/>
    <xs:enumeration value='rs232'/>
    <xs:enumeration value='scsi'/>
```

```
    <xs:enumeration value='ssa'/>
    <xs:enumeration value='disk'/>
  </xs:restriction>
</xs:simpleType>
```

Zurzeit werden die in der folgenden Tabelle enthaltenen Werte für dieses Attribut unterstützt:

Tabelle 6. Gültige Werte für das Attribut `physicalNetworkProtocol`

Wert für **physicalNetworkProtocol**	Bedeutung
ip	TCP/IP-Protokoll

InterfaceType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element `InterfaceType` besteht aus einer IP-Adresse, dem Namen eines Computers oder einer Maschine im Netz (auch *Clusterdomänenknoten* genannt) und dem Namen einer *Netzschnittstellenkarte* (NIC) des Clusterdomänenknotens.

„Superelemente"
„XML-Schemadefinition "
„Subelemente "
„Attribute"

Superelemente

Die folgenden Elementtypen enthalten `InterfaceType`-Subelemente:

- `PhysicalNetworkType`

XML-Schemadefinition

```
<xs:complexType name='InterfaceType'>
  <xs:sequence>
    <xs:element name='IPAddress'       type='IPAddressType'/>
  </xs:sequence>
  <xs:attribute name='interfaceName'   type='xs:string' use='required'/>
  <xs:attribute name='clusterNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

IPAddress

Typ: `IPAddressType`

Das Element `IPAddressType` enthält alle Einzelangaben zu einer IP-Adresse, z. B. die *Basisadresse*, die *Teilnetzmaske* und den Namen des Netzes, zu dem die IP-Adresse gehört.

Regeln zur Häufigkeit:
Geben Sie eine einzige IP-Adresse für `IPAddress` im Element `InterfaceType` an.

Attribute

interfaceName (erforderlich)
Geben Sie den Namen einer Netzschnittstellenkarte im Attribut `interfaceName` an. Die für `interfaceName` angegebene Netzschnittstellenkarte muss in dem Clusterdomänenknoten vorhanden sein, den Sie für das Attribut `clusterNodeName` angeben.

clusterNodeName (erforderlich)
Geben Sie den Namen des Clusterdomänenknotens an, der der IP-Adresse zugeordnet ist, die Sie für das Element `IPAddress` angeben.

IPAddressType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element `IPAddressType` enthält alle Einzelangaben zu einer IP-Adresse, z. B. die *Basisadresse*, die *Teilnetzmaske* und den Namen des Netzes, zu dem die IP-Adresse gehört.

„Superelemente"
„XML-Schemadefinition "
„Subelemente "
„Attribute"

Superelemente

Die folgenden Elementtypen enthalten `IPAddressType`-Subelemente:

- `PhysicalNetworkType`
- `InterfaceType`
- `DB2PartitionType`

XML-Schemadefinition

```
<xs:complexType name='IPAddressType'>
  <xs:attribute name='baseAddress' type='xs:string' use='required'/>
  <xs:attribute name='subnetMask'  type='xs:string' use='required'/>
  <xs:attribute name='networkName' type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

Keine.

Attribute

baseAddress (erforderlich)
Geben Sie die IP-Basisadresse in Form einer Zeichenfolge an, die ein gültiges IP-Adressformat aufweist: vier durch einen Punkt getrennte Zifferngruppen im Bereich von 0 bis 255. Beispiel:

```
162.148.31.101
```

subnetMask (erforderlich)
Geben Sie die IP-Basisadresse in Form einer Zeichenfolge an, die ein gültiges IP-Adressformat aufweist.

networkName (erforderlich)
Geben Sie für `networkName` denselben Wert an wie für das Attribut `physicalNetworkName` des Elements PhysicalNetworkType, in dem das Element `IPAddress` enthalten ist.

ClusterNodeType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element `ClusterNodeType` enthält Informationen zu einem bestimmten Computer oder einer bestimmten Maschine (auch *Clusterdomänenknoten* genannt) des Clusters.

„Superelemente”
„XML-Schemadefinition ”
„Subelemente ”
„Attribute”

Superelemente

Die folgenden Elementtypen enthalten `ClusterNodeType`-Subelemente:

- `ClusterDomainType`

XML-Schemadefinition

```
<xs:complexType name='ClusterNodeType'>
  <xs:attribute name='clusterNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

Keine

Attribute

clusterNodeName (erforderlich)
Geben Sie den Namen des Clusterdomänenknotens an.

FailoverPolicyType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element `FailoverPolicyType` gibt die *Richtlinie für Funktionsübernahme* an, die der Cluster-Manager in der Clusterdomäne anwenden soll.

„Superelemente”
„XML-Schemadefinition ”
„Subelemente” auf Seite 125
„Gültige Werte ” auf Seite 125

Superelemente

Die folgenden Elementtypen enthalten `InterfaceType`-Subelemente:

- `DB2ClusterType`

XML-Schemadefinition

```
<xs:complexType name='FailoverPolicyType'>
  <xs:choice>
    <xs:element name='RoundRobin'
                type='xs:string'
                minOccurs='0' />
    <xs:element name='Mutual'
                type='xs:string'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='NPlusM'
                type='xs:string'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='LocalRestart'
                type='xs:string'
                fixed=''/>
    <xs:element name='HADRFailover'
                type='xs:string'
```

```
                fixed=''/>
    <xs:element name='Custom'
                type='xs:string'
                minOccurs='0' />
  </xs:choice>
</xs:complexType>
```

Subelemente

Keine

Gültige Werte

Geben Sie dem Cluster-Manager eine der folgenden Optionen als den Funktionsübernahmetyp an, der im Falle von Störungen in der Clusterdomäne angewendet werden soll:

Eine *Richtlinie für Funktionsübernahme* gibt an, wie ein Cluster-Manager sich verhalten soll, wenn ein Clusterelement, wie z. B. eine Netzschnittstellenkarte oder ein Datenbankserver, ausfällt. Im Allgemeinen wird ein Cluster-Manager die Workload in diesem Fall von dem ausgefallenen Element auf ein alternatives Element übertragen, das zuvor für den Cluster-Manager als geeigneter Ersatz für das ausgefallene Element definiert wurde. Diese Übertragung der Workload von einem ausgefallenen Element auf ein anderes Element wird als *Funktionsübernahme* bezeichnet.

RoundRobin
Die *Funktionsübernahmerichtlinie 'Umlauf'* (RoundRobin) sieht vor, dass der Datenbankmanager bei einem Ausfall eines einzigen Computers (auch *Clusterdomänenknoten* oder kurz *Knoten* genannt) in der Clusterdomäne die Workload des ausgefallenen Clusterdomänenknotens auf einem beliebigen anderen Knoten in der Clusterdomäne erneut startet.

Mutual
Für die Konfiguration der *Funktionsübernahmerichtlinie 'Gegenseitige Übernahme'* (Mutual) müssen Sie zwei Computer (auch *Clusterdomänenknoten* oder einfach *Knoten* genannt) in der Clusterdomäne als ein Systempaar zuordnen. Tritt bei einem dieser beiden Knoten ein Ausfall auf, wird die Workload der Datenbankpartitionen des ausgefallenen Knotens von dem anderen Knoten übernommen. Eine gegenseitige Übernahme ist nur möglich, wenn Sie über mehrere Datenbankpartitionen verfügen.

NPlusM
Wenn Sie die *Funktionsübernahmerichtlinie 'M+N'* verwenden, wird bei einem Ausfall eines Computers in der Clusterdomäne (auch *Clusterdomänenknoten* oder kurz *Knoten* genannt) die Workload der Datenbankpartitionen auf dem ausgefallenen Knoten von einem beliebigen anderen Knoten in der Clusterdomäne übernommen. Wenn die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten aktiviert ist, wird der letzte fehlgeschlagene Knoten zum Bereitschaftsknoten, sobald dieser fehlgeschlagene Knoten wieder in den Online-Status gesetzt wird. Die HA-Funktionsübernahme mit wechselndem Bereitschaftsknoten für die Funktionsübernahmerichtlinie M+N wird nur unterstützt, wenn M=1. Eine M+N-Übernahme ist nur möglich, wenn Sie über mehrere Datenbankpartitionen verfügen.

LocalRestart
Wenn Sie die *Funktionsübernahmerichtlinie 'Lokaler Neustart'* (LocalRestart) verwenden, startet der Datenbankmanager bei einem Ausfall auf einem

Computer (auch *Clusterdomänenknoten* oder kurz *Knoten* genannt) in der Clusterdomäne die (lokal) vorhandene Datenbank auf dem Knoten, der ausgefallen ist, neu.

HADRFailover
Wenn Sie die *Funktionsübernahmerichtlinie 'HADR'* (HADRFailover) konfigurieren, ermöglichen Sie der DB2 HADR-Funktion (High Availability Disaster Recovery) die Verwaltung der Funktionsübernahme. Fällt eine HADR-Primärdatenbank aus, versetzt der Datenbankmanager die Workload von der ausgefallenen Datenbank auf eine HADR-Bereitschaftsdatenbank.

Custom
Wenn Sie die *Funktionsübernahmerichtlinie 'Benutzerdefiniert'* (Custom) konfigurieren, erstellen Sie eine Liste von Computern (auch *Clusterdomänenknoten* oder kurz *Knoten* genannt) in der Clusterdomäne, die der Datenbankmanager für eine Funktionsübernahme nutzen kann. Fällt ein Knoten in der Clusterdomäne aus, versetzt der Datenbankmanager die Workload vom ausgefallenen Knoten auf einen in der von Ihnen angegebenen Liste enthaltenen Knoten.

DB2PartitionSetType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element `DB2PartitionSetType` enthält Informationen zu Datenbankpartitionen. Eine Verwendung des Elements `DB2PartitionSetType` ist nur in Umgebungen mit partitionierten Datenbanken möglich.

„Superelemente"
„XML-Schemadefinition "
„Subelemente"
„Attribute" auf Seite 127

Superelemente

`InterfaceType` ist ein Subelement von:

- `PhysicalNetworkType`

XML-Schemadefinition

```
<xs:complexType name='DB2PartitionSetType'>
  <xs:sequence>
    <xs:element name='DB2Partition'
                type='DB2PartitionType'
                maxOccurs='unbounded'/>
  </xs:sequence>
</xs:complexType>
```

Subelemente

DB2Partition

Typ: `DB2PartitionType`

Das Element `DB2PartitionType` gibt eine Datenbankpartition sowie die DB2-Datenbankmanagerinstanz, zu der die Datenbankpartition gehört, und die Nummer der Datenbankpartition an.

Regeln zur Häufigkeit:
Sie müssen mindestens ein Element `DB2Partition` im Element `DB2PartitionSetType` angeben.

Attribute

Keine

DB2PartitionType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element DB2PartitionType gibt eine Datenbankpartition sowie die DB2-Datenbankmanagerinstanz, zu der die Datenbankpartition gehört, und die Nummer der Datenbankpartition an.

„Superelemente"
„XML-Schemadefinition "
„Subelemente "
„Attribute " auf Seite 128

Superelemente

InterfaceType ist ein Subelement von:

- DB2PartitionSetType

XML-Schemadefinition

```
<xs:complexType name='DB2PartitionType'>
 <xs:sequence>
    <xs:element name='VirtualIPAddress'
                type='IPAddressType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='Mount'
                type='MountType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='HADRDB'
                type='HADRDBType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='MutualPair'
                type='MutualPolicyType'
                minOccurs='0'
                maxOccurs='1'/>
    <xs:element name='NPlusMNode'
                type='NPlusMPolicyType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='CustomNode'
                type='CustomPolicyType'
                minOccurs='0'
                maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='instanceName'     type='xs:string'  use='required'/>
  <xs:attribute name='dbpartitionnum'   type='xs:integer' use='required'/>
</xs:complexType>
```

Subelemente

VirtualIPAddress

Typ: IPAddressType

Das Element IPAddressType enthält alle Einzelangaben zu einer IP-Adresse, z. B. die *Basisadresse*, die *Teilnetzmaske* und den Namen des Netzes, zu dem die IP-Adresse gehört.

Sie können die Angabe von VirtualIPAddress auslassen oder eine unbegrenzte Anzahl von VirtualIPAddress-Elementen in das Element DB2PartitionType einfügen.

Mount

Typ: MountType

Das Element MountType enthält Informationen zu einem *Mountpunkt*, z. B. dem Dateipfad, der die Position der angehängten Dateien angibt.

Sie können die Angabe von Mount auslassen oder eine unbegrenzte Anzahl von Mount-Elementen in das Element DB2PartitionType einfügen.

HADRDB

Typ: HADRDBType

Das Element HADRDBType enthält eine Liste mit Datenbankpaaren, die aus einer HADR-Primärdatenbank (High Availability Disaster Recovery) und einer HADR-Bereitschaftsdatenbank bestehen.

Sie können die Angabe von HADRDB auslassen oder eine unbegrenzte Anzahl von HADRDB-Elementen in das Element DB2PartitionType einfügen.

MutualPair

Typ: MutualPolicyType

Das Element MutualPolicyType enthält Informationen zu einem Paar von Clusterdomänenknoten, die für eine gegenseitige Funktionsübernahme definiert sind.

Sie können die Angabe von MutualPair auslassen oder eine unbegrenzte Anzahl von MutualPair-Elementen in das Element DB2PartitionType einfügen.

NPlusMNode

Typ: NPlusMPolicyType

Sie können die Angabe von NPlusMNode auslassen oder eine unbegrenzte Anzahl von NPlusMNode-Elementen in das Element DB2PartitionType einfügen.

CustomNode

Typ: CustomPolicyType

Sie können auf die Angabe von CustomNode verzichten oder eine unbegrenzte Anzahl von CustomNode-Elementen in das Element DB2PartitionType einfügen.

Attribute

instanceName (erforderlich)

Geben Sie im Attribut instanceName die DB2-Datenbankmanagerinstanz an, der dieses Element DB2PartitionType zugeordnet ist.

dbpartitionnum (erforderlich)

Geben Sie im Attribut dbpartitionnum die Datenbankpartitionsnummer an, die die Datenbankpartition eindeutig kennzeichnet (die Nummer dbpartitionnum wird z. B. in der Datei db2nodes.cfg angegeben).

MountType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element MountType enthält Informationen zu einem *Mountpunkt*, z. B. dem Dateipfad, der die Position der angehängten Dateien angibt.

„Superelemente”
„XML-Schemadefinition ”
„Subelemente ”
„Attribute”

Superelemente

Die folgenden Elementtypen enthalten MountType-Subelemente:

- DB2PartitionType

XML-Schemadefinition

```
<xs:complexType name='MountType'>
  <xs:attribute name='filesystemPath' type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

Keine

Attribute

filesystemPath (erforderlich)
Geben Sie den Pfad an, der dem Mountpunkt beim Anhängen des Dateisystems zugeordnet wurde.

MutualPolicyType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element MutualPolicyType enthält Informationen zu einem Paar von Clusterdomänenknoten, die für eine gegenseitige Funktionsübernahme definiert sind.

„Superelement”
„XML-Schemadefinition ”
„Subelemente ”
„Attribute”

Superelement

Die folgenden Elementtypen enthalten MutualPolicyType-Subelemente:

- DB2PartitionType

XML-Schemadefinition

```
<xs:complexType name='MutualPolicyType'>
  <xs:attribute name='systemPairNode1' type='xs:string' use='required'/>
  <xs:attribute name='systemPairNode2' type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

Keine

Attribute

systemPairNode1 (erforderlich)
Geben Sie für das Attribut systemPairNode1 den Namen eines Clusterdomänenknotens an, der für den im Attribut systemPairNode2 angegebenen Clusterdomänenknoten bei einer Funktionsübernahme verfügbar ist.

systemPairNode2 (erforderlich)
Geben Sie für das Attribut `systemPairNode2` den Namen eines Clusterdomänenknotens an, der für den im Attribut `systemPairNode1` angegebenen Clusterdomänenknoten bei einer Funktionsübernahme verfügbar ist.

NPlusMPolicyType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

'NPlusMPolicy' gibt an, dass beim Ausfall eines Computers in einer Clusterdomäne eine Funktionsübernahme der Datenbankpartitionen auf dem fehlgeschlagenen Knoten durch einen beliebigen anderen verfügbaren Knoten in derselben Clusterdomäne erfolgt. Ein XML-Schema definiert die Konfigurationen, die dieser HADR-Richtlinie zugeordnet sind.

„Superelemente”
„XML-Schemadefinition ”
„Subelemente ”
„Attribute”

Superelemente

Die folgenden Elementtypen enthalten `NPlusMPolicyType`-Subelemente:

- `DB2PartitionType`

XML-Schemadefinition

```
<xs:complexType name='NPlusMPolicyType'>
  <xs:attribute name='standbyNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

Keine

Attribute

standbyNodeName (erforderlich)
Geben Sie im Element `standbyNodeName` den Namen eines Clusterdomänenknotens an, der bei einer Funktionsübernahme für die Partition verfügbar ist, die das Element `NPlusMPolicyType` enthält.

CustomPolicyType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Die XML-Schemadefinition 'CustomPolicyType' definiert Konfigurationseinstellungen für eine angepasste HADR-Richtlinie. In diesem Schema können Sie die Knoten definieren, die bei einer Funktionsübernahme standardmäßig verwendet werden.

„Superelemente” auf Seite 131
„XML-Schemadefinition ” auf Seite 131
„Subelemente” auf Seite 131
„Attribute” auf Seite 131

Superelemente

Die folgenden Elementtypen enthalten CustomPolicyType-Subelemente:

- DB2PartitionType

XML-Schemadefinition

```
<xs:complexType name='NPlusMPolicyType'>
  <xs:attribute name='standbyNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

Keine

Attribute

customNodeName (erforderlich)
Geben Sie im Element customNodeName den Namen eines Clusterdomänenknotens an, der bei einer Funktionsübernahme für die Partition verfügbar ist, die das Element CustomPolicyType enthält.

HADRDBType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element HADRDBType enthält eine Liste mit Datenbankpaaren, die aus einer HADR-Primärdatenbank (High Availability Disaster Recovery) und einer HADR-Bereitschaftsdatenbank bestehen.

„Superelemente"
„XML-Schemadefinition "
„Subelemente "
„Attribute" auf Seite 132
„Hinweise zur Verwendung" auf Seite 132
„Einschränkungen" auf Seite 132

Superelemente

Die folgenden Elementtypen enthalten HADRDBType-Subelemente:

- DB2ClusterType
- DB2PartitionType

XML-Schemadefinition

```
<xs:complexType name='HADRDBType'>
 <xs:sequence>
   <xs:element name='HADRDB' type='HADRDBDefn'  minOccurs='1'  maxOccurs='1'/>
   <xs:element name='VirtualIPAddress' type='IPAddressType' minOccurs='0' maxOccurs='1'/>
 </xs:sequence>
</xs:complexType>
```

Subelemente

VirtualIPAddress

Typ: IPAddressType

Das Element IPAddressType enthält alle Einzelangaben zu einer IP-Adresse, z. B. die *Basisadresse*, die *Teilnetzmaske* und den Namen des Netzes, zu dem die IP-Adresse gehört.

Regeln zur Häufigkeit:
Sie können keine oder beliebig viele Elemente `VirtualIPAddress` in das Element `HADRDBType` aufnehmen.

HADRDB

Typ: `HADRDBDefn`

Das Element `HADRDBDefn` enthält Informationen zu einem HADR-Datenbankpaar (High Availability Disaster Recovery), das aus einer HADR-Primärdatenbank und einer HADR-Bereichschaftsdatenbank besteht.

Regeln zur Häufigkeit:
Sie können das Element `VirtualIPAddress` beliebig oft in das Element `HADRDBType` aufnehmen.

Attribute

Keine

Hinweise zur Verwendung

Wenn Sie ein Element `HADRDBType` in die Spezifikation für eine bestimmte Clusterdomäne aufnehmen, müssen Sie in derselben Clusterdomänenspezifikation ebenfalls ein Element `FailoverPolicy` einfügen, das `HADRFailover` angibt.

Einschränkungen

Sie können das Element `HADRDBType` nicht in einer Umgebung mit partitionierten Datenbanken verwenden.

Das folgende Beispiel zeigt ein `HADRDBType`-Element:

```
<HADRDBSet>
 <HADRDB databaseName="HADRDB" localInstance="db2inst1"
  remoteInstance="db2inst1" localHost="linux01" remoteHost="linux02" />
 <VirtualIPAddress baseAddress="9.26.124.22" subnetMask="255.255.245.0"
  networkName="db2_public_network_0"/>
</HADRDBSet>
```

HADRDBDefn - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element `HADRDBDefn` enthält Informationen zu einem HADR-Datenbankpaar (High Availability Disaster Recovery), das aus einer HADR-Primärdatenbank und einer HADR-Bereichschaftsdatenbank besteht.

„Superelemente"
„XML-Schemadefinition " auf Seite 133
„Subelemente " auf Seite 133
„Attribute" auf Seite 133

Superelemente

Die folgenden Elementtypen enthalten `HADRDBDefn`-Subelemente:

- `HADRDBType`

XML-Schemadefinition

```
<xs:complexType name='HADRDBDefn'>
  <xs:attribute name='databaseName'      type='xs:string' use='required'/>
  <xs:attribute name='localInstance'     type='xs:string' use='required'/>
  <xs:attribute name='remoteInstance'    type='xs:string' use='required'/>
  <xs:attribute name='localHost'         type='xs:string' use='required'/>
  <xs:attribute name='remoteHost'        type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

Keine

Attribute

databaseName (erforderlich)
Geben Sie den Namen der HADR-Datenbank an.

localInstance (erforderlich)
`localInstance` gibt die Datenbankmanagerinstanz der HADR-Primärdatenbank an.

remoteInstance (erforderlich)
`remoteInstance` gibt die Datenbankmanagerinstanz der HADR-Bereitschaftsdatenbank an.

localHost (erforderlich)
`localHost` gibt den Hostnamen des Clusterdomänenknotens an, auf dem sich die HADR-Primärdatenbank befindet.

remoteHost (erforderlich)
`remoteHost` gibt den Hostnamen des Clusterdomänenknotens an, auf dem sich die HADR-Bereitschaftsdatenbank befindet.

HADBType - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element `HADBType` enthält eine Liste der Datenbanken, die zu der Clusterdomäne gehören und eine hohe Verfügbarkeit aufweisen sollen.

„Superelemente”
„XML-Schemadefinition ”
„Subelemente ” auf Seite 134
„Attribute” auf Seite 134

Superelemente

Die folgenden Elementtypen enthalten `HADBType`-Subelemente:

- `DB2ClusterType`

XML-Schemadefinition

```
<xs:complexType name='HADBType'>
  <xs:sequence>
    <xs:element name='HADB'         type='HADBDefn' maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='instanceName' type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

HADB

Typ: HADBDefn

Das Element HADBDefn beschreibt eine Datenbank, die als Datenbank mit hoher Verfügbarkeit in die Clusterdomäne aufgenommen werden soll.

Regeln zur Häufigkeit:
Fügen Sie mindestens ein Element HADB im Element HADBType hinzu.

Attribute

instanceName (erforderlich)
Geben Sie im Attribut instanceName die DB2-Datenbankmanagerinstanz an, zu der die in den Elementen HADB angegebenen Datenbanken gehören.

HADBDefn - XML-Schemadefinition für Eingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Das Element HADBDefn beschreibt eine Datenbank, die als Datenbank mit hoher Verfügbarkeit in die Clusterdomäne aufgenommen werden soll.

„Superelemente"
„XML-Schemadefinition "
„Subelemente "
„Attribute"

Superelemente

HADBDefn ist ein Subelement von:

- HADRDBType

XML-Schemadefinition

```
<xs:complexType name='HADBDefn'>
  <xs:attribute name='databaseName' type='xs:string' use='required'/>
</xs:complexType>
```

Subelemente

Keine

Attribute

databaseName (erforderlich)
Geben Sie einen einzigen Datenbanknamen im Attribut databaseName an.

XML-Beispieleingabedateien für DB2 High Availability Instance Configuration Utility (db2haicu):

Im Verzeichnis sqllib befindet sich im Unterverzeichnis samples eine Gruppe von XML-Beispieleingabedateien, die Sie modifizieren und mit **db2haicu** zum Konfigurieren der Clusterumgebung verwenden können.

db2ha_sample_sharedstorage_mutual.xml:

Die Datei db2ha_sample_sharedstorage_mutual.xml ist ein Beispiel für eine XML-Eingabedatei, die an DB2 High Availability Instance Configuration Utility (db2haicu) übergeben wird, um eine neue *Clusterdomaine* anzugeben. db2ha_sample_sharedstorage_mutual.xml befindet sich im Verzeichnis sqllib/samples/ha/xml.

Merkmale

Die Beispieldatei db2ha_sample_sharedstorage_mutual.xml veranschaulicht, wie Sie **db2haicu** mit einer XML-Eingabedatei verwenden können, um eine Clusterdomäne mit den folgenden Details zu definieren:

- Quorumeinheit: Netz
- Anzahl der Computer im Cluster (Clusterdomänenknoten): 2
- Richtlinie für Funktionsübernahme: Gegenseitige Übernahme
- Datenbankpartitionen: 1
- Virtuelle IP-Adressen (Service-IP-Adressen): 1
- Gemeinsame Mountpunkte für Funktionsübernahme: 1

XML-Quellencode

```
<!-- ================================================================== -->
<!-- = Verwenden Sie die XML-Schemadefinition db2ha.xsd für          = -->
<!-- = db2haicu, und geben Sie IBM Tivoli System                     = -->
<!-- = Automation for Multiplatforms (SA MP) Base Component          = -->
<!-- = als Cluster-Manager an.                                       = -->
<!-- ================================================================== -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
            clusterManagerName="TSA"
            version="1.0">

  <!-- ================================================================== -->
  <!-- = Erstellen Sie eine Clusterdomäne mit dem Namen 'db2HAdomain'. = -->
  <!-- ================================================================== -->
  <ClusterDomain domainName="db2HAdomain">

    <!-- ================================================================== -->
    <!-- = Geben Sie eine Netzquorumeinheit (IP-Adresse: 19.126.4.5)   = -->
    <!-- = an. Die IP muss jederzeit von allen Clusterdomänenknoten    = -->
    <!-- = mit Ping überprüft werden können.                           = -->
    <!-- ================================================================== -->
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

    <!-- ================================================================== -->
    <!-- = Erstellen Sie ein Netz mit dem Namen 'db2_public_network_0' = -->
    <!-- = mit einem IP-Netzprotokoll.                                 = -->
    <!-- = Dieses Netz enthält 2 Computer: 'hasys01' und 'hasys02'.    = -->
    <!-- = Jeder Computer hat eine einzige Netzschnittstellenkarte     = -->
    <!-- = (NIC) mit der Bezeichnung 'eth0'.                           = -->
    <!-- = Die IP-Adresse der NIC auf 'hasys01' ist '19.126.52.139'.   = -->
    <!-- = Die IP-Adresse der NIC auf 'hasys02' ist '19.126.52.140'.   = -->
    <!-- ================================================================== -->
    <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                     physicalNetworkProtocol="ip">

      <Interface interfaceName="eth0" clusterNodeName="hasys01">
        <IPAddress baseAddress="19.126.52.139"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

      <Interface interfaceName="eth0" clusterNodeName="hasys02">
```

```
        <IPAddress baseAddress="19.126.52.140"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

    </PhysicalNetwork>

    <!-- ============================================================= -->
    <!-- = Listen Sie die Computer (Knoten) in der Clusterdomäne auf.  = -->
    <!-- ============================================================= -->
    <ClusterNode clusterNodeName="hasys01"/>
    <ClusterNode clusterNodeName="hasys02"/>

  </ClusterDomain>


  <!-- ================================================================= -->
  <!-- = Die Richtlinie für Funktionsübernahme gibt die Reihenfolge    = -->
  <!-- = für die Funktionsübernahme bei den Clusterdomänenknoten an.   = -->
  <!-- ================================================================= -->
  <FailoverPolicy>
     <Mutual />
  </FailoverPolicy>


  <!-- ================================================================= -->
  <!-- = Geben Sie alle Details zu der Datenbankpartition an.          = -->
  <!-- ================================================================= -->
  <DB2PartitionSet>

    <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
       <VirtualIPAddress baseAddress="19.126.52.222"
                         subnetMask="255.255.255.0"
                         networkName="db2_public_network_0"/>
       <Mount filesystemPath="/home/db2inst1"/>
       <MutualPair systemPairNode1="hasys01" systemPairNode2="hasys02" />
    </DB2Partition>

  </DB2PartitionSet>

</DB2Cluster>
```

db2ha_sample_DPF_mutual.xml:

Die Datei db2ha_sample_DPF_mutual.xml ist ein Beispiel für eine XML-Eingabedatei, die an DB2 High Availability Instance Configuration Utility (db2haicu) übergeben wird, um eine neue *Clusterdomaine* anzugeben. db2ha_sample_DPF_mutual.xml befindet sich im Verzeichnis sqllib/samples/ha/xml.

Merkmale

Die Beispieldatei db2ha_sample_DPF_mutual.xml veranschaulicht, wie Sie **db2haicu** mit einer XML-Eingabedatei verwenden können, um eine Clusterdomäne mit den folgenden Details zu definieren:

- Quorumeinheit: Netz
- Anzahl der Computer im Cluster (Clusterdomänenknoten): 4
- Richtlinie für Funktionsübernahme: Gegenseitige Übernahme
- Datenbankpartitionen: 2
- Virtuelle IP-Adressen (Service-IP-Adressen): 1
- Gemeinsame Mountpunkte für Funktionsübernahme: 2
- Für hohe Verfügbarkeit konfigurierte Datenbanken: 2

XML-Quellencode

```
<!-- ================================================================= -->
<!-- = Verwenden Sie die XML-Schemadefinition db2ha.xsd für          = -->
<!-- = db2haicu, und geben Sie IBM Tivoli System                     = -->
<!-- = Automation for Multiplatforms (SA MP) Base Component          = -->
<!-- = als Cluster-Manager an.                                       = -->
<!-- ================================================================= -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
            clusterManagerName="TSA"
            version="1.0">

  <!-- ================================================================= -->
  <!-- = Erstellen Sie eine Clusterdomäne mit dem Namen 'db2HAdomain'. = -->
  <!-- ================================================================= -->
  <ClusterDomain domainName="db2HAdomain">

    <!-- ================================================================= -->
    <!-- = Geben Sie eine Netzquorumeinheit (IP-Adresse: 19.126.4.5)   = -->
    <!-- = an. Die IP muss jederzeit von allen Clusterdomänenknoten    = -->
    <!-- = mit Ping überprüft werden können.                           = -->
    <!-- ================================================================= -->
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

    <!-- ================================================================= -->
    <!-- = Erstellen Sie ein Netz mit dem Namen 'db2_public_network_0' = -->
    <!-- = mit einem IP-Netzprotokoll.                                 = -->
    <!-- = Dieses Netz enthält vier Computer: hasys01, hasys02,        = -->
    <!-- = hasys03 und hasys04.                                        = -->
    <!-- = Jeder Computer hat eine Netzschnittstellenkarte (eth0).     = -->
    <!-- = Die IP-Adresse von 'eth0' auf 'hasys01' ist '19.126.124.30'.= -->
    <!-- = Die IP-Adresse von 'eth0' auf 'hasys02' ist '19.126.124.31'.= -->
    <!-- = Die IP-Adresse von 'eth0' auf 'hasys03' ist '19.126.124.32'.= -->
    <!-- = Die IP-Adresse von 'eth0' auf 'hasys04' ist '19.126.124.33'.= -->
    <!-- ================================================================= -->
    <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                     physicalNetworkProtocol="ip">

      <Interface interfaceName="eth0" clusterNodeName="hasys01">
        <IPAddress baseAddress="19.126.124.30"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

      <Interface interfaceName="eth0" clusterNodeName="hasys02">
        <IPAddress baseAddress="19.126.124.31"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

      <Interface interfaceName="eth0" clusterNodeName="hasys03">
        <IPAddress baseAddress="19.126.124.32"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

      <Interface interfaceName="eth0" clusterNodeName="hasys04">
        <IPAddress baseAddress="19.126.124.33"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

    </PhysicalNetwork>

    <!-- ================================================================= -->
    <!-- = Erstellen Sie ein Netz mit dem Namen 'db2_private_network_0'= -->
    <!-- = mit einem IP-Netzprotokoll.                                 = -->
```

```
  <!-- = Dieses Netz enthält vier Computer: hasys01, hasys02,        = -->
  <!-- = hasys03 und hasys04 (wie 'db2_public_network_0'.)           = -->
  <!-- = Zusätzlich zu 'eth0' verfügt jeder Computer über eine       = -->
  <!-- = Netzschnittstellenkarte mit dem Namen 'eth1'.               = -->
  <!-- = Die IP-Adresse von 'eth1' auf 'hasys01' ist 192.168.23.101. = -->
  <!-- = Die IP-Adresse von 'eth1' auf 'hasys02' ist 192.168.23.102. = -->
  <!-- = Die IP-Adresse von 'eth1' auf 'hasys03' ist 192.168.23.103. = -->
  <!-- = Die IP-Adresse von 'eth1' auf 'hasys04' ist 192.168.23.104. = -->
  <!-- ================================================================ -->
  <PhysicalNetwork physicalNetworkName="db2_private_network_0"
                   physicalNetworkProtocol="ip">

    <Interface interfaceName="eth1" clusterNodeName="hasys01">
      <IPAddress baseAddress="192.168.23.101"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys02">
      <IPAddress baseAddress="192.168.23.102"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys03">
      <IPAddress baseAddress="192.168.23.103"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys04">
      <IPAddress baseAddress="192.168.23.104"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ================================================================ -->
  <!-- = Listen Sie die Computer (Knoten) in der Clusterdomäne auf.  = -->
  <!-- ================================================================ -->
  <ClusterNode clusterNodeName="hasys01"/>
  <ClusterNode clusterNodeName="hasys02"/>
  <ClusterNode clusterNodeName="hasys03"/>
  <ClusterNode clusterNodeName="hasys04"/>

</ClusterDomain>


<!-- ================================================================ -->
<!-- = Die Richtlinie für Funktionsübernahme gibt die Reihenfolge    = -->
<!-- = für die Funktionsübernahme bei den Clusterdomänenknoten an.   = -->
<!-- ================================================================ -->
<FailoverPolicy>
   <Mutual />
</FailoverPolicy>


<!-- ================================================================ -->
<!-- = Geben Sie alle Details zu den Datenbankpartitionen an.        = -->
<!-- ================================================================ -->
<DB2PartitionSet>

  <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
     <VirtualIPAddress baseAddress="19.126.124.251"
                       subnetMask="255.255.255.0"
                       networkName="db2_public_network_0"/>
```

```
      <Mount filesystemPath="/hafs/db2inst1/NODE0000"/>
      <MutualPair systemPairNode1="hasys01" systemPairNode2="hasys02" />
    </DB2Partition>

    <DB2Partition dbpartitionnum="1" instanceName="db2inst1">
      <Mount filesystemPath="/hafs/db2inst1/NODE0001"/>
      <MutualPair systemPairNode1="hasys02" systemPairNode2="hasys01" />
    </DB2Partition>

    <DB2Partition dbpartitionnum="2" instanceName="db2inst1">
      <Mount filesystemPath="/hafs/db2inst1/NODE0002"/>
      <MutualPair systemPairNode1="hasys03" systemPairNode2="hasys04" />
    </DB2Partition>

    <DB2Partition dbpartitionnum="3" instanceName="db2inst1">
      <Mount filesystemPath="/hafs/db2inst1/NODE0003"/>
      <MutualPair systemPairNode1="hasys04" systemPairNode2="hasys03" />
    </DB2Partition>

  </DB2PartitionSet>


  <!-- ================================================================ -->
  <!-- = Liste der als hochverfügbar zu konfigurierenden Datenbanken   = -->
    <!-- ================================================================ -->
  <HADBSet instanceName="db2inst1">
    <HADB databaseName = "SAMPLE" />
    <HADB databaseName = "MYDB" />
  </HADBSet>


</DB2Cluster>
```

db2ha_sample_DPF_NPlusM.xml:

Die Datei db2ha_sample_DPF_NPlusM.xml ist ein Beispiel für eine XML-Eingabedatei, die an DB2 High Availability Instance Configuration Utility (db2haicu) übergeben wird, um eine neue *Clusterdomaine* anzugeben. db2ha_sample_DPF_NPlusM.xml befindet sich im Verzeichnis sqllib/samples/ha/xml.

Merkmale

Die Beispieldatei db2ha_sample_DPF_NPlusM.xml veranschaulicht, wie Sie **db2haicu** mit einer XML-Eingabedatei verwenden können, um eine Clusterdomäne mit den folgenden Details zu definieren:

- Quorumeinheit: Netz
- Anzahl der Computer im Cluster (Clusterdomänenknoten): 4
- Richtlinie für Funktionsübernahme: M+N
- Datenbankpartitionen: 2
- Virtuelle IP-Adressen (Service): 1
- Gemeinsame Mountpunkte für Funktionsübernahme: 4

XML-Quellencode

```
<!-- ================================================================ -->
<!-- = Verwenden Sie die XML-Schemadefinition db2ha.xsd für          = -->
<!-- = db2haicu, und geben Sie IBM Tivoli System                     = -->
<!-- = Automation for Multiplatforms (SA MP) Base Component          = -->
<!-- = als Cluster-Manager an.                                       = -->
<!-- ================================================================ -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
```

```
          clusterManagerName="TSA"
          version="1.0">

<!-- ================================================================= -->
<!-- = Erstellen Sie eine Clusterdomäne mit dem Namen 'db2HAdomain'. = -->
<!-- ================================================================= -->
<ClusterDomain domainName="db2HAdomain">

  <!-- ================================================================= -->
  <!-- = Geben Sie eine Netzquorumeinheit (IP-Adresse: 19.126.4.5)   = -->
  <!-- = an. Die IP muss jederzeit von allen Clusterdomänenknoten    = -->
  <!-- = mit Ping überprüft werden können.                           = -->
  <!-- ================================================================= -->
  <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

  <!-- ================================================================= -->
  <!-- = Erstellen Sie ein Netz mit dem Namen 'db2_public_network_0' = -->
  <!-- = mit einem IP-Netzprotokoll.                                 = -->
  <!-- = Dieses Netz enthält vier Computer: hasys01, hasys02,        = -->
  <!-- = hasys03 und hasys04.                                        = -->
  <!-- = Jeder Computer hat eine Netzschnittstellenkarte (eth0).     = -->
  <!-- = Die IP-Adresse von 'eth0' auf 'hasys01' ist '19.126.124.30'.= -->
  <!-- = Die IP-Adresse von 'eth0' auf 'hasys02' ist '19.126.124.31'.= -->
  <!-- = Die IP-Adresse von 'eth0' auf 'hasys03' ist '19.126.124.32'.= -->
  <!-- = Die IP-Adresse von 'eth0' auf 'hasys04' ist '19.126.124.33'.= -->
  <!-- ================================================================= -->
  <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                   physicalNetworkProtocol="ip">

    <Interface interfaceName="eth0" clusterNodeName="hasys01">
      <IPAddress baseAddress="19.126.124.30"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys02">
      <IPAddress baseAddress="19.126.124.31"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys03">
      <IPAddress baseAddress="19.126.124.32"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys04">
      <IPAddress baseAddress="19.126.124.33"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ================================================================= -->
  <!-- = Erstellen Sie ein Netz mit dem Namen 'db2_private_network_0'= -->
  <!-- = mit einem IP-Netzprotokoll.                                 = -->
  <!-- = Dieses Netz enthält vier Computer: hasys01, hasys02,        = -->
  <!-- = hasys03 und hasys04 (wie 'db2_public_network_0'.)           = -->
  <!-- = Zusätzlich zu 'eth0' verfügt jeder Computer über eine       = -->
  <!-- = Netzschnittstellenkarte mit dem Namen 'eth1'.               = -->
  <!-- = Die IP-Adresse von 'eth1' auf 'hasys01' ist 192.168.23.101. = -->
  <!-- = Die IP-Adresse von 'eth1' auf 'hasys02' ist 192.168.23.102. = -->
  <!-- = Die IP-Adresse von 'eth1' auf 'hasys03' ist 192.168.23.103. = -->
  <!-- = Die IP-Adresse von 'eth1' auf 'hasys04' ist 192.168.23.104. = -->
  <!-- ================================================================= -->
```

```
  <PhysicalNetwork physicalNetworkName="db2_private_network_0"
                   physicalNetworkProtocol="ip">

    <Interface interfaceName="eth1" clusterNodeName="hasys01">
      <IPAddress baseAddress="192.168.23.101"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys02">
      <IPAddress baseAddress="192.168.23.102"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys03">
      <IPAddress baseAddress="192.168.23.103"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

    <Interface interfaceName="eth1" clusterNodeName="hasys04">
      <IPAddress baseAddress="192.168.23.104"
                 subnetMask="255.255.255.0"
                 networkName="db2_private_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ================================================================ -->
  <!-- = Listen Sie die Computer (Knoten) in der Clusterdomäne auf.  = -->
  <!-- ================================================================ -->
  <ClusterNode clusterNodeName="hasys01"/>
  <ClusterNode clusterNodeName="hasys02"/>
  <ClusterNode clusterNodeName="hasys03"/>
  <ClusterNode clusterNodeName="hasys04"/>

</ClusterDomain>


<!-- ================================================================ -->
<!-- = Die Richtlinie für Funktionsübernahme gibt die Reihenfolge    = -->
<!-- = für die Funktionsübernahme bei den Clusterdomänenknoten an.   = -->
<!-- ================================================================ -->
<FailoverPolicy>
   <NPlusM />
</FailoverPolicy>


<!-- ================================================================ -->
<!-- = Geben Sie alle Details zu den Datenbankpartitionen an.        = -->
<!-- ================================================================ -->
<DB2PartitionSet>

  <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
     <VirtualIPAddress baseAddress="19.126.124.250"
                       subnetMask="255.255.255.0"
                       networkName="db2_public_network_0"/>
     <Mount filesystemPath="/ha_dpf1/db2inst1/NODE0000"/>
     <Mount filesystemPath="/hafs/NODE0000"/>
     <NPlusMNode standbyNodeName="hasys03" />
  </DB2Partition>

  <DB2Partition dbpartitionnum="1" instanceName="db2inst1">
     <Mount filesystemPath="/ha_dpf1/db2inst1/NODE0001"/>
     <Mount filesystemPath="/hafs/NODE0001"/>
     <NPlusMNode standbyNodeName="hasys04" />
```

```
    </DB2Partition>

  </DB2PartitionSet>

</DB2Cluster>
```

db2ha_sample_HADR.xml:

Die Datei db2ha_sample_DPF_HADR.xml ist ein Beispiel für eine XML-Eingabedatei, die an DB2 High Availability Instance Configuration Utility (db2haicu) übergeben wird, um eine neue *Clusterdomaine* anzugeben. db2ha_sample_HADR.xml befindet sich im Verzeichnis sqllib/samples/ha/xml.

Merkmale

Die Beispieldatei db2ha_sample_HADR.xml veranschaulicht, wie Sie **db2haicu** mit einer XML-Eingabedatei verwenden können, um eine Clusterdomäne mit den folgenden Details zu definieren:

- Quorumeinheit: Netz
- Anzahl der Computer im Cluster (Clusterdomänenknoten): 2
- Richtlinie für Funktionsübernahme: HADR
- Datenbankpartitionen: 1
- Virtuelle IP-Adressen (Service-IP-Adressen): Keine
- Gemeinsame Mountpunkte für Funktionsübernahme: Keine

XML-Quellencode

```
<!-- ================================================================== -->
<!-- = DB2-Konfigurationsschema für Hochverfügbarkeit                = -->
<!-- = Das Schema beschreibt die Elemente der Basiskomponente von    = -->
<!-- = DB2 High Availability Automation for Multiplatforms (SA MP),  = -->
<!-- = die in der Konfiguration eines HA-Clusters verwendet werden.  = -->
<!-- ================================================================== -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="db2ha.xsd" cluster ManagerName="TSA" version="1.0">

<!-- ================================================================== -->
  <!-- = ClusterDomain-Element                                        = -->
  <!-- = Dieses Element bindet die Spezifikation der                  = -->
  <!-- = Clusterkonfiguration ein.                                    = -->
  <!-- = Clusterdomäne namens db2HAdomain erstellen.                  = -->
  <!-- = IP-Quorumeinheit (IP 19.126.4.5) erstellen.                  = -->
  <!-- = Die IP muss jederzeit von allen Knoten in der Clusterdomäne   = -->
  <!-- = mit Ping überprüft werden können.                            = -->
<!-- ================================================================== -->
 <ClusterDomain domainName="db2HAdomain">
  <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

<!-- ================================================================== -->
  <!-- = PhysicalNetwork-Element                                      = -->
  <!-- = Das physische Netz gibt den Netztyp, das Protokoll, die      = -->
  <!-- = IP-Adresse, die Teilnetzmaske und den NIC-Namen an.          = -->
  <!-- = Zwei logische Gruppierungen von NICs definieren.             = -->
  <!-- = Zwei logische Gruppierungen von NICs definieren.             = -->
<!-- ================================================================== -->
  <PhysicalNetwork physicalNetworkName="db2_public_network_0"
physicalNetworkProtocol="ip">
   <Interface interfaceName="eth0" clusterNodeName="hasys01">
    <IPAddress baseAddress="19.126.52.139"
 subnetMask="255.255.255.0" networkName="db2_public_network_0"/>
   </Interface>
   <Interface interfaceName="eth0" clusterNodeName="hasys02">
```

```
    <IPAddress baseAddress="19.126.52.140"
 subnetMask="255.255.255.0" networkName="db2_public_network_0"/>
   </Interface>
  </PhysicalNetwork>

  <PhysicalNetwork physicalNetworkName="db2_private_network_0"
physicalNetworkProtocol="ip">
   <Interface interfaceName="eth1" clusterNodeName="hasys01">
    <IPAddress baseAddress="192.168.23.101"
 subnetMask="255.255.255.0" networkName="db2_private_network_0"/>
   </Interface>
   <Interface interfaceName="eth1" clusterNodeName="hasys02">
     <IPAddress baseAddress="192.168.23.102"
 subnetMask="255.255.255.0" networkName="db2_private_network_0"/>
   </Interface>
  </PhysicalNetwork>

<!-- ============================================================ -->
  <!-- = ClusterNodeName-Element                                  = -->
  <!-- = Die Gruppe der Knoten in der Clusterdomäne.              = -->
  <!-- = Die defierte Gruppe der Knoten in der Domäne ist hier    = -->
  <!-- = hasys01, hasys02.                                        = -->
<!-- ============================================================ -->
  <ClusterNode clusterNodeName="hasys01"/>
  <ClusterNode clusterNodeName="hasys02"/>
 </ClusterDomain>

<!-- ================================================================ -->
  <!-- = Element der Richtlinie für Funktionsübernahme              = -->
  <!-- = Die Richtlinie für Funktionsübernahme gibt die Reihenfolge = -->
  <!-- = für die Funktionsübernahme durch die Clusterknoten an.     = -->
  <!-- = Im aktuellen Beispiel startet die Richtlinie für Funktions- = -->
  <!-- = übernahme die vorhandene Instanz. (LocalRestart)           = -->
<!-- ================================================================ -->
 <FailoverPolicy>
  <HADRFailover></HADRFailover>
 </FailoverPolicy>

<!-- ================================================================ -->
  <!-- = Element für DB2-Partition                                  = -->
  <!-- = Der DB2-Partitionstyp gibt einen DB2-Instanzname und eine  = -->
  <!-- = Partitionsnummer an.                                       = -->
<!-- ================================================================ -->
 <DB2PartitionSet>
  <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
  </DB2Partition>
 </DB2PartitionSet>

<!-- ================================================================ -->
  <!-- = HADRDBSet                                                  = -->
  <!-- = Die Gruppe der HADR-Datenbanken für dies Instanz.          = -->
  <!-- = Geben Sie den Datenbanknamen, den Namen der lokalen Instanz = -->
  <!-- = auf diesem System, die die HADR-Datenbank steuert, den Namen = -->
  <!-- = der fernen Instanz in diesem HADR-Paar, den lokalen Hostnamen = -->
  <!-- = und den fernen Hostnamen für die ferne Instanz an.         = -->
<!-- ================================================================ -->
 <HADRDBSet>
  <HADRDB databaseName="HADRDB" localInstance="db2inst1"
   remoteInstance="db2inst1" localHost="hasys01" remoteHost="hasys02"/>
 </HADRDBSet>
</DB2Cluster>
```

DB2 High Availability Instance Configuration Utility (db2haicu) - Voraussetzungen

Vor der Verwendung von DB2 High Availability Instance Configuration Utility (db2haicu) müssen bestimmte Tasks ausgeführt werden.

Allgemein

Bevor der Eigner einer Datenbankmanagerinstanz **'db2haicu'** ausführen kann, muss ein Benutzer mit Rootberechtigung den Befehl **'preprpnode'** ausführen.

'preprpnode' ist Teil der RSCT-Dateigruppe (RSCT = Reliable Scalable Cluster Technology) für AIX und des RSCT-Pakets für Linux. **'preprpnode'** führt die Initialisierung der Knoten für die clusterinterne Kommunikation aus. Der Befehl **'preprpnode'** wird im Rahmen der Clusterkonfiguration ausgeführt. Weitere Informationen zu 'preprpnode' finden Sie auf den Websites zu folgenden Themen:

- `preprpnode` Command (AIX)
- `preprpnode` Command (Linux)

Weitere Informationen zu RSCT finden Sie unter RSCT Administration Guide - What is RSCT?

Darüber hinaus muss ein Benutzer mit Rootberechtigung die Module iTCO_wdt und iTCO_vendor_support inaktivieren.

- Unter SUSE müssen Sie die folgenden Zeilen zur Datei `/etc/modprobe.d/blacklist` hinzufügen:

  ```
  alias iTCO_wdt off
  alias iTCO_vendor_support off
  ```

- Unter RHEL müssen Sie die folgenden Zeilen zur Datei `/etc/modprobe.conf` hinzufügen:

  ```
  blacklist iTCO_wdt
  blacklist iTCO_vendor_support
  ```

Sie können überprüfen, ob die Module inaktiviert wurden, indem Sie den Befehl **lsmod** verwenden.

Vor der Ausführung von **db2haicu** muss der Eigner einer Datenbankmanagerinstanz die folgenden Tasks ausführen:

- Servicedateien auf allen Maschinen synchronisieren, die dem Cluster hinzugefügt werden.
- Script **db2profile** für die Datenbankmanagerinstanz ausführen, die zur Erstellung der Clusterdomäne verwendet wird.
- Datenbankmanager mit dem Befehl **db2start** starten.

DB2 High Availability Disaster Recovery (HADR)

Bei Verwendung der HADR-Funktion müssen Sie die folgenden Tasks ausführen:

- Stellen Sie sicher, dass alle DB2 HADR-Datenbanken in den entsprechenden Primär- und Bereitschaftsdatenbankrollen gestartet wurden und sich alle HADR-Datenbankpaare (Primär- und Bereitschaftsdatenbanken) im Peerstatus befinden.
- Konfigurieren Sie **hadr_peer_window** für alle HADR-Datenbanken mit einem Wert von mindestens 120 Sekunden.
- Inaktivieren Sie den DB2-Fehlermonitor.

Umgebung mit partitionierter Datenbank

Führen Sie die folgenden Schritte aus, wenn Sie mehrere Datenbankpartitionen für hohe Verfügbarkeit konfigurieren müssen:

- Konfigurieren Sie die Registrierdatenbankvariable **DB2_NUM_FAILOVER_NODES** auf allen Maschinen, die der Clusterdomäne hinzugefügt werden.

- (Optional) Aktivieren Sie die Datenbank vor der Ausführung von **db2haicu**.

Erstellen einer Clusterdomäne mit DB2 High Availability Instance Configuration Utility (db2haicu)

Wenn Sie DB2 High Availability Instance Configuration Utility (db2haicu) zum ersten Mal für eine Datenbankmanagerinstanz ausführen, erstellt **db2haicu** ein Modell des Clusters, das als *Clusterdomäne* bezeichnet wird.

Von DB2 High Availability Instance Configuration Utility (db2haicu) automatisch erkannte Datenbankpfade:

Wenn Sie DB2 High Availability Instance Configuration Utility (db2haicu) zum ersten Mal ausführen, sucht **db2haicu** im Datenbanksystem nach Informationen zur Datenbankkonfiguration, die sich auf die Clusterkonfiguration beziehen.

Umgebung mit Einzelpartitionsdatenbank

In einer Umgebung mit einer Einzelpartitionsdatenbank erkennt **db2haicu** automatisch folgende Pfade:

- Ausgangsverzeichnispfad der Instanz
- Prüfprotokollpfad
- Protokollpfad des Prüfarchivs
- Protokollpfad des Synchronisationspunktmanagers (SPM)
- Pfad des DB2-Diagnoseprotokolls (Protokolldatei **db2diag**)
- Datenbankspezifische Pfade:
  - Datenbankprotokollpfad
  - Pfad für Tabellenspeicherbereichscontainer der Datenbank
  - Pfad für das Tabellenspeicherbereichsverzeichnis der Datenbank
  - Lokales Datenbankverzeichnis

  Anmerkung: Wenn es sich bei einem oder mehreren der datenbankbezogenen Verzeichnisse um einen symbolischen Link handelt, gibt das Dienstprogramm **db2haicu** eine Fehlernachricht aus und wird beendet.

Umgebung mit mehreren Datenbankpartitionen

In einer Umgebung mit mehreren Datenbankpartitionen erkennt **db2haicu** lediglich die folgenden Pfade automatisch:

- Datenbankprotokollpfad
- Pfad für Tabellenspeicherbereichscontainer der Datenbank
- Pfad für das Tabellenspeicherbereichsverzeichnis der Datenbank
- Lokales Datenbankverzeichnis

Verwalten einer Clusterdomäne mit DB2 High Availability Instance Configuration Utility (db2haicu)

Wenn Sie das Clusterdomänenmodell der Clusterumgebung mit **db2haicu** modifizieren, gibt der Datenbankmanager die entsprechenden Änderungen an die Datenbankmanagerinstanz und die Clusterkonfiguration weiter.

Vorbereitende Schritte

Vor der Konfiguration der Clusterumgebung mit **db2haicu** müssen Sie eine Clusterdomäne erstellen und konfigurieren. Weitere Informationen hierzu finden Sie in

„Erstellen einer Clusterdomäne mit DB2 High Availability Instance Configuration Utility (db2haicu)" auf Seite 145.

Informationen zu diesem Vorgang

Zu den Wartungsaufgaben von **db2haicu** gehört das Hinzufügen von Datenbank- oder Clusterelementen wie beispielsweise Datenbanken oder Clusterknoten zur Clusterdomäne sowie das Entfernen von Elementen aus der Clusterdomäne. Ferner gehört zu den Wartungsaufgaben von **db2haicu** das Modifizieren der Details von Clusterdomänenelementen wie der Richtlinie für Funktionsübernahme für die Datenbankmanagerinstanz.

Vorgehensweise

1. Führen Sie **db2haicu** aus.

   Bei der Ausführung von **db2haicu** im Wartungsmodus zeigt **db2haicu** eine Liste der Operationen an, die Sie für die Clusterdomäne ausführen können:
   - Fügen Sie Clusterknoten hinzu, oder entfernen Sie sie. (Die Maschine wird durch den Hostnamen identifiziert.)
   - Fügen Sie eine Netzschnittstelle (Netzschnittstellenkarte) hinzu, oder entfernen Sie sie.
   - Fügen Sie DB2-Datenbankpartitionen hinzu, oder entfernen Sie sie. (Nur in Umgebungen mit partitionierten Datenbanken.)
   - Fügen Sie DB2-HADR-Datenbanken hinzu oder entfernen Sie sie.
   - Fügen Sie eine Hochverfügbarkeitsdatenbank hinzu, oder entfernen Sie sie.
   - Fügen Sie einen Mountpunkt hinzu, oder entfernen Sie ihn.
   - Fügen Sie eine IP-Adresse hinzu, oder entfernen Sie sie.
   - Fügen Sie einen nicht kritischen Pfad hinzu, oder entfernen Sie ihn.
   - Versetzen Sie DB2-Datenbankpartitionen und HADR-Datenbanken für die planmäßige Wartung.
   - Ändern Sie die Richtlinie für Funktionsübernahme für diese Instanz.
   - Erstellen Sie eine neue Quorumeinheit für die Domäne.
   - Löschen Sie die Domäne.
2. Wählen Sie eine Task aus, und beantworten Sie die Fragen, die Ihnen von **db2haicu** gestellt werden.

Ergebnisse

Der Datenbankmanager verwendet die Informationen in der Clusterdomäne für die Koordination mit dem Cluster-Manager. Wenn Sie Datenbank- und Clusterelemente mit **db2haicu** konfigurieren, werden diese Elemente in die integrierte und automatisierte Clusterkonfiguration und -verwaltung von DB2 High Availability Feature eingefügt. Wenn Sie mit **db2haicu** eine Konfigurationsänderung der Datenbankmanagerinstanz vornehmen, führt der Datenbankmanager die erforderliche Konfigurationsänderung für den Cluster-Manager für Sie durch, damit kein nachfolgender Aufruf an den Cluster-Manager erforderlich ist.

Nächste Schritte

DB2 High Availability Instance Configuration Utility (db2haicu) verfügt über kein separates Diagnoseprotokoll. Fehler bei **db2haicu** können mit dem Diagnoseprotokoll des Datenbankmanagers, der Protokolldatei **db2diag.log** und dem Tool **db2pd** untersucht und diagnostiziert werden. Weitere Informationen hierzu finden Sie in

„Fehlerbehebung bei DB2 High Availability Instance Configuration Utility (db2haicu)".

Fehlerbehebung bei DB2 High Availability Instance Configuration Utility (db2haicu)

DB2 High Availability Instance Configuration Utility (db2haicu) verfügt über kein separates Diagnoseprotokoll. Fehler bei **db2haicu** können mit dem Diagnoseprotokoll des Datenbankmanagers, der Protokolldatei **db2diag.log** und dem Tool **db2pd** untersucht und diagnostiziert werden.

DB2 High Availability Instance Configuration Utility (db2haicu) - Einschränkungen

Für die Verwendung von DB2 High Availability Instance Configuration Utility (db2haicu) gelten bestimmte Einschränkungen.

- „Software und Hardware"
- „Konfigurationstasks"
- „Hinweise zur Verwendung"
- „Empfehlungen" auf Seite 149

Software und Hardware

- **db2haicu** unterstützt IP-Version 6 nicht.
- **db2haicu** unterstützt Logical Volume Manager (LVM) auf keiner anderen Plattform als AIX.

Konfigurationstasks

Die folgenden Tasks können mit **db2haicu** nicht ausgeführt werden:

- Sie können die automatische Clientweiterleitung nicht mithilfe von **db2haicu** konfigurieren.
- Wenn Sie ein Upgrade von DB2 for Linux, UNIX and Windows Version 9.5 auf eine höhere Version durchführen, können Sie die Clusterkonfiguration mit **db2haicu** nicht migrieren. Zur Migration einer Clusterkonfiguration müssen Sie die folgenden Schritte ausführen:
  1. Löschen Sie gegebenenfalls die vorhandene Clusterdomäne.
  2. Führen Sie ein Upgrade des Datenbankservers durch.
  3. Erstellen Sie mit **db2haicu** eine neue Clusterdomäne.

Hinweise zur Verwendung

Das Dienstprogramm 'db2haicu' wird nicht in einer DB2 pureScale-Umgebung unterstützt. Verwenden Sie zum Konfigurieren von Clusterumgebungen stattdessen das Dienstprogramm 'db2cluster'.

Beachten Sie bei der Planung von Clusterkonfiguration und Verwaltungsaktivitäten die folgenden Hinweise zur Verwendung von **db2haicu**:

- Obwohl **db2haicu** einige Verwaltungstasks ausführt, für die normalerweise eine Rootberechtigung erforderlich ist, wird **db2haicu** mit den Zugriffsrechten des Eigners der Datenbankmanagerinstanz ausgeführt. Die von einem Root ausgeführte Initialisierung von **db2haicu** ermöglicht **db2haicu** die Ausführung der erforderlichen Konfigurationsänderungen, obwohl nur die Zugriffsrechte des Instanzeigners vorliegen.

- Bei der Erstellung einer neuen Clusterdomäne überprüft **db2haicu** nicht, ob der angegebene Name für die neue Clusterdomäne korrekt ist. **db2haicu** prüft beispielsweise nicht, ob die Länge des Namens korrekt ist, ob der Name gültige Zeichen enthält oder dem Namen einer vorhandenen Clusterdomäne entspricht.
- **db2haicu** überprüft bzw. bestätigt keine Informationen, die von einem Benutzer angegeben und an einen Cluster-Manager übermittelt werden. Da **db2haicu** nicht alle Einschränkungen des Cluster-Managers hinsichtlich der Namen von Clusterobjekten kennen kann, leitet **db2haicu** beispielsweise Textdaten an den Cluster-Manager weiter, ohne diesen Text auf gültige Zeichen, Textlänge usw. zu überprüfen.
- Wenn ein Fehler auftritt und **db2haicu** während der Erstellung und Konfiguration einer neuen Clusterdomäne fehlschlägt, müssen Sie die folgenden Schritte ausführen:
  1. Entfernen Sie die Ressourcengruppen der teilweise erstellten Clusterdomäne, indem Sie **db2haicu** unter Verwendung des Parameters **-delete** ausführen.
  2. Wiederholen Sie die Erstellung der neuen Clusterdomäne, indem Sie **db2haicu** erneut aufrufen.
- Bei der Ausführung von **db2haicu** mit dem Parameter **-delete** werden die Ressourcengruppen, die der aktuellen Datenbankmanagerinstanz zugeordnet sind, von **db2haicu** unverzüglich gelöscht, ohne zu überprüfen, ob diese Ressourcengruppen gesperrt sind.
- Führen Sie die folgenden Schritte aus, um Ressourcengruppen zu entfernen, die den Datenbankmanagerinstanzen eines DB2 HADR-Datenbankpaares (Primär- und Bereitschaftsdatenbank) zugeordnet sind:
  1. Führen Sie **db2haicu** mit dem Parameter **-delete** zuerst für die Datenbankmanagerinstanz der HADR-Bereitschaftsdatenbank aus.
  2. Führen Sie anschließend **db2haicu** mit dem Parameter **-delete** für die Datenbankmanagerinstanz der HADR-Primärdatenbank aus.
- Zum Entfernen einer virtuellen IP aus einer HADR-Ressourcengruppe mit **db2haicu** müssen Sie diese aus der Instanz entfernen, in der sie erstellt wurde.
- Wenn eine Cluster-Operation, die Sie mit **db2haicu** ausführen möchten, das Zeitlimit überschreitet, gibt **db2haicu** keinen Fehler zurück. Sie erhalten nur dann Kenntnis davon, dass eine Cluster-Operation das Zeitlimit überschritten hat, wenn Sie nach dem **db2haicu**-Aufruf die Diagnoseprotokolle überprüfen oder nach dem Fehlschlagen einer nachfolgenden Clusteraktion den Fehler untersuchen und dabei feststellen, dass die ursprüngliche Cluster-Operation das Zeitlimit überschritten hat.
- Wenn Sie versuchen, die Richtlinie für Funktionsübernahme für eine bestimmte Datenbankinstanz in 'Aktiv/Passiv' zu ändern, gibt es eine Bedingung, bei deren Eintreten die Konfigurationsoperation fehlschlägt, für die **db2haicu** jedoch keinen Fehler zurückgibt. Wenn Sie eine Maschine, die momentan offline ist, als *aktive* Maschine angeben, macht **db2haicu** diese Maschine nicht zur aktiven Maschine, gibt aber keinen Fehler zurück, der auf das Fehlschlagen der Änderung hinweist.
- Bei einer Konfiguration für eine gemeinsam genutzte Platte unterstützt **db2haicu** keine verschachtelte Mountkonfiguration, weil DB2 die Plattenmountreihenfolge nicht umsetzt.
- Beim Hinzufügen von Netzschnittstellenkarten (NICs) zu einem Netz können diesem mithilfe von **db2haicu** keine NICs mit unterschiedlichen Teilnetzmasken hinzugefügt werden. Wenn Sie NICs mit unterschiedlichen Teilnetzmasken in dasselbe Netz einfügen möchten, verwenden Sie den folgenden SA MP-Befehl:

  ```
  mkequ <name> IBM.NetworkInterface:<eth0>:<node0>,...,<ethN>:<nodeN>
  ```

Empfehlungen

Die folgende Liste enthält Empfehlungen zur Konfiguration des Clusters und der Datenbankmanagerinstanzen bei Verwendung von **db2haicu**.

- Wenn Sie für den Cluster neue Mountpunkte hinzufügen, indem Sie unter `/etc/fstab` Einträge einfügen, können Sie mit der Option **noauto** verhindern, dass die Mountpunkte automatisch an mehrere Maschinen im Cluster angehängt werden. Beispiel:

  ```
  dev/vpatha1      /db/svtpdb/NODE0010        ext3  noauto 0 0
  ```

Unterstützte Cluster-Management-Software

Cluster-Management-Software ermöglicht Ihnen die Weiterleitung von DB2-Datenbankoperationen von einer fehlgeschlagenen Primärdatenbank in einem Clusterknoten an eine sekundäre Datenbank in einem anderen Clusterknoten.

Die DB2-Datenbank unterstützt die folgende Cluster-Management-Software:

- IBM PowerHA SystemMirror for AIX (bisher High Availability Cluster Multi-Processing für AIX oder HACMP)

  Detaillierte Informationen zur Konfiguration von PowerHA SystemMirror mit DB2-Datenbankprodukten finden Sie in http://www.redbooks.ibm.com/abstracts/sg247363.html?Open.
- Tivoli System Automation for Multiplatforms

  Detaillierte Informationen zur Konfiguration von Tivoli System Automation mit DB2-Datenbankprodukten finden Sie in http://www.redbooks.ibm.com/abstracts/sg247363.html?Open.
- Microsoft Cluster Server für Windows-Betriebssysteme.

  Detaillierte Informationen zur Konfiguration von Microsoft Cluster Server mit DB2-Datenbankprodukten finden Sie in http://www.redbooks.ibm.com/abstracts/sg247363.html?Open.
- Sun Cluster oder VERITAS Cluster Server für das Solaris-Betriebssystem

  Informationen zu Sun Cluster finden Sie im White Paper „DB2 Universal Database and High Availability on Sun Cluster 3.X", das auf der Website der IBM Softwarebibliothek (http://www.ibm.com/software/sw-library/) verfügbar ist. Informationen zu VERITAS Cluster Server finden Sie im White Paper „DB2 UDB and High Availability with VERITAS Cluster Server", das auf der Website „IBM Support and downloads" (unter http://www.ibm.com/support/docview.wss?uid=swg21045033) verfügbar ist.
- Multi-Computer/ServiceGuard für Hewlett-Packard

IBM PowerHA SystemMirror for AIX (bisher High Availability Cluster Multi-Processing für AIX oder HACMP)

IBM PowerHA SystemMirror for AIX ist eine Cluster-Management-Software. Die Knoten in PowerHA SystemMirror-Clustern tauschen Nachrichten aus, sog. *Heartbeats* (Überwachungssignale) oder Keepalive-Pakete. Wenn ein Knoten diese Nachrichten nicht mehr sendet, ruft PowerHA SystemMirror die Funktionsübernahme (Failover) für die anderen Knoten im Cluster auf, und wenn der fehlgeschlagene Knoten repariert ist, reintegriert ihn PowerHA SystemMirror wieder in den Cluster.

Es gibt zwei Ereignistypen: Standardereignisse, die innerhalb der Operationen von PowerHA SystemMirror abgefangen werden, und benutzerdefinierte Ereignisse, die mit der Überwachung von Parametern in Hardware- und Softwarekomponenten verbunden sind.

Eines der Standardereignisse ist das node_down-Ereignis. Dies bedeutet, ein Knoten im Cluster ist fehlgeschlagen, und PowerHA SystemMirror hat die Funktionsübernahme für alle anderen Knoten im Cluster initialisiert. Bei der Planung, welche Maßnahmen als Teil des Recoveryprozesses durchzuführen sind, ermöglicht PowerHA SystemMirror zwei Funktionsübernahmeoptionen: den reinen Bereitschaftsmodus (Hot bzw. Idle Standby) und den Modus der gegenseitigen Übernahme (Mutual Takeover).

Anmerkung: Wenn Sie PowerHA SystemMirror verwenden, stellen Sie sicher, dass DB2-Instanzen nicht beim Booten des Systems gestartet werden. Verwenden Sie dazu das Dienstprogramm **db2iauto** wie folgt:

```
db2iauto -off instanzname
```

Dabei gilt Folgendes: *instanzname* ist der Anmeldename der Instanz.

Clusterkonfiguration

In einer Konfiguration im *Bereitschaftsmodus* (Hot Standby) hat der AIX-Prozessorknoten, der als Übernahmeknoten fungiert, *keine andere* Auslastung. Bei einer Konfiguration für die *gegenseitige Funktionsübernahme* hat der AIX-Prozessorknoten, der als Übernahmeknoten fungiert, *andere* Auslastungen.

Allgemein wird die DB2-Datenbank in Umgebungen mit partitionierten Datenbanken im Modus der gegenseitigen Übernahme (Mutual Takeover) mit Datenbankpartitionen auf jedem Knoten ausgeführt. Eine Ausnahme bildet ein Szenario, in dem die Katalogpartition Teil einer Bereitschaftskonfiguration ist.

Ein Planungsaspekt ist die Verwaltung großer Cluster. Die Verwaltung eines kleinen Clusters ist einfacher als die eines großen. Die Verwaltung eines einzelnen großen Clusters ist jedoch immer noch einfacher als die vieler kleiner Cluster. Bei der Planung ist auch zu bedenken, wie die Anwendungen in der Clusterumgebung verwendet werden. Wenn zum Beispiel eine einzige, umfangreiche und homogene Anwendung auf 16 Knoten betrieben wird, ist die Konfiguration als ein Cluster wahrscheinlich einfacher zu verwalten, als sie in acht Cluster mit jeweils zwei Knoten zu zerlegen. Wenn dieselben 16 Knoten viele verschiedene Anwendungen mit verschiedenen Netzen, Platten und Knotenbeziehungen enthalten, ist es wahrscheinlich besser, die Knoten in kleinere Cluster zu gruppieren. Beachten Sie, dass Knoten nur nacheinander, d. h., nur einer gleichzeitig, in einen PowerHA SystemMirror-Cluster integriert werden. Eine Konfiguration mit mehreren Clustern lässt sich schneller starten als ein einziger großer Cluster. PowerHA SystemMirror unterstützt sowohl einzelne als auch mehrere Cluster, vorausgesetzt, ein Knoten und der zugehörige Ausweichknoten befinden sich im selben Cluster.

Die PowerHA SystemMirror-Funktion der Recovery durch Funktionsübernahme ermöglicht eine vordefinierte Zuordnung einer Ressourcengruppe zu einem physischen Knoten (auch als *hintereinandergeschaltete Zuordnung* bezeichnet). Die Recovery durch Funktionsübernahme ermöglicht außerdem eine gleitende (auch als *rotierende Zuordnung* bezeichnete) Zuordnung einer Ressourcengruppe zu einem physischen Knoten. IP-Adressen und externe Datenträgergruppen oder Dateisysteme oder NFS-Dateisysteme und Anwendungsserver innerhalb jeder Ressourcengruppe geben entweder eine Anwendung oder eine Anwendungskomponente an, die von PowerHA SystemMirror zwischen den physischen Knoten durch Funktionsübernahme und Reintegration beeinflusst werden können. Die Funktionsübernahme und Reintegration wird durch die Art der erstellten Ressourcengruppe und durch die Anzahl der in der Ressourcengruppe angelegten Knoten angegeben.

Betrachten Sie zum Beispiel eine DB2-Datenbankpartition (logischer Knoten). Wenn die Protokoll- und Tabellenbereichscontainer auf externen Platten angelegt und andere Knoten mit diesen Platten verbunden würden, wäre es möglich, dass die anderen Knoten auf diese Platten zugreifen und die Datenbankpartition (auf einem Übernahmeknoten) neu starten. Eben diese Art von Operation wird durch PowerHA SystemMirror automatisiert. PowerHA SystemMirror kann außerdem dazu verwendet werden, NFS-Dateisysteme wiederherzustellen, die von Hauptbenutzerverzeichnissen der DB2-Instanz verwendet werden.

Lesen Sie sich im Rahmen Ihrer Planung für die Recovery mit dem DB2-Datenbanksystem in einer Umgebung mit partitionierten Datenbanken die Dokumentation zu PowerHA SystemMirror sorgfältig durch. Sie sollten die Handbücher zu Konzepten, Planung, Installation und Verwaltung lesen und anschließend die Recoveryarchitektur für Ihre Umgebung entwerfen. Ermitteln Sie für die einzelnen Subsysteme, die Sie aufgrund bekannter möglicher Fehlerpunkte zur Recovery vorgesehen haben, die benötigten PowerHA SystemMirror-Cluster und die Recoveryknoten (entweder im Bereitschaftsmodus oder im Modus für gegenseitige Übernahme).

Wenn Sie beabsichtigen, PowerHA SystemMirror auf zwei oder mehr Computern einzusetzen, die dasselbe Ausgangsverzeichnis gemeinsam nutzen, müssen Sie den Datenbankmanager in demselben Installationspfad installieren. Bei Verwendung von symbolischen Links zu einem ähnlichen Installationspfad könnten in diesem Szenario Probleme auftreten. Die Installationspfade müssen derselbe physische Pfad sein.

Es ist ausdrücklich zu empfehlen, sowohl Platten als auch Adapter in der externen Plattenkonfiguration zu spiegeln. Bei physischen DB2-Knoten, die für PowerHA SystemMirror konfiguriert sind, muss sorgfältig sichergestellt werden, dass die Knoten in der Datenträgergruppe von den gemeinsamen externen Platten abweichen können. In einer Konfiguration für gegenseitige Übernahme macht diese Anordnung eine zusätzliche Planung erforderlich, sodass die zu Paaren verbundenen Knoten ohne Konflikt auf die Datenträgergruppen des jeweils anderen Knotens zugreifen können. In einer Umgebung mit partitionierten Datenbanken bedeutet dies, dass alle Containernamen über alle Datenbanken hinweg für alle SMS- oder DMS-Tabellenbereiche eindeutig sein müssen. Tabellenbereiche des dynamischen Speichers übernehmen die entsprechenden Verarbeitungsschritte zur Erfüllung dieser Anforderung.

Eine Methode zur Sicherstellung dieser Eindeutigkeit besteht darin, die Datenbankpartitionsnummer als Teil des Namens zu verwenden. Sie können einen Knotenausdruck für die Syntax von Containerzeichenfolgen angeben, wenn Sie SMS- oder DMS-Container erstellen. Wenn Sie den Ausdruck angeben, kann die Knotennummer Teil des Containernamens sein, oder, wenn Sie zusätzliche Argumente angeben, können die Ergebnisse dieser Argumente Teil des Containernamens sein. Verwenden Sie das Argument " $N" (blank]$N), um den Knotenausdruck anzugeben. Das Argument muss an das Ende der Containerzeichenfolge gesetzt werden und kann nur in einem der folgenden Formate verwendet werden:

Tabelle 7. Argumente zur Containererstellung.. Als Knotennummer wird fünf (5) angenommen.

Syntax	Beispiel	Wert
blank]$N	" $N"	5
blank]$N+ number]	" $N+1011"	1016
blank]$N% number]	" $N%3"	2

Tabelle 7. Argumente zur Containererstellung. (Forts.). Als Knotennummer wird fünf (5) angenommen.

Syntax	Beispiel	Wert
blank]$N+ number]% number]	" $N+12%13"	4
blank]$N% number]+ number]	" $N%3+20"	22

Anmerkung:

1. % bedeutet Modulus.
2. In allen Fällen werden die Operatoren von links nach rechts ausgewertet.

Es folgen einige Beispiele zur Erstellung von Containern mithilfe dieses speziellen Arguments:

- Erstellen von Containern zur Verwendung auf einem Zweiknotensystem

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
        (device '/dev/rcont $N' 20000)
```

  Die folgenden Container würden verwendet:

```
/dev/rcont0   - auf Knoten 0
   /dev/rcont1   - auf Knoten 1
```

- Erstellen von Containern zur Verwendung auf einem Vierknotensystem

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
        (file '/DB2/containers/TS2/container $N+100' 10000)
```

  Die folgenden Container würden verwendet:

```
/DB2/containers/TS2/container100   - auf Knoten 0
   /DB2/containers/TS2/container101   - auf Knoten 1
   /DB2/containers/TS2/container102   - auf Knoten 2
   /DB2/containers/TS2/container103   - auf Knoten 3
```

- Erstellen von Containern zur Verwendung auf einem Zweiknotensystem

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
        ('/TS3/cont $N%2, '/TS3/cont $N%2+2')
```

  Die folgenden Container würden verwendet:

```
/TS3/cont0   - auf Knoten 0
   /TS3/cont2   - auf Knoten 0
   /TS3/cont1   - auf Knoten 1
   /TS3/cont3   - auf Knoten 1
```

Konfigurieren von DB2-Datenbankpartitionen für PowerHA SystemMirror

Nach der Konfiguration werden die Datenbankpartitionen in einer Instanz durch PowerHA SystemMirror nacheinander, ein physischer Knoten nach dem anderen, gestartet. Mehrere Cluster werden empfohlen, wenn parallele DB2-Konfigurationen mit mehr als vier Knoten gestartet werden sollen. Beachten Sie, dass es in einer parallelen DB2-Konfiguration mit 64 Knoten schneller ist, 32 PowerHA SystemMirror-Cluster mit zwei Knoten als vier Cluster mit 16 Knoten parallel zu starten.

Eine Scriptdatei gehört zum Lieferumfang von DB2 Enterprise Server Edition, um Hilfestellung bei der Konfiguration für PowerHA SystemMirror-Funktionsübernahme bzw. -Recovery auf Bereitschaftsknoten oder Knoten mit gegenseitiger Übernahme zu geben. Die Scriptdatei hat den Namen `rc.db2pe.ee` für einen Einzelknoten und `rc.db2pe.eee` für mehrere Knoten. Diese Dateien befinden sich im

Verzeichnis `sqllib/samples/hacmp/es`. Kopieren Sie die entsprechende Datei in das Verzeichnis `/usr/bin` auf jedem System im PowerHA SystemMirror-Cluster und benennen Sie sie in `rc.db2pe` um.

Zusätzlich können aus `rc.db2pe` heraus DB2-Pufferpoolgrößen während der Funktionsübernahme bei Konfigurationen mit gegenseitiger Übernahme angepasst werden. (Pufferpoolgrößen können konfiguriert werden, um eine ordnungsgemäße Ressourcenzuordnung sicherzustellen, wenn zwei Datenbankpartitionen auf demselben physischen Knoten ausgeführt werden.)

PowerHA SystemMirror-Ereignisüberwachung und benutzerdefinierte Ereignisse

Ein Beispiel für ein benutzerdefiniertes Ereignis ist das Einleiten einer Funktionsübernahmeoperation, wenn auf einem bestimmten Knoten ein Prozess abgebrochen wird. Ereignisse müssen im Rahmen der Clusterkonfiguration manuell als benutzerdefinierte Ereignisse konfiguriert werden.

Detaillierte Informationen zu Implementierung und Design von hoch verfügbaren IBM DB2-Datenbankumgebungen finden Sie auf der Website der IBM Softwarebibliothek (http://www.ibm.com/software/sw-library/).

Zugehörige Informationen:

Information Center für PowerHA SystemMirror

IBM Tivoli System Automation for Multiplatforms (Linux und AIX)

IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) ist eine Software für das Cluster-Management, mit der die automatische Verlegung von Benutzern, Anwendungen und Daten von einem Datenbanksystem zu einem anderen innerhalb eines Clusters möglich wird. Tivoli SA MP automatisiert die Steuerung von IT-Ressourcen wie z. B. Prozessen, Dateisystemen und IP-Adressen.

Tivoli SA MP stellt ein Framework zur automatischen Verwaltung der Verfügbarkeit so genannter Ressourcen bereit. Ressourcen sind zum Beispiel:

- Jede Softwarekomponente, zu deren Steuerung Start-, Überwachungs- und Stoppscripts geschrieben werden können.
- Jede beliebige Netzschnittstellenkarte (NIC), auf die Tivoli SA MP Zugriff erteilt wurde. Das heißt, Tivoli SA MP verwaltet die Verfügbarkeit jeder IP-Adresse, die ein Benutzer verwenden will durch eine fließende Zuordnung dieser IP-Adresse unter den Netzschnittstellenkarten, auf die Zugriff besteht.

Zum Beispiel haben sowohl eine DB2-Instanz als auch die High Availability Disaster Recovery-Funktion Start-, Stopp- und Überwachungsbefehle. Dies ermöglicht das Schreiben von Tivoli SA MP-Scripts, um die Verwaltung dieser Ressourcen zu automatisieren. Ressourcen, die eng zusammengehören (z. B. solche, die gleichzeitig zusammen auf demselben Knoten ausgeführt werden), werden als *Ressourcengruppe* bezeichnet.

DB2-Ressourcen

In einer DB2-Umgebung mit nur einer Partition wird eine einzelne DB2-Instanz auf einem Server ausgeführt. Diese DB2-Instanz hat lokalen Zugriff auf Daten (auf das eigene ausführbare Image sowie auf Datenbanken, deren Eigner die Instanz ist). Wenn diese DB2-Instanz fernen Clients zugänglich gemacht wird, muss dieser DB2-Instanz eine nicht genutzte IP-Adresse zugewiesen werden.

Die DB2-Instanz, die lokalen Daten und die IP-Adresse werden sämtlich als Ressourcen betrachtet, die durch Tivoli SA MP automatisiert werden müssen. Da diese Ressourcen eng zusammengehören (z. B. werden sie gleichzeitig zusammen auf demselben Knoten ausgeführt), werden Sie als Ressourcengruppe bezeichnet.

Die gesamte Ressourcengruppe befindet sich auf einem Knoten im Cluster. Im Fall einer Funktionsübernahme wird die gesamte Ressourcengruppe auf einem anderen Knoten gestartet.

Die folgenden Abhängigkeiten bestehen zwischen den Ressourcen in der Gruppe:

- Die DB2-Instanz muss nach der lokalen Platte gestartet werden.
- Die DB2-Instanz muss vor der lokalen Platte gestoppt werden.
- Die HA-IP-Adresse muss mit der Instanz zusammengelegt werden.

Plattenspeicher

Die DB2-Datenbank kann die folgenden Ressourcen als lokalen Datenspeicher nutzen:

- Rohplatteneinheit (Beispiel: `/dev/sda1`)
- Logischer, durch Logical Volume Manager (LVM) verwalteter Datenträger
- Dateisystem (Beispiele: ext3, jfs)

DB2-Daten können entweder insgesamt auf einer oder mehreren Rohplatteneinheiten, insgesamt auf logischen Datenträgern, insgesamt in Dateisystemen oder in einer Mischung als allen drei Möglichkeiten gespeichert werden. Alle ausführbaren Dateien müssen sich in einer Art Dateisystem befinden.

DB2-Datenbankanforderungen für die HA-IP-Adresse

Die DB2-Datenbank hat keine besonderen Anforderungen im Hinblick auf die IP-Adresse. Es ist nicht erforderlich, eine hoch verfügbare IP-Adresse zu definieren, damit die Instanz als hoch verfügbar betrachtet wird. Jedoch ist es wichtig zu beachten, dass die IP-Adresse, die geschützt wird (sofern vorhanden), den Zugriffspunkt des Clients auf die Daten darstellt, sodass diese Adresse als solche allen Clients bekannt sein muss. Für die Praxis ist zu empfehlen, dass diese IP-Adresse die Adresse ist, die von den Clients in ihren Befehlen **CATALOG TCPIP NODE** verwendet wird.

Tivoli SA MP-Ressourcengruppen

IBM Tivoli System Automation for Multiplatforms ist ein Produkt, das eine hohe Verfügbarkeit realisiert, indem es Ressourcen wie Prozesse, Anwendungen, IP-Adressen und andere in Linux-basierten Clustern automatisiert. Zur Automatisierung einer IT-Ressource (wie zum Beispiel einer IP-Adresse) muss die Ressource in Tivoli SA MP definiert werden. Darüber hinaus müssen alle diese Ressourcen in mindestens einer Ressourcengruppe enthalten sein. Wenn diese Ressourcen stets auf derselben Hostmaschine aktiv sein müssen, sollten sie alle in dieselbe Ressourcengruppe eingefügt werden.

Jede Anwendung muss als Ressource definiert werden, um mit Tivoli SA MP verwaltet und automatisiert werden zu können. Anwendungsressourcen werden in der Regel in der generischen Ressourcenklasse IBM.Application definiert. In dieser Ressourcenklasse sind verschiedene Attribute zur Definition einer Ressource enthalten. Mindestens drei von ihnen sind jedoch anwendungsspezifisch:

- StartCommand
- StopCommand
- MonitorCommand

Diese Befehle können Scripts oder ausführbare Binärdateien sein.

Konfigurieren von Tivoli SA MP in Ihrer DB2-Umgebung

Wenn Sie detaillierte Konfigurationsinformationen benötigen, die Ihnen bei der Einrichtung von Tivoli SA MP in Ihrer DB2-Umgebung helfen, suchen Sie nach "Tivoli System Automation" auf der Website der IBM Softwarebibliothek (http://www.ibm.com/software/sw-library/).

Unterstützung für Microsoft Failover Clustering (Windows)

Microsoft Failover Clustering (Funktionsübernahmeclustering) unterstützt Server-Cluster unter Windows-Betriebssystemen. Dieses Feature erkennt Server- oder Anwendungsfehler automatisch und reagiert entsprechend; darüber hinaus kann die Auslastung der Server ausgeglichen werden.

Einführung

Microsoft Failover Clustering ist ein Feature von Windows-Serverbetriebssystemen. Diese Software unterstützt die Verbindung zweier Server (bis zu vier Servern in DataCenter Server) in einem Cluster für hohe Verfügbarkeit sowie eine einfachere Verwaltung von Daten und Anwendungen. Failover Clustering stellt automatisch Server- und Anwendungsfehler bzw. -ausfälle fest und führt eine Recovery durch. Mithilfe von Failover Clustering können Serverauslastungen verlagert werden, um eine gleichmäßige Maschinennutzung zu gewährleisten und geplante Wartungsarbeiten ohne Ausfallzeiten durchführen zu können.

Die folgenden DB2-Datenbankprodukte stellen Unterstützung für Microsoft Failover Clustering bereit:

- DB2 Connect-Serverprodukte (DB2 Connect Enterprise Edition, DB2 Connect Application Server Edition, DB2 Connect Unlimited Edition für iSeries und DB2 Connect Unlimited Edition für zSeries).
- DB2 Advanced Enterprise Server Edition
- DB2 Enterprise Server Edition
- DB2 Express Edition
- DB2 Workgroup Server Edition

DB2 Failover Clustering-Komponenten

Ein Cluster ist eine Konfiguration mit mindestens zwei Knoten, die jeweils unabhängige Computersysteme sind. Der Cluster erscheint Netzclients wie ein einzelner Server.

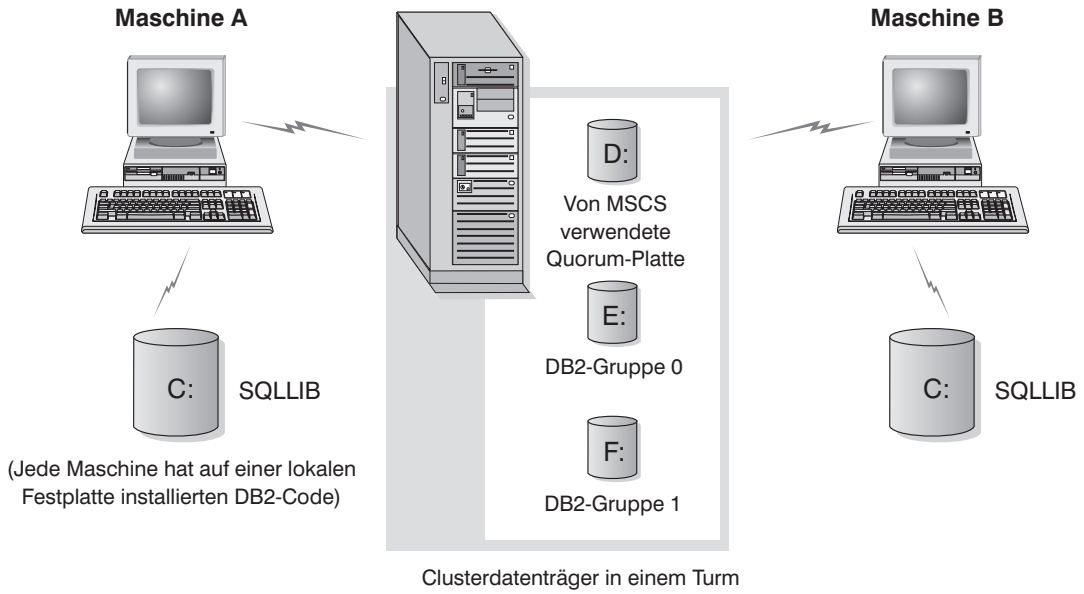


Abbildung 4. Beispiel für eine Failover Clustering-Konfiguration

Die Knoten in einem Failover Clustering-Cluster werden durch mindestens einen gemeinsam genutzten Speicherbus und mindestens ein physisch unabhängiges Netz miteinander verbunden. Das Netz, das nur die Server miteinander verbindet, nicht jedoch die Clients mit dem Cluster, wird als *privates* Netz bezeichnet. Das Netz, das die Clientverbindungen unterstützt, wird als *öffentliches* Netz bezeichnet. Auf jedem Knoten befindet sich mindestens eine lokale Platte. Jeder gemeinsam genutzte Speicherbus ist mit mindestens einer Platte verbunden. Für jede Platte im gemeinsam genutzten Bus liegt das Eigentumsrecht jeweils nur bei einem Clusterknoten. Die DB2-Software befindet sich auf der lokalen Platte. Die DB2-Datenbankdateien (z. B. Tabellen, Indizes, Protokolldateien) befinden sich auf gemeinsam genutzten Platten. Da Microsoft Failover Clustering die Verwendung von unformatierten Partitionen in einem Cluster nicht unterstützt, kann DB2 in Microsoft Failover Clustering-Umgebungen nicht zur Verwendung von Roheinheiten konfiguriert werden.

Die DB2-Ressource

In einer Microsoft Failover Clustering-Umgebung ist eine Ressource eine Entität, die von der Clustersoftware verwaltet wird. Beispielsweise kann eine Platte, eine IP-Adresse oder ein generischer Service als Ressource verwaltet werden. Zur Integration von Microsoft Failover Clustering erstellt DB2 einen eigenen Ressourcentyp mit dem Namen 'DB2 Server'. Jede DB2-Serverressource verwaltet eine DB2-Instanz. Bei Ausführung in einer Umgebung mit partitionierten Datenbanken verwaltet jede DB2-Serverressource eine Datenbankpartition. Der Name der DB2-Serverressource ist der Instanzname. In Umgebungen mit partitionierten Datenbanken setzt sich der Name der DB2-Serverressource jedoch aus dem Instanznamen und der Nummer der Datenbankpartition (bzw. des Knotens) zusammen.

Prä-Online- und Post-Online-Scripts

Sie können Scripts ausführen, bevor eine DB2-Ressource in den Onlinestatus versetzt wird und nachdem sie in den Onlinestatus versetzt wurde. Diese Scripts werden als Prä-Online-Scripts bzw. Post-Online-Scripts bezeichnet. Diese Scripts sind BAT-Dateien, die DB2- und Systembefehle ausführen können.

In Situationen, in denen möglicherweise mehrere DB2-Instanzen auf derselben Maschine ausgeführt werden, können Sie mithilfe der Prä- und Post-Online-Scripts die Konfiguration anpassen, sodass beide Instanzen erfolgreich gestartet werden können. Wenn eine Funktionsübernahme stattfindet, können Sie mit dem Post-Online-Script eine manuelle Datenbankrecovery ausführen. Ein Post-Online-Script kann auch zum Starten von Anwendungen oder Services verwendet werden, die von DB2 abhängig sind.

Die DB2-Gruppe

Zugehörige oder abhängige Ressourcen werden in Ressourcengruppen organisiert. Alle in einer Gruppe vorhandenen Ressourcen werden zwischen Clusterknoten als eine Einheit versetzt. In einer typischen DB2-Clusterumgebung mit einer einzelnen Partition ist beispielsweise eine DB2-Gruppe vorhanden, die die folgenden Ressourcen enthält:

1. Ressource 'DB2'. Die Ressource 'DB2' verwaltet die DB2-Instanz (oder den Knoten).
2. Ressource 'IP-Adresse'. Die Ressource 'IP-Adresse' ermöglicht es Clientanwendungen, eine Verbindung zum DB2-Server herzustellen.
3. Ressource 'Netzname'. Die Ressource 'Netzname' ermöglicht es Clientanwendungen, unter Verwendung eines Namens anstatt einer IP-Adresse eine Verbindung zum DB2-Server herzustellen. Für die Ressource 'Netzname' besteht eine Abhängigkeit zur Ressource 'IP-Adresse'. Die Ressource 'Netzname' ist optional. (Das Konfigurieren einer Ressource 'Netzname' kann sich auf die Leistung der Funktionsübernahme auswirken.)
4. Mindestens eine Ressource 'Physische Platte'. Jede Ressource 'Physische Platte' verwaltet eine gemeinsam genutzte Platte im Cluster.

Anmerkung: Die Ressource 'DB2' ist so konfiguriert, dass sie von allen anderen Ressourcen in derselben Gruppe abhängt, sodass der DB2-Server nur dann gestartet werden kann, wenn alle anderen Ressourcen online sind.

Failover-Konfigurationen

Es sind zwei Konfigurationstypen verfügbar:

- Aktiv/Passiv
- Gegenseitige Übernahme

In einer Umgebung mit partitionierten Datenbanken verfügen nicht alle Cluster über denselben Konfigurationstyp. Sie können Cluster haben, die im Modus "Aktiv/Passiv" konfiguriert sind, und andere, die für gegenseitige Übernahme konfiguriert sind. Wenn Ihre DB2-Instanz zum Beispiel aus fünf Workstations besteht, können Sie zwei Maschinen zur gegenseitigen Übernahme und zwei im passiven Bereitschaftsmodus konfigurieren und eine Maschine von der Funktionsübernahmekonfiguration ausnehmen.

Aktive/Passive Konfiguration

In einer Konfiguration für den aktiven/passiven Modus stellt eine Maschine im Microsoft Failover Clustering-Cluster dedizierte Übernahmeunterstützung bereit, die andere Maschine ist Teil des Datenbanksystems. Wenn die zum Datenbanksystem gehörige Maschine ausfällt, wird ihr Datenbankserver auf der Übernahmemaschine gestartet. Wenn Sie in einer Umgebung mit partitionierten Datenbanken mehrere logische Knoten auf einer Maschine betreiben und die Maschine ausfällt,

werden die logischen Knoten auf der Übernahmemaschine gestartet. Abb. 5 zeigt ein Beispiel einer Konfiguration für den aktiven/passiven Modus.

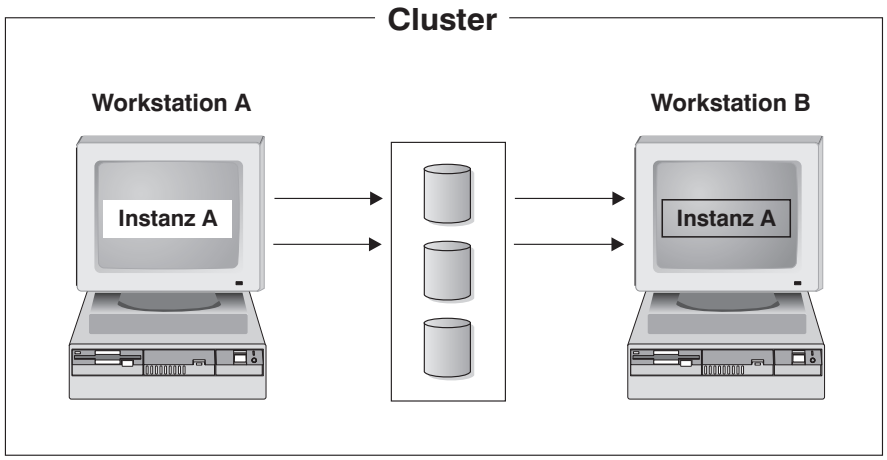


Abbildung 5. Aktive/Passive Konfiguration

Konfiguration zur gegenseitigen Übernahme

In einer Konfiguration zur gegenseitigen Übernahme sind beide Workstations an einem Datenbanksystem beteiligt (d. h., auf jeder Maschine wird mindestens ein Datenbankserver ausgeführt). Wenn eine der Workstations im Microsoft Failover Clustering-Cluster ausfällt, wird der Datenbankserver der ausgefallenen Maschine zur Ausführung auf der anderen Maschine gestartet. In einer Konfiguration zur gegenseitigen Übernahme kann ein Datenbankserver auf einer Maschine unabhängig vom Datenbankserver auf einer anderen Maschine ausfallen. Jeder Datenbankserver kann zu einem beliebigen Zeitpunkt auf jeder Maschine aktiv sein. Abb. 6 zeigt ein Beispiel für die Konfiguration zur gegenseitigen Übernahme.

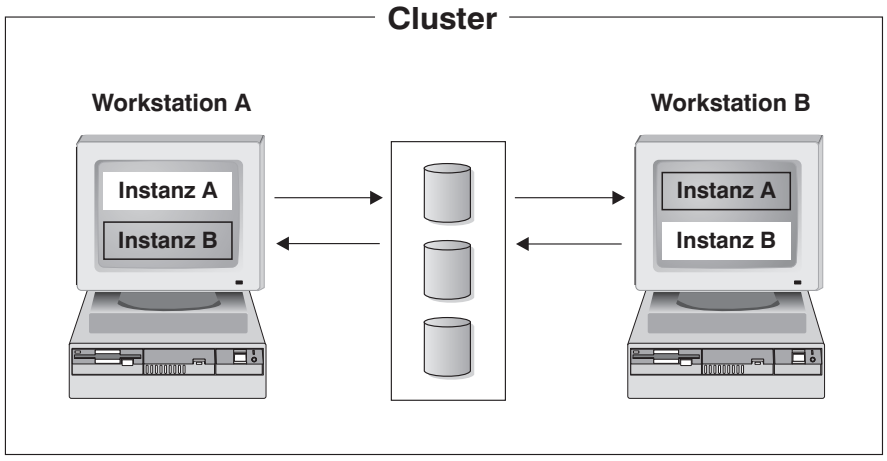


Abbildung 6. Konfiguration zur gegenseitigen Übernahme

Detaillierte Informationen zu Implementierung und Design von hoch verfügbaren IBM DB2-Datenbankumgebungen unter Windows-Betriebssystemen finden Sie auf der Website "IBM Software Library" (http://www.ibm.com/software/sw-library/).

Unterstützung für Failover Clustering unter Windows Server 2008

Gehen Sie wie folgt vor, um partitionierte DB2-Datenbanksysteme für die Ausführung auf Windows Server 2008-Funktionsübernahmeclustern zu konfigurieren:

1. Gehen Sie entsprechend den im White Paper „Implementing IBM DB2 Universal Database V8.1 Enterprise Server Edition with Microsoft Cluster Server" beschriebenen Prozeduren vor, das auf der Website der IBM-Softwarebibliothek (http://www.ibm.com/software/sw-library/) zur Verfügung steht.
2. Aufgrund von Änderungen an der Failover Clustering-Komponente von Windows Server 2008 ist möglicherweise folgende zusätzliche Konfiguration erforderlich:
   - In den Windows Server 2008-Funktionsübernahmeclustern wird der Windows-Clusterservice unter einem besonderen Account des lokalen Systems ausgeführt, während der Windows-Clusterservice in Windows Server 2003 unter einem Administatoraccount ausgeführt wird. Dies wirkt sich auf die Operationen der DB2-Ressource (`db2server.dll`) aus, die im Kontext des Cluster-Service-Kontos ausgeführt wird.

     Die Gruppen DB2ADMNS und DB2USERS müssen Domänengruppen sein, wenn die Registrierdatenbankvariable **DB2_EXTSECURITY** in einem Windows-Funktionsübernahmecluster auf den Wert `YES` gesetzt wurde.

     Wenn eine Mehrpartitionsinstanz in einem Windows-Funktionsübernahmecluster ausgeführt wird, muss für den Pfad INSTPROF ein Netzpfad festgelegt werden (z. B. `\\`*`netzname`*`\DB2MSCS-DB2\DB2PROFS`). Dies erfolgt automatisch, wenn Sie das DB2-Datenbanksystem mit dem Befehl **db2mscs** zu einem Cluster zusammenfassen.

     Wenn der Windows Server 2008-Funktionsübernahmecluster eingerichtet ist, wird in Active Directory ein Computerobjekt erstellt, das den neuen Cluster darstellt. Wenn beispielsweise der Name des Clusters MYCLUSTER lautet, wird in Active Directory das Computerobjekt MYCLUSTER erstellt. Wenn ein Benutzer eine Instanz mit mehreren Partitionen zu einem Cluster zusammenfasst und die Registrierdatenbankvariable **DB2_EXTSECURITY** auf `YES` (die Standardeinstellung) gesetzt wird, muss dieses Computerobjekt zu der Gruppe DB2ADMNS hinzugefügt werden. Dies ist erforderlich, damit die DB2-Ressourcen-DLL auf den Pfad `\\`*`netzname`*`\DB2MSCS-DB2\DB2PROFS` zugreifen kann. Wenn beispielsweise die DB2-Administratorgruppe die Bezeichnung MYDOMAIN\DB2ADMNS hat, muss das Computerobjekt MYCLUSTER zu dieser Gruppe hinzugefügt werden. Nachdem das Computerobjekt der Gruppe DB2ADMNS hinzugefügt wurde, müssen Sie zum Abschluss für beide Knoten im Cluster einen Warmstart durchführen.
   - In Windows Server 2008 Failover Clustering wird die „Dateifreigaberessource des Clusters ("cluster fileshare resource")" nicht mehr unterstützt. Stattdessen wird der Dateiserver des Clusters verwendet. Die Dateifreigabe (eine normale Dateifreigabe) basiert auf der Dateiserverressource des Clusters. Microsoft erfordert, dass die im Cluster erstellten Dateiserver des Clusters das Domain Name System (DNS) für die Namensauflösung verwenden. Bei der Ausführung von Instanzen mit mehreren Partitionen ist für die Unterstützung der Dateifreigabe eine Dateiserverressource erforderlich. Die in der Datei `db2mscs.cfg` angegebenen Parameter **NETNAME_NAME**, **NETNAME_VALUE** und **NETNAME_DEPENDENCY** werden zum Erstellen der Dateiserver- und Dateifreigaberessourcen verwendet. Der *netzname* basiert auf einer IP-Adresse und dieser *netzname* muss in DNS vorliegen. Beispiel: Wenn die Datei `db2mscs.cfg` die folgenden Parameter enthält, wird eine Dateifreigabe namens `\\MSCSV\DB2MSCS-DB2` erstellt:

```
...
NETNAME_NAME = MSCSN
NETNAME_VALUE = MSCSV
...
```

Der MSCSV-Name muss in DNS registriert werden. Andernfalls schlägt der Dateiserver oder die Dateifreigabe, der bzw. die für den DB2-Cluster erstellt wurde, fehl, wenn die DNS-Auflösung nicht erfolgreich ist.

Unterstützung für Solaris-Betriebssystemcluster

DB2 unterstützt zwei für das Betriebssystem Solaris verfügbare Cluster-Manager: Sun Cluster und Veritas Cluster Server (VCS).

Anmerkung: Wenn Sie Sun Cluster 3.0 oder Veritas Cluster Server verwenden, stellen Sie sicher, dass DB2-Instanzen beim Booten nicht gestartet werden. Verwenden Sie dazu das Dienstprogramm **db2iauto** wie folgt:

```
db2iauto -off instanzname
```

Dabei ist *instanzname* der Anmeldename der Instanz.

Hohe Verfügbarkeit

Computersysteme, die als Host für Datenbankservices dienen, enthalten viele unterschiedliche Komponenten, und jede Komponente besitzt eine für sie ermittelte „mittlere ausfallfreie Zeit" (MTBF - Mean Time Before Failure). Die MTBF ist die durchschnittliche Dauer, die eine Komponente verwendbar bleibt. Die MTBF für ein Qualitätsfestplattenlaufwerk liegt in der Größenordnung von 1 Million Stunden (annähernd 114 Jahren). Obwohl dies wie ein langer Zeitraum erscheint, ist es wahrscheinlich, dass eine von 200 Festplatten innerhalb eines Zeitraums von sechs Monaten ausfällt.

Wenngleich es eine Reihe von Methoden zur Erhöhung der Verfügbarkeit für einen Datenbankservice gibt, ist die am häufigsten eingesetzte Methode ein HA-Cluster (High Availability-Cluster). Wenn ein Cluster zur Implementierung hoher Verfügbarkeit eingesetzt wird, besteht er aus mindestens zwei Maschinen, einer Gruppe privater Netzschnittstellen, mindestens einer öffentlichen Netzschnittstelle sowie einigen gemeinsam benutzten Platten. Diese spezielle Konfiguration ermöglicht das Versetzen eines Datenbankservice von einer Maschine auf eine andere. Durch das Versetzen des Datenbankservice auf eine andere Maschine im Cluster wird es möglich, den Zugriff auf die zugehörigen Daten weiterhin aufrecht zu erhalten. Das Versetzen eines Datenbankservice von einer Maschine auf eine andere wird als *Funktionsübernahme* bezeichnet (siehe Abb. 7 auf Seite 161 zur Illustration).

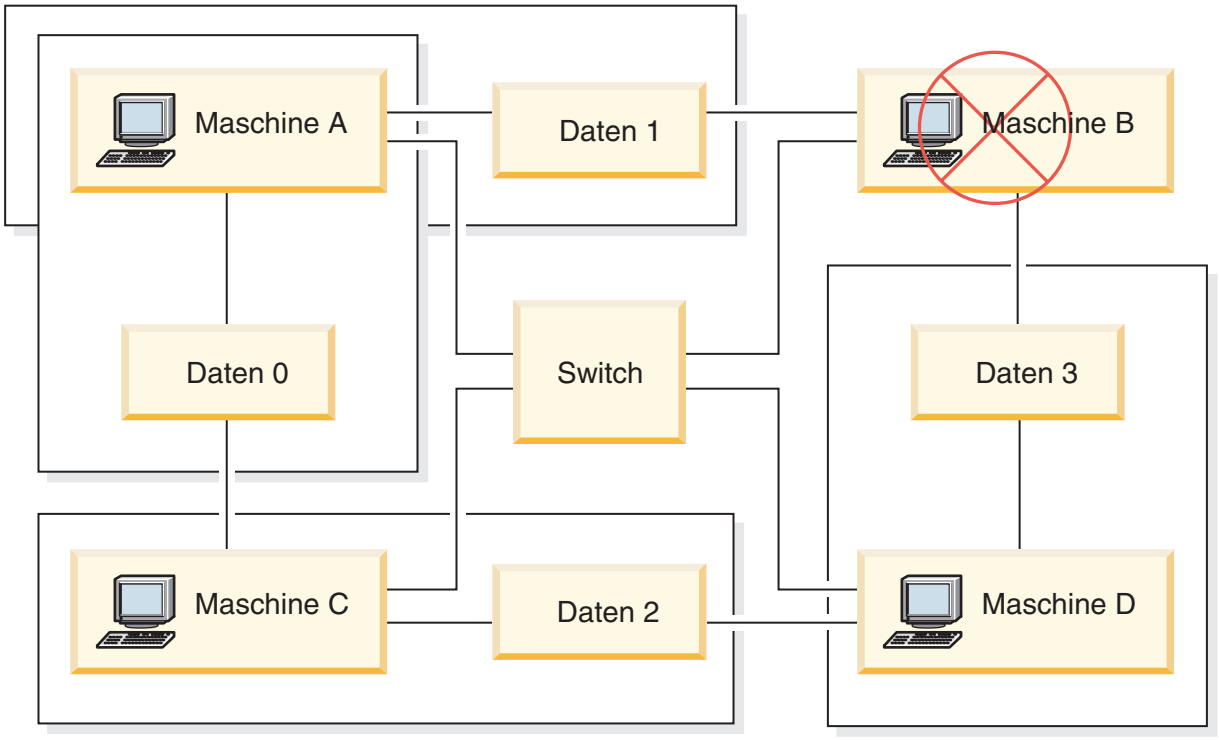


Abbildung 7. Funktionsübernahme. Wenn Maschine B ausfällt, wird ihr Datenbankservice auf eine andere Maschine im Cluster versetzt, sodass weiter auf die Daten zugegriffen werden kann.

Die privaten Netzschnittstellen werden zum Senden von *Überwachungssignal*nachrichten und von Steuernachrichten zwischen den Maschinen im Cluster verwendet. Die öffentlichen Netzschnittstellen dienen zur direkten Kommunikation mit Clients des HA-Clusters. Die Platten in einem HA-Cluster sind mit zwei oder mehr Maschinen im Cluster verbunden, sodass bei Ausfall einer Maschine eine andere Maschine auf sie zugreifen kann.

Ein Datenbankservice, der auf einem HA-Cluster aktiv ist, besitzt mindestens eine öffentliche Netzschnittstelle und eine Gruppe von Platten, die ihm zugeordnet sind. Die Clients eines HA-Datenbankservice stellen eine Verbindung über TCP/IP nur zu den logischen Netzschnittstellen des Datenbankservices her. Wenn es zu einer Funktionsübernahme kommt, werden der Datenbankservice zusammen mit den zugehörigen Netzschnittstellen und der Gruppe von Platten auf eine andere Maschine versetzt.

Einer der Vorteile eines HA-Clusters liegt darin, dass ein Datenbankservice ohne Hilfe durch Unterstützungspersonal wiederhergestellt werden kann, und dass dies zu jedem beliebigen Zeitpunkt möglich ist. Ein weiterer Vorteil ist die Redundanz. Alle Teile im Cluster sollten redundant sein, einschließlich der Maschinen selbst. Der Cluster sollte in der Lage sein, jeden einzelnen Fehlerpunkt (SPoF, Single Point of Failure) überbrücken zu können.

Auch wenn sich hochverfügbare Datenbankservices in ihrer Art sehr unterscheiden können, stellen sie einige allgemeine Anforderungen. Clients eines hochverfügbaren Datenbankservice erwarten, dass sich die Netzadresse und der Hostname des Datenbankservices nicht ändern und dass sie die Möglichkeit haben, ihre Anforderungen unabhängig von der Maschine, auf der der Datenbankservice aktiv ist, auf dieselbe Weise zu stellen.

Denken Sie z. B. an einen Web-Browser, der auf einen hochverfügbaren Web-Server zugreift. Die Anforderung wird mit einer URL-Adresse (Uniform Resource Locator) abgesetzt, die sowohl einen Hostnamen als auch den Pfad zu einer Datei auf dem Web-Server enthält. Der Browser erwartet, dass sowohl der Hostname als auch der Pfad nach einer Übernahme der Web-Server-Funktion gleich bleiben. Wenn der Browser gerade eine Datei vom Web-Server herunterlädt, wenn die Serverfunktion von einer anderen Maschine übernommen wird, muss der Browser die Anforderung erneut absetzen.

Die Verfügbarkeit eines Datenbankservice wird in der Zeitdauer gemessen, die der Datenbankservice seinen Benutzern zur Verfügung steht. Die am häufigsten verwendete Maßeinheit für Verfügbarkeit ist der Prozentsatz der Betriebszeit (Up Time). Diese wird oft in Form einer Neunerreihe angegeben:

```
99,99% => Service fällt (maximal) 52,6 Minuten / Jahr aus
99,999% => Service fällt (maximal) 5,26 Minuten / Jahr aus
99,9999% => Service fällt (maximal) 31,5 Sekunden / Jahr aus
```

Beachten Sie beim Entwerfen und Testen eines HA-Clusters folgende Punkte:

1. Stellen Sie sicher, dass der Administrator des Clusters mit dem System vertraut ist und weiß, was geschehen soll, wenn eine Funktionsübernahme auftritt.
2. Stellen Sie sicher, dass jeder Teil des Clusters wirklich redundant ist und im Falle eines Ausfalls rasch ausgetauscht werden kann.
3. Erzwingen Sie die Funktionsübernahme eines Testsystems in einer kontrollierten Umgebung, und stellen Sie sicher, dass die Funktionsübernahme jedes Mal ordnungsgemäß vollzogen wird.
4. Protokollieren Sie die Gründe für jede eingetretene Funktionsübernahme. Obwohl dies nicht oft geschehen sollte, ist es wichtig, alle Punkte zu untersuchen, die einen Cluster instabil machen. Wenn zum Beispiel eine Komponente des Clusters fünf Mal in einem Monat eine Funktionsübernahme erforderlich machte, stellen Sie fest, woran dies liegt, und beheben Sie das Problem.
5. Stellen Sie sicher, dass das Unterstützungspersonal für den Cluster benachrichtigt wird, wenn eine Funktionsübernahme eintritt.
6. Überlasten Sie den Cluster nicht. Stellen Sie sicher, dass die verbliebenen Systeme nach einer Funktionsübernahme die Auslastung weiterhin mit einer akzeptablen Leistung bewältigen können.
7. Überprüfen Sie ausfallanfälligere Komponenten (z. B. Festplatten) häufig, sodass sie ersetzt werden können, bevor Probleme auftreten.

Fehlertoleranz

Eine andere Methode zur Erhöhung der Verfügbarkeit eines Datenbankservice ist Fehlertoleranz. Bei einer *fehlertoleranten* Maschine ist sämtliche Redundanz integriert, sodass die Maschine in der Lage ist, einen Einzelausfall einer beliebigen Komponente, einschließlich CPU und Speicher, zu überbrücken. Fehlertolerante Maschinen werden hauptsächlich in Nischenmärkten eingesetzt und ihre Implementierung ist gewöhnlich mit hohen Kosten verbunden. Ein HA-Cluster mit Maschinen an verschiedenen geographischen Standorten hat den zusätzlichen Vorteil, einen Ausfall, der nur einen Teil dieser Standorte betrifft, überbrücken zu können.

Ein HA-Cluster ist die gängigste Lösung zur Erhöhung der Verfügbarkeit, weil sie skalierbar, einfach zu handhaben und relativ kostengünstig in der Implementierung ist.

Unterstützung für Sun Cluster 3.0 (und höher):

Wenn Sie vorhaben, Ihre DB2-Datenbanklösung auf einem Solaris-Betriebssystemcluster auszuführen, können Sie Sun Cluster 3.0 für die Verwaltung des Clusters verwenden. Ein Agent für hohe Verfügbarkeit fungiert als Mediator zwischen der DB2-Datenbank und Sun Cluster 3.0.

Die Anweisungen in diesem Abschnitt zur Unterstützung für Sun Cluster 3.0 gelten auch für höhere Sun Cluster-Versionen.

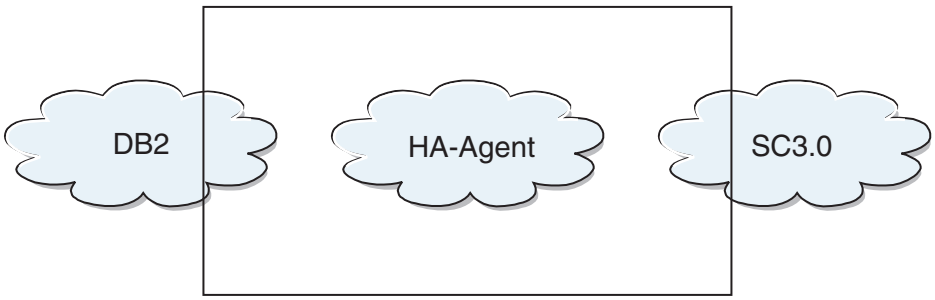


Abbildung 8. DB2, Sun Cluster 3.0 und hohe Verfügbarkeit. Die Beziehungen zwischen der DB2-Datenbank, Sun Cluster 3.0 und dem Agenten für hohe Verfügbarkeit.

Funktionsübernahme

Sun Cluster 3.0 bietet hohe Verfügbarkeit mittels Funktionsübernahme für Anwendungen. Jeder Knoten wird regelmäßig überwacht, und die Clustersoftware verlagert eine clustersensitive Anwendung automatisch von einem fehlgeschlagenen primären Knoten auf einen angegebenen sekundären Knoten. Bei einer Funktionsübernahme stellen Clients möglicherweise eine kurze Unterbrechung des Service fest und müssen die Verbindung zum Server wiederherstellen. Die Tatsache, dass sie nun auf einem anderen physischen Server auf die Anwendungen und Daten zugreifen, hat jedoch keine Auswirkungen. Dadurch, dass bei einem Ausfall des primären Servers automatisch andere Knoten in einem Cluster die Arbeitslast übernehmen, sorgt Sun Cluster 3.0 für eine erhebliche Reduzierung der Ausfallzeiten und eine spürbare Steigerung der Produktivität.

Mehrhostplatten

Sun Cluster 3.0 erfordert Mehrhostplattenspeicher. Das bedeutet, dass Platten gleichzeitig mit mehr als einem Knoten verbunden sein können. In einer Sun Cluster 3.0-Umgebung ermöglicht der Mehrhostspeicher eine hohe Verfügbarkeit der Platteneinheiten. Platteneinheiten, die sich im Mehrhostspeicher befinden, können die Ausfälle einzelner Knoten tolerieren, da über alternative Serverknoten immer ein physischer Pfad zu den Daten verfügbar ist. Auf Mehrhostplatten kann global über einen primären Knoten zugegriffen werden. Wenn Clientanforderungen über einen Knoten auf Daten zugreifen und dieser Knoten fehlschlägt, werden die Anforderungen an einen anderen Knoten umgeleitet, der eine direkte Verbindung zu derselben Platte hat. Ein Volume Manager stellt Spiegelkonfigurationen oder Konfigurationen mit RAID 5 bereit, die die Datenredundanz auf den Mehrhostplatten gewährleisten. Sun Cluster 3.0 unterstützt derzeit Solstice DiskSuite und VERITAS Volume Manager als Datenträgermanager. Die Kombination von Mehrhostplatten mit Plattenspiegelung und einheitenübergreifendem Lesen und Schreiben von Daten bietet gleichermaßen Schutz vor Knotenausfällen und Ausfällen einzelner Platten.

Globale Einheiten

Globale Einheiten stellen clusterweiten Zugriff mit hoher Verfügbarkeit auf alle Einheiten in einem Cluster bereit, und zwar von jedem Knoten aus und unabhängig von der physischen Position der Einheit. Alle Platten sind im globalen Namensbereich mit einer zugeordneten Einheiten-ID (Device ID, DID) erfasst und als globale Einheiten konfiguriert. Daher sind die Platten für alle Clusterknoten sichtbar.

Dateisysteme und globale Dateisysteme

Ein Cluster bzw. ein globales Dateisystem ist ein Proxy zwischen dem Kernel (auf einem Knoten) und dem Datenträgermanager des zugrunde liegenden Dateisystems (auf einem Knoten, der eine physische Verbindung zu mindestens einer Platte hat). Clusterdateisysteme sind von globalen Einheiten abhängig, die jeweils zu mindestens einem Knoten eine physische Verbindung haben. Sie sind unabhängig von dem zugrunde liegenden Dateisystem und Datenträgermanager. Derzeit können Clusterdateisysteme auf UFS eingerichtet werden, unter Verwendung von Solstice DiskSuite oder VERITAS Volume Manager. Die Daten werden nur dann allen Knoten zur Verfügung gestellt, wenn die Dateisysteme auf den Platten global als Clusterdateisystem eingerichtet sind.

Einheitengruppe

Eine Mehrhostplatte muss vom Sun Cluster-Framework gesteuert werden. Auf einer Mehrhostplatte werden zuerst so genannte Plattengruppen (Disk Groups) erstellt, die entweder von Solstice DiskSuite oder VERITAS Volume Manager verwaltet werden. Anschließend werden sie als Sun Cluster-Platteneinheitengruppen registriert. Eine Platteneinheitengruppe ist ein Typ der globalen Einheiten. Mehrhosteinheitengruppen haben eine hohe Verfügbarkeit. Wenn ein Knoten, der die Einheitengruppe gerade verwaltet, ausfällt, sind die Platten über einen alternativen Pfad weiter verfügbar. Der Ausfall des Knotens, der die Einheitengruppe verwaltet, hat keine Auswirkungen auf die Einheitengruppe, bis auf eine kurze zeitliche Unterbrechung, die zur Ausführung der Recovery und der Konsistenzprüfung erforderlich ist. Während dieser Zeit werden alle Anforderungen blockiert (für die Anwendung transparent), bis das System die Einheitengruppe wieder zur Verfügung stellt.

Ressourcengruppenmanager (RGM)

Der RGM stellt einen Hochverfügbarkeitsmechanismus bereit und wird auf jedem Clusterknoten als Dämon ausgeführt. Er übernimmt anhand von vordefinierten Richtlinien das automatische Starten und Stoppen von Ressourcen auf ausgewählten Knoten. Der RGM ermöglicht die hohe Verfügbarkeit einer Ressource im Fall eines Knotenausfalls bzw. stoppt die Ressource auf dem betreffenden Knoten und startet sie auf einem anderen Knoten erneut. Der RGM startet und stoppt ebenfalls automatisch ressourcenspezifische Monitore, die Ressourcenfehler feststellen und fehlschlagende Ressourcen auf andere Knoten verlagern können.

Datenbankservices

Mit dem Begriff Datenbankservice wird eine Anwendung eines Fremdanbieters bezeichnet, die zur Ausführung auf einem Cluster anstatt auf einem einzelnen Server konfiguriert ist. Ein Datenbankservice umfasst die Anwendungssoftware und die Software Sun Cluster 3.0, die die Anwendung startet, stoppt und überwacht. Sun Cluster 3.0 stellt Datenbankservicemethoden bereit, mit denen die Anwendung innerhalb des Clusters gesteuert und überwacht wird. Diese Methoden werden unter der Steuerung des Ressourcengruppenmanagers (RGM) ausgeführt, der mit diesen Methoden

die Anwendung auf den Clusterknoten startet, stoppt und überwacht. Zusammen mit der Clusterframeworksoftware und mit Mehrhostplatten ermöglichen es diese Methoden, dass Anwendungen als Datenbankservices mit hoher Verfügbarkeit fungieren können. Als Datenbankservices mit hoher Verfügbarkeit können sie wesentliche Anwendungsunterbrechungen nach einzelnen Ausfällen innerhalb des Clusters verhindern, unabhängig davon, ob der Ausfall bzw. Fehler auf einem Knoten, einer Schnittstellenkomponente oder in der Anwendung selbst auftrat. Der RGM verwaltet auch die Ressourcen im Cluster, einschließlich der Netzressourcen (wie logische Hostnamen und gemeinsam benutze Adressen) und der Anwendungsinstanzen.

Ressourcentyp, Ressource und Ressourcengruppe

Ein Ressourcentyp setzt sich aus Folgendem zusammen:

1. Einer Softwareanwendung, die auf dem Cluster ausgeführt wird.
2. Steuerprogrammen, die vom RGM als Rückrufmethoden verwendet werden, um die Anwendung als Clusterressource zu verwalten.
3. Einigen Eigenschaften, die Teil der statischen Konfiguration eines Clusters sind.

Der RGM verwendet die Ressourcentypeigenschaften zur Verwaltung der Ressourcen eines bestimmten Typs.

Eine Ressource erbt die Eigenschaftenen und Werte ihres Ressourcentyps. Eine Ressource ist eine Instanz der zugrunde liegenden Anwendung, die auf dem Cluster ausgeführt wird. Der Name der Instanz muss auf Clusterebene eindeutig sein. Jede Ressource muss in einer Ressourcengruppe konfiguriert sein. Der RGM versetzt alle Ressourcen in einer Gruppe auf demselben Knoten gleichzeitig in den Online- oder Offlinestatus. Zum Versetzen einer Ressourcengruppe in den Online- oder Offlinestatus ruft der RGM für jede Ressource in der Gruppe Rückrufmethoden auf.

Die Knoten, auf denen eine Ressourcengruppe online ist, werden als ihre primären Knoten oder Primärknoten bezeichnet. Eine Ressourcengruppe wird von ihren Primärknoten verwaltet. Jede Ressourcengruppe hat eine zugeordnete Eigenschaft, eine "Knotenliste", die vom Clusteradministrator definiert wird und anhand derer alle potenziellen Primärknoten oder Master der Ressourcengruppe angegeben werden.

Unterstützung für VERITAS Cluster Server:

Wenn Sie vorhaben, Ihre DB2-Datenbanklösung auf einem Solaris-Betriebssystemcluster auszuführen, können Sie VERITAS Cluster Server für die Verwaltung des Clusters verwenden.

VERITAS Cluster Server kann eine große Bandbreite an Anwendungen in heterogenen Umgebungen verwalten und unterstützt bis zu 32 Knotencluster sowohl in Speicherbereichsnetzen (SAN) als auch in traditionellen Client/Server-Umgebungen.

Hardwarevoraussetzungen

VERITAS Cluster Server unterstützt derzeit folgende Hardware:

- Für Serverknoten:
  - Jeden SPARC/Solaris-Server von Sun Microsystems mit mindestens 128 MB RAM, auf dem Solaris 2.6 oder höher ausgeführt wird.
- Für Plattenspeicher:

- EMC Symmetrix, IBM Enterprise Storage Server, HDS 7700 und 9xxx, Sun T3, Sun A5000, Sun A1000, Sun D1000 und jeden anderen Plattenspeicher, der von VCS 2.0 oder höher unterstützt wird. Weitere Auskünfte zu den unterstützten Plattensubsystemen können Sie bei Ihrem VERITAS-Ansprechpartner einholen oder der VCS-Dokumentation entnehmen.
- Typische Umgebungen erfordern (in jedem Clusterknoten) gespiegelte, private Platten für die DB2-Binärdateien sowie von Knoten gemeinsam genutzte Platten für die DB2-Daten.

• Für Netzverbindungen:
- Für die öffentlichen Netzverbindungen eine beliebige Netzverbindung, die eine Adressierung auf IP-Basis unterstützt.
- Für die Überwachungssignalverbindungen (clusterintern) sind redundante Überwachungssignalverbindungen erforderlich. Diese Voraussetzung kann durch den Einsatz von zwei zusätzlichen Ethernet-Controllern pro Server bzw. durch die Verwendung von einem zusätzlichen Ethernet-Controller pro Server und einer gemeinsam genutzten GAB-Platte pro Cluster erfüllt werden.

Softwarevoraussetzungen

Die folgenden VERITAS-Softwarekomponenten gelten als qualifizierte Konfigurationen:

• VERITAS Volume Manager 3.2 oder höher, VERITAS File System 3.4 oder höher, VERITAS Cluster Server 2.0 oder höher

• DB Edition für DB2 für Solaris 1.0 oder höher

Obwohl VERITAS Cluster Server keinen Datenträgermanager erfordert, wird dringend die Verwendung von VERITAS Volume Manager empfohlen, um die Installation, Konfiguration und Verwaltung zu vereinfachen.

Funktionsübernahme

VERITAS Cluster Server ist eine Hochverfügbarkeitslösung für Cluster, die eine höhere Verfügbarkeit von Anwendungsservices, zum Beispiel einer DB2-Datenbank, sicherstellt, indem sie eine Funktionsübernahme für Anwendungen ermöglicht. Der Status jedes einzelnen Clusterknotens und seiner zugehörigen Softwareservices wird regelmäßig überwacht. Wird ein Anwendungsservice (in diesem Fall der DB2-Datenbankservice) durch einen Fehler unterbrochen, stellt VERITAS Cluster Server oder der VCS HA-DB2-Agent (oder beide) den Fehler fest und führt automatisch Schritte zur Wiederherstellung des Service aus. Diese Schritte zur Wiederherstellung des Service können den Neustart des DB2-Datenbanksystems auf demselben Knoten oder das Versetzen des DB2-Datenbanksystems auf einen anderen Knoten im Cluster und den anschließenden Neustart auf diesem Knoten beinhalten. Wenn eine Anwendung auf einen neuen Knoten migriert werden muss, versetzt VERITAS Cluster Server alles zur Anwendung Gehörende (z. B. IP-Netzadressen, das Eigentumsrecht an zugehörigem Speicher) auf den neuen Knoten, sodass Benutzer nicht bemerken, dass der Service auf einem anderen Knoten ausgeführt wird. Die Benutzer verwenden zum Zugriff auf den Service weiterhin dieselben IP-Adressen, die nur jetzt auf einen anderen Clusterknoten verweisen.

Wenn in VERITAS Cluster Server ein Fehler auftritt, ist es möglich, dass Benutzer eine Unterbrechung des Service feststellen oder auch nicht. Dies hängt von dem Verbindungstyp ('stateful' oder 'stateless', d. h. mit oder ohne Austausch von Statusinformationen) zwischen dem Client und dem

Anwendungsservice ab. In Anwendungsumgebungen, in denen Verbindungen mit Austausch von Statusinformationen verwendet werden (wie bei der DB2-Datenbank), stellen Benutzer möglicherweise eine kurze Serviceunterbrechung fest und müssen sich nach erfolgter Funktionsübernahme gegebenenfalls neu anmelden. In Anwendungsumgebungen, die Verbindungen ohne Austausch von Statusinformationen verwenden (wie bei NFS), stellen Benutzer möglicherweise eine kurze Serviceverzögerung, jedoch keine Unterbrechung fest und müssen sich nicht erneut anmelden.

Durch die Unterstützung einer Anwendung als Service, der automatisch zwischen Clusterknoten migriert werden kann, reduziert VERITAS Cluster Server nicht nur ungeplante Ausfallzeiten, sondern kann auch geplante Ausfallzeiten (z. B. für Wartung und Upgrades) verkürzen. Funktionsübernahmen können auch manuell eingeleitet werden. Wenn auf einem bestimmten Knoten ein Hardware- oder Betriebssystemupgrade ausgeführt werden muss, kann das DB2-Datenbanksystem auf einen anderen Knoten im Cluster migriert werden, das Upgrade ausgeführt und das DB2-Datenbanksystem anschließend wieder auf den ursprünglichen Knoten zurückmigriert werden.

Für diese Typen von Clusterumgebungen wird der Einsatz von absturztoleranten Anwendungen empfohlen. Eine absturztolerante Anwendung kann nach einem unvermittelt auftretenden Absturz wiederhergestellt werden, und die Integrität der festgeschrieben Daten bleibt erhalten. Absturztolerante Anwendungen werden auch als *clustertolerante* Anwendungen bezeichnet. Das DB2-Datenbanksystem ist eine absturztolerante Anwendung.

Gemeinsam genutzter Speicher

Beim Einsatz zusammen mit dem VCS HA-DB2-Agenten erfordert VERITAS Cluster Server gemeinsam genutzten Speicher. Gemeinsam genutzter Speicher ist ein Speicher, der eine physische Verbindung zu mehreren Knoten im Cluster hat. Im gemeinsam genutzten Speicher vorhandene Platteneinheiten sind Knotenausfällen gegenüber tolerant, da der physische Pfad zu den Platteneinheiten über die anderen Cluster gewährleistet bleibt.

Durch die Steuerung von VERITAS Cluster Server erhalten Clusterknoten Zugriff auf gemeinsam genutzten Speicher über ein logisches Konstrukt, so genannte Plattengruppen (Disk Groups). Plattengruppen sind eine Sammlung logisch definierter Speichereinheiten, deren Eigentumsrecht automatisch zwischen Knoten in einem Cluster migriert werden kann. Eine Plattengruppe kann nur jeweils in einen Knoten importiert werden. Wenn z. B. Plattengruppe A in Knoten 1 importiert wird und Knoten 1 ausfällt, kann Plattengruppe A aus dem ausgefallenen Knoten exportiert und in einen neuen Knoten im Cluster importiert werden. VERITAS Cluster Server kann gleichzeitig mehrere Plattengruppen in einem einzelnen Cluster steuern.

Zusätzlich zur Definition von Plattengruppen kann ein Datenträgermanager unter Verwendung von Spiegelungstechnologie oder RAID 5 auf gemeinsam genutztem Speicher redundante Datenkonfigurationen bereitstellen. VERITAS Cluster Server unterstützt VERITAS Volume Manager und Solstice DiskSuite als Manager für logische Datenträger (LVM - Logical Volume Manager). Die Kombination von gemeinsam genutztem Speicher mit Plattenspiegelung und einheitenübergreifendem Lesen und Schreiben von Daten bietet gleichermaßen Schutz vor Knotenausfällen und Ausfällen einzelner Platten oder Controller.

VERITAS Cluster Server Global Atomic Broadcast (GAB) und Low Latency Transport (LLT)

In Clusterkonfigurationen ist ein knotenübergreifender Kommunikationsmechanismus erforderlich, damit Knoten Informationen zum Hardware- und Softwarestatus austauschen, Clusterzugehörigkeiten verfolgen und diese Informationen über alle Clusterknoten hinweg synchron halten können. VERITAS Cluster Server nutzt hierzu GAB (Global Atomic Broadcast), das zusammen mit LLT (Low Latency Transport) den erforderlichen Hochgeschwindigkeitsmechanismus mit geringen Latenzzeiten bereitstellt. GAB wird auf jedem Clusterknoten als Kernelmodul geladen und bietet einen ganzheitlichen Broadcastmechanismus, mit dem sichergestellt werden kann, dass alle Knoten die aktualisierten Statusinformationen gleichzeitig erhalten.

Durch den Einsatz eines Leistungsspektrums für kernelübergreifende Kommunikation bietet LLT eine Hochgeschwindigkeitsübertragungsmöglichkeit mit geringen Latenzzeiten für alle Informationen, die zwischen Clustern ausgetauscht und synchronisiert werden müssen. GAB wird auf LLT ausgeführt. VERITAS Cluster Server verwendet nicht IP als Überwachungssignalmechanismus, sondern bietet zwei zuverlässigere Optionen an. Es kann entweder GAB mit LLT als Überwachungssignalmechanismus konfiguriert werden, oder es kann eine GAB-Platte als Überwachungssignalfunktion auf Plattenbasis konfiguriert werden. Das Überwachungssignal muss über redundante Verbindungen ausgeführt werden. Diese Verbindungen können entweder zwei private Ethernet-Verbindungen zwischen Clusterknoten sein oder eine private Ethernet-Verbindung und eine GAB-Plattenverbindung. Die Verwendung von zwei GAB-Platten wird als Konfiguration nicht unterstützt, da für den Austausch von Clusterstatusinformationen zwischen Knoten eine private Ethernet-Verbindung erforderlich ist.

Weitere Informationen zu GAB oder LLT bzw. zu deren Konfiguration in VERITAS Cluster Server-Konfigurationen entnehmen Sie bitte dem Benutzerhandbuch von VERITAS Cluster Server 2.0 für Solaris.

Agentenbündel und Unternehmensagenten

Ein Agent ist ein Programm, das die Verfügbarkeit einer bestimmten Ressource oder Anwendung verwaltet. Wird ein Agent gestartet, ruft er die erforderlichen Konfigurationsinformationen von VCS ab und beginnt, die Ressource oder Anwendung regelmäßig zu überwachen und VCS mit dem Status zu aktualisieren. Im Allgemeinen werden Agenten dazu verwendet, Ressourcen in den Online- oder Offlinestatus zu versetzen und Ressourcen zu überwachen, wobei vier Servicetypen bereitgestellt werden: 'start' (starten), 'stop' (stoppen), 'monitor' (überwachen) und 'clean' (bereinigen). Mit 'start' und 'stop' werden Ressourcen in den Online- oder Offlinestatus versetzt, mit 'monitor' wird eine bestimmte Ressource oder Anwendung auf ihren Status hin überprüft, und 'clean' wird im Recoveryprozess verwendet.

Im Lieferumfang von VERITAS Cluster Server ist ein Agentenbündel enthalten, das mit VERITAS Cluster Server installiert wird. Die darin enthaltenen Agenten sind VCS-Prozesse, die vordefinierte Ressourcentypen verwalten, die in der Regel in Clusterkonfigurationen vorkommen, z. B. 'IP', 'mount' (anhängen), 'process' (verarbeiten) und 'share' (gemeinsam nutzen). Diese Agenten vereinfachen die Installation und Konfiguration von Clustern erheblich. Das mit VERITAS Cluster Server gelieferte Bündel umfasst über 20 Agenten.

Unternehmensagenten werden häufig auf bestimmte Anwendungen, z. B. die DB2-Datenbankanwendung, fokussiert. Der VCS HA-DB2-Agent kann

als Unternehmensagent betrachtet werden, und er tauscht über das VCS-Agentenframework Daten mit VCS aus.

Ressourcen, Ressourcentypen und Ressourcengruppen von VCS

Ein Ressourcentyp ist eine Objektdefinition, mit der innerhalb eines VCS-Clusters die zu überwachenden Ressourcen definiert werden können. Ein Ressourcentyp umfasst den Ressourcentypnamen und eine Reihe von Eigenschaften, die dieser Ressource zugeordnet sind und die unter Gesichtspunkten der Hochverfügbarkeit wichtig sind. Eine Ressource übernimmt die Eigenschaften und Werte ihres Ressourcentyps, und der Ressourcenname muss auf Clusterebene eindeutig sein.

Es gibt zwei Ressourcentypen: 'persistent' und 'standard' (nicht persistent). Persistente Ressourcen sind Ressourcen wie Netzschnittstellencontroller (NICs), die zwar überwacht, aber von VCS nicht in den Online- oder Offlinestatus versetzt werden. Standardressourcen sind Ressourcen, deren Online- und Offlinestatus von VCS gesteuert wird.

Das auf unterster Ebene überwachte Objekt ist eine Ressource, und es gibt verschiedene Ressourcentypen (z. B. gemeinsam genutzt, angehängt). Jede Ressource muss in eine Ressourcengruppe konfiguriert werden, und VCS versetzt alle Ressourcen einer Ressourcengruppe gleichzeitig in den Online- oder Offlinestatus. Zum Versetzen einer Ressourcengruppe in den Online- oder Offlinestatus ruft VCS für jede Ressource in der Gruppe die Methoden 'start' oder 'stop' auf. Es gibt zwei Ressourcengruppentypen: 'failover' (Funktionsübernahme) und 'parallel'. Eine DB2-Datenbankkonfiguration mit hoher Verfügbarkeit (in einer partitionierten oder nicht partitionierten Umgebung) verwendet Ressourcengruppen des Typs 'failover'.

Ein primärer Knoten bzw. Masterknoten ist ein Knoten, der potenziell eine Ressource enthalten kann. Mit dem Ressourcengruppenattribut `systemlist` wird angegeben, welche Knoten in einem Cluster primäre Knoten für eine bestimmte Ressourcengruppe sein können. In einem Cluster mit zwei Knoten sind normalerweise beide Knoten in der `systemlist` enthalten. In großen Clustern mit mehreren Knoten, die vielleicht mehrere Hochverfügbarkeitsanwendungen enthalten, kann es jedoch erforderlich sein, sicherzustellen, dass für bestimmte, von ihren Ressourcen auf der untersten Ebene definierte Anwendungsservices keine Funktionsübernahme auf bestimmten Knoten erfolgt.

Zwischen Ressourcengruppen können Abhängigkeiten definiert werden. VERITAS Cluster Server greift auf die Hierarchie dieser Ressourcengruppenabhängigkeiten zurück, um die Auswirkungen verschiedener Ressourcenausfälle zu bewerten und die Recovery entsprechend zu verwalten. Beispiel: Falls die Ressourcengruppe 'ClientAnw1' erst in den Onlinestatus versetzt werden kann, wenn die Ressourcengruppe 'DB2' erfolgreich gestartet wurde, wird die Ressourcengruppe 'ClientAnw1' als abhängig von Ressourcengruppe 'DB2' betrachtet.

Synchronisieren der Systemzeit in einer Umgebung mit partitionierten Datenbanken

Unter den Datenbankpartitionsservern sollte für eine relative hohe Synchronisation der Systemuhren gesorgt werden, um einen reibungslosen Datenbankbetrieb und eine uneingeschränkte Möglichkeit zur aktualisierenden Recovery zu gewährleisten.

Zeitunterschiede zwischen den Datenbankpartitionsservern, einschließlich aller potenziellen betriebs- und kommunikationsbedingten Verzögerungen für eine Transaktion, sollten kleiner sein als der für den Konfigurationsparameter *max_time_diff* des Datenbankmanagers angegebene Wert, mit dem die maximale Zeitdifferenz zwischen Knoten definiert wird.

Um sicherzustellen, dass die Zeitmarken der Protokollsätze die Reihenfolge von Transaktionen in einer Umgebung mit partitionierten Datenbanken richtig wiedergeben, verwendet DB2 die Systemuhr und die in der Datei `SQLOGCTL.LFH` gespeicherte virtuelle Zeitmarke der jeweiligen Maschine als Basis für die Zeitmarken in den Protokollsätzen. Wenn die Systemuhr jedoch vorgestellt wird, wird die Protokolluhr automatisch mit vorgestellt. Obwohl die Systemuhr wieder zurückgestellt werden kann, kann die Uhr für die Protokolle dies nicht und bleibt auf dieser *vorgestellten* Zeit, bis die Systemuhr mit dieser Zeit übereinstimmt. Dann sind die Uhren synchron. Dies hat zur Konsequenz, dass ein kurzfristiger Systemuhrfehler auf einem Datenbankknoten einen langfristigen Effekt auf die Zeitmarken von Datenbankprotokollen haben kann.

Nehmen Sie zum Beispiel an, dass die Systemuhr auf Datenbankpartitionsserver A irrtümlicherweise auf den 7. November 2005 gesetzt wird, obwohl das Jahr 2003 ist, und nehmen Sie weiter an, dass der Fehler korrigiert wurde, *nachdem* eine Transaktion zur Aktualisierung in der Datenbankpartition auf diesem Datenbankpartitionsserver festgeschrieben wurde. Wenn die Datenbank ständig verwendet und regelmäßig aktualisiert wird, bleibt jeder Zeitpunkt zwischen dem 7. November 2003 und dem 7. November 2005 durch die aktualisierende Recovery praktisch unerreichbar. Wenn die COMMIT-Operation auf Datenbankpartitionsserver A erfolgt ist, wird die Zeitmarke im Datenbankprotokoll auf 2005 gesetzt und die Uhr des Protokolls bleibt auf dem 7. November 2005 stehen, bis die Systemuhr diesen Zeitpunkt erreicht. Wenn Sie versuchen, eine aktualisierende Recovery bis zu einem Zeitpunkt innerhalb dieses Zeitrahmens durchzuführen, stoppt die Operation an der ersten Zeitmarke, die über den angegebenen Stoppzeitpunkt, d. h. den 7. November 2003, hinausgeht.

Zwar kann DB2 etwaige Aktualisierungen der Systemuhr nicht steuern, der Konfigurationsparameter des Datenbankmanagers *max_time_diff* kann jedoch die Wahrscheinlichkeit reduzieren, dass diesbezüglich Probleme auftreten:

- Die konfigurierbaren Werte für diesen Parameter reichen von 1 Minute bis zu 24 Stunden.
- Wenn die erste Verbindungsanforderung an eine Nichtkatalogpartition erfolgt, sendet der Datenbankpartitionsserver seine Zeit an die Katalogpartition für die Datenbank. Die Katalogpartition überprüft, ob die Zeit in der Datenbankpartition, die die Verbindung anfordert, und ihre eigene Zeit innerhalb des durch den Parameter *max_time_diff* definierten Bereichs liegen. Falls dieser Bereich überschritten wird, wird die Verbindung verweigert.
- Eine Aktualisierungstransaktion, die auf mehr als zwei Datenbankpartitionsserver in der Datenbank zugreift, muss überprüfen, ob die Systemuhren auf den beteiligten Datenbankpartitionsservern synchron sind, bevor die Aktualisierung festgeschrieben werden kann. Wenn zwei oder mehr Datenbankpartitionsserver eine Zeitdifferenz aufweisen, die den durch den Parameter *max_time_diff* gesetzten Grenzwert überschreitet, wird die Transaktion rückgängig gemacht, um zu verhindern, dass die falsche Zeit auf weitere Datenbankpartitionsserver übertragen wird.

Konvertieren von Client/Server-Zeitmarken

Das Konvertieren von Zeitmarken unterstützt Sie bei der präzisen Aufzeichnung von Datenbankaktivitäten. Es ermöglicht die Anzeige von Aktivitäten in Ortszeit, die im GMT-Zeitzonenformat aufgezeichnet werden, auch wenn sich der Datenbankserver an einem fernen Standort mit einer anderen Zeitzone befindet.

Zeitmarken sind zu Prüfzwecken unerlässlich. Es ist von zentraler Bedeutung, dass in einer Umgebung mit partitionierten Datenbanken die Integrität der Zeitmarken über alle Datenpartitionen hinweg gewährleistet wird.

In diesem Abschnitt wird die Generierung von Zeitmarken in einer Client/Server-Umgebung erläutert:

- Wenn Sie für eine aktualisierende Recovery eine Ortszeit angeben, werden alle Nachrichten ebenfalls in Ortszeit zurückgegeben.

  Anmerkung: Alle Zeitangaben werden auf dem Server und (in Umgebungen mit partitionierten Datenbanken) in der Katalogdatenbankpartition konvertiert.
- Die Zeitmarkenzeichenfolge wird auf dem Server in GMT konvertiert, sodass die Zeitmarke die Zeitzone des Servers und nicht die des Clients wiedergibt. Wenn sich der Client in einer anderen Zeitzone befindet als der Server, sollte die Ortszeit des Servers verwendet werden.
- Wenn die Zeichenfolge der Zeitmarke nahe am Zeitwechsel aufgrund der Sommerzeit liegt, muss klar sein, ob die Stoppzeit vor oder nach dem Zeitwechsel liegt, damit sie korrekt angegeben wird.

Kapitel 5. Verwaltung und Pflege einer Hochverfügbarkeitslösung

Wenn Ihre hoch verfügbare DB2-Datenbanklösung erstellt und konfiguriert ist und aktiv ausgeführt wird, gibt es einige laufend auszuführenden Aktivitäten. Sie müssen Ihre Datenbanklösung überwachen, pflegen und reparieren, damit sie für die Clientanwendungen zur Verfügung steht.

Vorgehensweise

Die folgenden Dinge müssen von Ihnen bei der Ausführung Ihres Datenbanksystems überwacht und gehandhabt werden:

1. Das Verwalten von Protokolldateien:

   Protokolldateien nehmen an Umfang zu und müssen archiviert werden; einige Protokolldateien müssen kopiert oder versetzt werden, damit sie für eine Restoreoperation zur Verfügung stehen.
2. Das Ausführen von Verwaltungsaktivitäten:
   - Software installieren
   - Upgrades für Hardware durchführen
   - Datenbanktabellen reorganisieren
   - Datenbankleistung optimieren
   - Datenbank-Backups
3. Primäre und sekundäre bzw. Bereitschaftsdatenbank synchronisieren, sodass die Funktionsübernahme (Failover) reibungslos funktioniert.
4. Unerwartete Ausfälle oder Fehler in der Hardware oder Software identifizieren und beseitigen.

Protokolldateiverwaltung

Der DB2 Datenbankmanager verwendet ein Nummerierungsschema zur Benennung der Protokolldateien. Diese Benennungsstrategie hat Auswirkungen auf die Wiederverwendung von Protokolldateien und auf die Protokollfolgen. Darüber hinaus verwendet eine DB2-Datenbank ohne Clientanwendungsverbindung eine neue Protokolldatei, wenn die nächste Clientanwendung eine Verbindung zu dem Datenbankserver herstellt.

Diese beiden Aspekte des Datenbankprotokollierungsverhaltens von DB2 Data Server wirken sich auf Ihre Auswahl in Bezug auf die Protokolldateiverwaltung aus.

Bei der Verwaltung von Datenbankprotokollen sind folgende Faktoren zu beachten:

- Das Nummerierungsschema für archivierte Protokolldateien beginnt mit `S0000000.LOG` und reicht bis `S9999999.LOG` und bietet somit Platz für bis zu 10 Millionen Protokolldateien. Der Datenbankmanager beginnt in den folgenden Situationen erneut bei `S0000000.LOG`:
  - Die Konfigurationsdatei der Datenbank wurde geändert, um die aktualisierende Recovery zu aktivieren.
  - Die Konfigurationsdatei der Datenbank wurde geändert, um die aktualisierende Recovery zu *inaktivieren*.
  - `S9999999.LOG` wurde verwendet.

Protokolldateinamen werden vom DB2-Datenbankmanager nach dem Wiederherstellen einer Datenbank (mit oder ohne aktualisierende Recovery) wiederverwendet. Der Datenbankmanager stellt sicher, dass bei der aktualisierenden Recovery kein falsches Protokoll verwendet wird. Falls der DB2-Datenbankmanager nach einer Wiederherstellungsoperation einen Protokolldateinamen wiederverwendet, werden die neuen Protokolldateien in separaten Verzeichnissen archiviert, damit es möglich ist, mehrere Protokolldateien mit demselben Namen zu archivieren. Die Position der Protokolldateien wird in der Datei des Recoveryprotokolls aufgezeichnet, sodass die Dateien während einer aktualisierenden Recovery angewendet werden können. Sie müssen sicherstellen, dass die richtigen Protokolle für die aktualisierende Recovery zur Verfügung stehen.

Wenn die aktualisierende Recovery erfolgreich durchgeführt wurde, wird das letzte bei der aktualisierenden Recovery verwendete Protokoll abgeschnitten und die Protokollierung mit dem nächsten sequenziellen Protokoll fortgesetzt. Dies hat zur Folge, dass jedes Protokoll im Protokollpfadverzeichnis mit einer Folgenummer, die größer als die des letzten für die aktualisierende Recovery verwendeten Protokolls ist, erneut verwendet wird. Im abgeschnittenen Protokoll werden alle auf die Position des Schnitts folgenden Einträge mit Nullen überschrieben. Stellen Sie sicher, dass Sie vor dem Aufrufen des Dienstprogramms zur aktualisierenden Recovery eine Kopie der Protokolle erstellen. (Sie können ein Benutzerexitprogramm aufrufen, um die Protokolle an eine andere Position zu kopieren.)

- Wenn eine Datenbank nicht aktiviert wurde (über den Befehl **ACTIVATE DATABASE**), schneidet der DB2-Datenbankmanager die laufende Protokolldatei ab, sobald für alle Anwendungen die Verbindung zur Datenbank getrennt worden ist. Beim nächsten Herstellen einer Verbindung zur Datenbank durch die Anwendung startet der DB2-Datenbankmanager die Protokollierung in einer neuen Protokolldatei. Wenn auf Ihrem System viele kleine Protokolldateien erstellt werden, kommt vielleicht die Verwendung des Befehls **ACTIVATE DATABASE** in Betracht. Sie sparen hierdurch nicht nur Systemaufwand zur Initialisierung der Datenbank, wenn Anwendungen eine Verbindung herstellen, sondern auch den Systemaufwand, eine große Protokolldatei zuzuordnen, die Datei abzuschneiden und eine neue große Protokolldatei zuzuordnen.
- Ein archiviertes Protokoll kann zwei oder mehreren verschiedenen *Protokollsequenzen* einer Datenbank zugeordnet sein, weil die Protokollnamen wieder verwendet werden (siehe Abb. 9 auf Seite 175). Wenn Sie beispielsweise Backup 2 wiederherstellen wollen, könnten hierfür zwei verschiedene Protokollfolgen verwendet werden. Wenn Sie während einer vollständigen Recovery der Datenbank die aktualisierende Recovery bis zu einem bestimmten Zeitpunkt durchführen und dann vor Erreichen des Protokollendes stoppen, erstellen Sie dadurch eine neue Protokollfolge. Es ist nicht möglich, die beiden Protokollfolgen zu kombinieren. Wenn Sie über ein Image eines Online-Backups verfügen, das sich über die erste Protokollfolge erstreckt, müssen Sie die aktualisierende Recovery mit der ersten Protokollfolge beenden.

  Wenn Sie nach der Recovery eine neue Protokollfolge erstellt haben, sind Tabellenbereichsbackups in den alten Protokollfolgen ungültig. Dies wird normalerweise beim Restore erkannt, das Restoredienstprogramm kann jedoch ein Tabellenbereichsbackup in einer alten Protokollfolge nicht erkennen, wenn die Restoreoperation für den Tabellenbereich unmittelbar auf eine Restoreoperation für die Datenbank folgt. Bis zur tatsächlichen aktualisierenden Recovery der Datenbank ist die zu verwendende Protokollfolge nicht bekannt. Wenn der Tabellenbereich zu einer alten Protokollfolge gehört, muss er von der Operation zur aktualisierenden Recovery des Tabellenbereichs „erfasst" werden. Eine Restoreoperation, bei der ein ungültiges Backup-Image verwendet wird, kann erfolg-

reich beendet werden, eine anschließende aktualisierende Recovery dieses Tabellenbereichs wird jedoch fehlschlagen, und der Tabellenbereich befindet sich weiter im Status *Restore anstehend*.

Nehmen Sie beispielsweise an, dass eine Backup-Operation auf Tabellenbereichsebene, Backup 3, zwischen `S0000013.LOG` und `S0000014.LOG` in der oberen Protokollfolge beendet wird (siehe Abb. 9). Wenn Sie unter Verwendung des Backup-Images auf Datenbankebene, Backup 2, einen Restore und eine aktualisierende Recovery durchführen wollen, müssen Sie die aktualisierende Recovery bis `S0000012.LOG` durchführen. Anschließend könnten Sie die aktualisierende Recovery entweder bis zum Ende der oberen Protokollfolge oder bis zum Ende der unteren (neueren) Protokollfolge fortsetzen. Wenn Sie die untere Protokollfolge verwenden, können Sie das Backup-Image auf Tabellenbereichsebene, Backup 3, nicht zum Restore und zur aktualisierenden Recovery verwenden.

Um unter Verwendung des Backup-Images auf Tabellenbereichsebene, Backup 3, für einen Tabellenbereich eine aktualisierende Recovery bis zum Protokollende auszuführen, müssen Sie das Backup-Image auf Datenbankebene, Backup 2, wiederherstellen und anschließend die obere Protokollfolge für eine aktualisierende Recovery verwenden. Nach dem Restore des Backup-Images auf Tabellenbereichsebene, Backup 3, können Sie eine aktualisierende Recovery bis zum Ende der Protokolle durchführen.

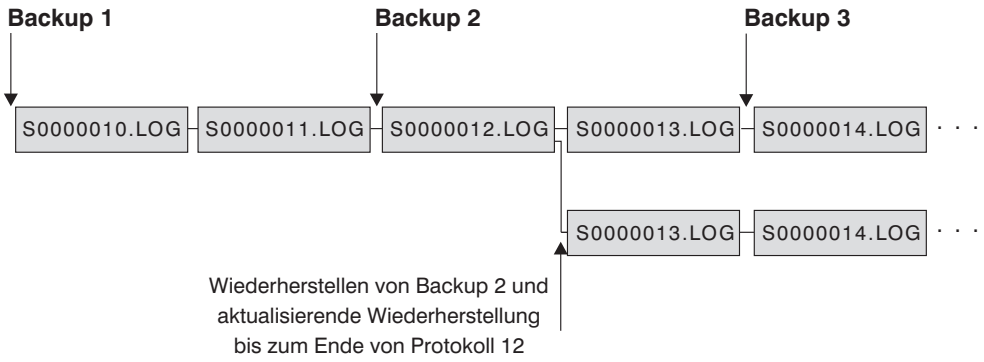


Abbildung 9. Erneutes Verwenden der Namen von Protokolldateien

Bedarfsgesteuerte Protokollarchivierung

Der IBM DB2-Server unterstützt das Schließen (und die Archivierung, falls aktiviert) der aktiven Protokolldatei bei einer wiederherstellbaren Datenbank zu jedem beliebigen Zeitpunkt. Dadurch haben Sie die Möglichkeit, eine Reihe von Protokolldateien bis zu einem bekannten Zeitpunkt zu sammeln und diese Dateien anschließend zur Aktualisierung einer Bereitschaftsdatenbank zu verwenden.

Sie können die bedarfsgesteuerte Protokollarchivierung einleiten, indem Sie den Befehl **ARCHIVE LOG** oder die Anwendungsprogrammierschnittstelle db2ArchiveLog aufrufen.

Protokollarchivierung mit db2tapemgr

Sie können das Dienstprogramm **db2tapemgr** zum Speichern von archivierten Protokolldateien auf Bandeinheiten verwenden. Das Dienstprogramm **db2tapemgr** kopiert die Protokolldateien von der Platte auf die angegebene Bandeinheit und aktualisiert die Datei des Recoveryprotokolls mit der neuen Speicherposition der kopierten Protokolldateien.

Konfiguration

Setzen Sie den Datenbankkonfigurationsparameter **logarchmeth1** für die Protokolldateien, die Sie auf Band kopieren möchten, auf die momentane Speicherposition auf der Platte. Das Dienstprogramm **db2tapemgr** liest den Wert **logarchmeth1**, um die zu kopierenden Protokolldateien zu suchen. In einer Umgebung mit partitionierten Datenbanken muss der Konfigurationsparameter **logarchmeth1** auf allen Datenbankpartitionen gesetzt werden, die zu kopierende Protokolldateien enthalten.

Das Dienstprogramm **db2tapemgr** verwendet nicht den Datenbankkonfigurationsparameter **logarchmeth2**.

Parameter STORE und DOUBLE STORE

Setzen Sie den Befehl **db2tapemgr** entweder mit dem Parameter **STORE** oder mit dem Parameter **DOUBLE STORE** ab, um archivierte Protokolle von der Platte auf Band zu übertragen.

- Der Parameter **STORE** speichert einige oder alle Protokolldateien aus dem Protokollarchivverzeichnis auf einer angegebenen Bandeinheit und löscht die Dateien von der Platte.
- Der Parameter **DOUBLE STORE** durchsucht die Verlaufsdatei, um zu prüfen, ob zuvor bereits Protokolle auf Band gespeichert wurden.
  - Wenn ein Protokoll zuvor nicht gespeichert wurde, speichert **db2tapemgr** die Protokolldatei auf Band, löscht sie jedoch nicht von der Platte.
  - Wenn ein Protokoll zuvor gespeichert wurde, speichert **db2tapemgr** die Protokolldatei auf Band und löscht sie von der Platte.

  Verwenden Sie **DOUBLE STORE**, wenn Sie Duplikatkopien Ihrer archivierten Protokolle auf Band und auf Platte behalten möchten oder wenn Sie dieselben Protokolle auf zwei verschiedenen Bändern speichern wollen.

Wenn Sie den Befehl **db2tapemgr** entweder mit dem Parameter **STORE** oder mit dem Parameter **DOUBLE STORE** absetzen, durchsucht das Dienstprogramm **db2tapemgr** zunächst die Verlaufsdatei nach Einträgen, in denen der Konfigurationsparameter **logarchmeth1** auf eine Plattenposition eingestellt ist. Wenn es feststellt, dass Dateien, die sich auf Platte befinden sollten, dort nicht zu finden sind, gibt es eine Warnung aus. Wenn das Dienstprogramm **db2tapemgr** keine Protokolldateien zum Speichern findet, stoppt es die Ausführung und gibt eine Nachricht aus, um Sie zu informieren, dass nichts zu tun ist.

RETRIEVE-Parameter

Setzen Sie den Befehl **db2tapemgr** mit dem Parameter **RETRIEVE** ab, um Dateien vom Band auf Platte zu übertragen.

- Verwenden Sie den Parameter **RETRIEVE ALL LOGS** oder **LOGS** *n* **TO** *n*, um alle archivierten Protokolle, die den von Ihnen angegebenen Kriterien entsprechen, abzurufen und auf Platte zu kopieren.
- Verwenden Sie den Parameter **RETRIEVE FOR ROLLFORWARD TO POINT-IN-TIME**, um alle archivierten Protokolle, die zur Durchführung einer aktualisierenden Recoveryoperation erforderlich sind, abzurufen und auf Platte zu kopieren.
- Verwenden Sie den Parameter **RETRIEVE HISTORY FILE**, um die Verlaufsdatei vom Band abzurufen und auf Platte zu kopieren.

Verhalten

- Wenn das Dienstprogramm **db2tapemgr** Protokolldateien auf Platte findet, liest es die Bandkopfdaten, um sicherzustellen, dass die Protokolldateien auf das Band geschrieben werden können. Außerdem aktualisiert es die Verlaufsdatei für die Dateien, die sich momentan auf Band befinden. Wenn die Aktualisierung fehlschlägt, stoppt es die Ausführung und zeigt eine Fehlernachricht an.
- Wenn das Band beschreibbar ist, kopiert das Dienstprogramm **db2tapemgr** die Protokolle auf Band. Nach dem Kopieren der Dateien werden die Protokolldateien von der Platte gelöscht. Zum Schluss kopiert das Dienstprogramm **db2tapemgr** die Verlaufsdatei auf das Band und löscht sie von der Platte.
- Das Dienstprogramm **db2tapemgr** hängt keine Protokolldateien an ein Band an. Wenn eine Speicheroperation nicht das gesamte Band füllt, kann der verbleibende freie Bandspeicherplatz nicht mehr genutzt werden.
- Das Dienstprogramm **db2tapemgr** speichert Protokolldateien nur einmal auf einem gegebenen Band. Durch diese Einschränkung sollen Probleme, die mit dem Schreiben auf Banddatenträgern verbunden sind, zum Beispiel eine Streckung des Bandes, vermieden werden.
- In einer Umgebung mit partitionierten Datenbanken lässt sich das Dienstprogramm **db2tapemgr** nur für jeweils eine Datenbankpartition ausführen. Sie müssen den entsprechenden Befehl für jede Datenbankpartition ausführen, indem Sie die Datenbankpartitionsnummer mit dem Parameter **ON DBPARTITIONNUM** des Befehls **db2tapemgr** angeben. Darüber hinaus müssen Sie sicherstellen, dass jede Datenbankpartition Zugriff auf eine Bandeinheit besitzt.
- Das Dienstprogramm **db2tapemgr** wird in DB2 pureScale-Umgebungen nicht unterstützt.

Beispiele

Das folgende Beispiel zeigt, wie der Befehl **db2tapemgr** verwendet wird, um alle Protokolldateien aus dem primären Archivprotokollpfad für die Datenbank SAMPLE in der Datenbankpartition mit der Nummer 0 auf einer Bandeinheit zu speichern und die Protokolldateien aus dem Archivprotokollpfad zu entfernen:

```
db2tapemgr db sample on dbpartitionnum 0 store on /dev/rmt0.1 all logs
```

Das folgende Beispiel zeigt, wie die ersten 10 Protokolldateien aus dem primären Archivprotokollpfad auf einer Bandeinheit gespeichert und aus dem Archivprotokollpfad entfernt werden:

```
db2tapemgr db sample on dbpartitionnum store on /dev/rmt0.1 10 logs
```

Das folgende Beispiel zeigt, wie die ersten 10 Protokolldateien aus dem primären Archivprotokollpfad auf einer Bandeinheit gespeichert werden, anschließend auf einem zweiten Band gespeichert und schließlich aus dem Archivprotokollpfad entfernt werden:

```
db2tapemgr db sample on dbpartitionnum double store on /dev/rmt0.1 10 logs
db2tapemgr db sample on dbpartitionnum double store on /dev/rmt1.1 10 logs
```

Das folgende Beispiel zeigt, wie alle Protokolldateien von einem Band in ein Verzeichnis abgerufen werden:

```
db2tapemgr db sample on dbpartitionnum retrieve all logs from /dev/rmt1.1
        to /home/dbuser/archived_logs
```

Archivierung und Abruf von Protokolldateien mithilfe von Benutzerexitprogrammen automatisieren

Sie können zur Automatisierung der Archivierung und des Abrufs von Protokolldateien ein Benutzerexitprogramm erstellen, das vom DB2-Datenbankmanager aufgerufen wird, um die Archivierungs- oder Abrufoperation auszuführen.

Wenn der DB2-Datenbankmanager Ihr Benutzerexitprogramm aufruft, geschieht Folgendes:

- Der Datenbankmanager übergibt die Steuerung an das Benutzerexitprogramm,
- der Datenbankmanager übergibt Parameter an das Benutzerexitprogramm und
- das Benutzerexitprogramm gibt bei seiner Beendigung einen Rückkehrcode an den Datenbankmanager zurück.

Konfiguration

Bevor Sie ein Benutzerexitprogramm zur Archivierung oder zum Abruf von Protokolldateien aufrufen, muss der Datenbankkonfigurationsparameter **logarchmeth1** auf USEREXIT gesetzt werden. Dies ermöglicht außerdem die aktualisierende Recovery der Datenbank.

Voraussetzungen für Benutzerexitprogramme

- Die ausführbare Datei für Ihr Benutzerexitprogramm muss auf den Namen db2uext2 lauten.
- Benutzerexitprogramme müssen Protokolldateien aus dem Pfad für aktive Protokolldateien in den Archivprotokollpfad kopieren, nicht versetzen. Entfernen Sie keine Protokolldateien aus dem Pfad für aktive Protokolldateien. Wenn Sie Protokolldateien aus dem Pfad für aktive Protokolldateien entfernen, kann Ihre DB2-Datenbank nach einem Fehler oder Ausfall möglicherweise nicht mehr erfolgreich wiederhergestellt werden.

  Für die DB2-Datenbank müssen die Protokolldateien während der Recovery im Pfad für aktive Protokolldateien vorhanden sein. Der DB2-Datenbankserver entfernt archivierte Protokolldateien aus dem Pfad der aktiven Protokolldateien, sobald diese Protokolldateien für die Recovery nicht mehr erforderlich sind.
- Benutzerexitprogramme müssen Fehlerbedingungen handhaben können. Dies ist notwendig, da der DB2-Datenbankmanager nur eine begrenzte Anzahl an Rückkehrbedingungen bearbeiten kann.

  Siehe hierzu „Fehlerbehandlung für Benutzerexits” auf Seite 180.
- Jede DB2-Datenbankmanagerinstanz kann lediglich ein Benutzerexitprogramm aufrufen. Daher müssen Sie für jede Operation, die Ihr Benutzerexitprogramm möglicherweise ausführen soll, einen Abschnitt entwickeln.

Benutzerexitbeispielprogramme

Benutzerexitbeispielprogramme sind für alle unterstützten Plattformen verfügbar. Sie können diese Programme modifizieren, um sie an Ihre spezifischen Anforderungen anzupassen. Die Beispielprogramme sind gut kommentiert und helfen Ihnen, sie sehr effektiv zu nutzen.

Sie müssen beachten, dass Benutzerexitprogramme Protokolldateien aus dem Pfad für aktive Protokolldateien in den Archivprotokollpfad *kopieren* müssen. Entfernen Sie keine Protokolldateien aus dem Pfad für aktive Protokolldateien. (Dies könnte bei der Datenbankrecovery Fehler verursachen.) DB2 entfernt archivierte Protokolldateien aus dem Pfad der aktiven Protokolldateien, sobald diese Protokolldateien für die Recovery nicht mehr erforderlich sind.

Im Folgenden werden die mit DB2 Data Server ausgelieferten Benutzerexitbeispielprogramme beschrieben.

- **UNIX-Betriebssysteme**

  Die Benutzerexitbeispielprogramme für DB2 Data Server für UNIX-Betriebssysteme befinden sich im Unterverzeichnis `sqllib/samples/c`. Obwohl die Beispielprogramme in der Programmiersprache C codiert sind, können Sie Ihr Benutzerexitprogramm auch in einer anderen Programmiersprache schreiben.

  Das Benutzerexitprogramm muss eine ausführbare Datei sein, deren Name `db2uext2` lautet.

  Die folgenden vier Benutzerexitbeispielprogramme für UNIX-Betriebssysteme sind verfügbar:

  – `db2uext2.ctsm`

    Dieses Beispielprogramm verwendet Tivoli Storage Manager, um Datenbankprotokolldateien zu archivieren und abzurufen.

  – `db2uext2.ctape`

    Dieses Beispielprogramm verwendet Banddatenträger, um Datenbankprotokolldateien zu archivieren und abzurufen.

  – `db2uext2.cdisk`

    Dieses Beispielprogramm verwendet den Betriebssystembefehl COPY und Platten, um Datenbankprotokolldateien zu archivieren und abzurufen.

  – `db2uxt2.cxbsa`

    Dieses Beispiel arbeitet mit XBSA Draft 0.8 von X/Open Group. Sie können damit Datenbankprotokolldateien archivieren und abrufen. Dieses Beispielprogramm wird nur unter AIX unterstützt.

- **Windows-Betriebssysteme**

  Die Benutzerexitbeispielprogramme für DB2 Data Server für Windows-Betriebssysteme befinden sich im Unterverzeichnis `sqllib\samples\c`. Obwohl die Beispielprogramme in der Programmiersprache C codiert sind, können Sie Ihr Benutzerexitprogramm auch in einer anderen Programmiersprache schreiben.

  Das Benutzerexitprogramm muss eine ausführbare Datei sein, deren Name `db2uext2` lautet.

  Für Windows-Betriebssysteme sind zwei Benutzerexitbeispielprogramme verfügbar:

  – `db2uext2.ctsm`

    Dieses Beispielprogramm verwendet Tivoli Storage Manager, um Datenbankprotokolldateien zu archivieren und abzurufen.

  – `db2uext2.cdisk`

Dieses Beispielprogramm verwendet den Betriebssystembefehl COPY und Platten, um Datenbankprotokolldateien zu archivieren und abzurufen.

Aufrufformat des Benutzerexitprogramms

Wenn der DB2-Datenbankmanager ein Benutzerexitprogramm aufruft, übergibt er einen Satz von Parametern (mit dem Datentyp CHAR) an das Programm.

Befehlssyntax

```
     db2uext2 -OS<bs> -RL<db2rel> -RQ<anforderung>  -DB<dbname>
     -NN<knotennr> -LP<protokollpfad> -LN<protokollname> -AP<tsmkennwort>
  -SP<startseite> -LS<protokollgröße>
```

bs Gibt die Plattform an, auf der die Instanz aktiv ist. Gültige Werte: `AIX`, `Solaris`, `HP-UX`, `SCO`, `Linux` und `NT`.

db2rel Gibt den Releasestand von DB2 an. Beispiel: `SQL07020`

anforderung
Gibt den Anforderungstyp an. Gültige Werte: `ARCHIVE` und `RETRIEVE`.

dbname
Gibt einen Datenbanknamen an.

nodenum
Gibt die Nummer des lokalen Knotens an, z. B. 5.

logpath
Gibt den vollständig qualifizierten Pfad zu den Protokolldateien an. Der Pfad muss das abschließende Pfadtrennzeichen enthalten. Beispiel: `/u/database/log/path/` oder `d:\logpath\`.

protokollname
Gibt den Namen der Protokolldatei an, die archiviert oder abgerufen werden soll, z. B. `S0000123.LOG`.

tsmkennwort
Gibt das TSM-Kennwort an. (Wurde zuvor für den Datenbankkonfigurationsparameter *tsm_password* ein Wert angegeben, wird dieser Wert an das Benutzerexitprogramm übergeben.)

startseite
Gibt die Zahl der relativen 4-KB-Adressseiten der Einheit an, auf welcher der Protokollspeicherbereich beginnt.

protokollgröße
Gibt die Größe des Protokollspeicherbereichs in 4-KB-Seiten an. Dieser Parameter ist nur gültig, wenn Sie zum Protokollieren eine unformatierte Einheit einsetzen.

Fehlerbehandlung für Benutzerexits

Wenn Sie ein Benutzerexitprogramm zur Automatisierung der Archivierung und des Abrufs von Protokolldateien erstellen, übergibt das Benutzerexitprogramm Rückkehrcodes an den DB2-Datenbankmanager, der das Programm aufgerufen hat.

Der DB2-Datenbankmanager kann nur eine begrenzte Liste spezifischer Fehlercodes bearbeiten. Ihr Benutzerexitprogramm trifft jedoch möglicherweise auf viele verschiedene Fehlerbedingungen wie z. B. Betriebssystemfehler. Das Benutzerexitprogramm muss die auftretenden Fehlerbedingungen den Fehlercodes zuordnen, die der Datenbankmanager bearbeiten kann.

Tabelle 8 zeigt die Codes, die von einem Benutzerexitprogramm zurückgegeben werden können, und beschreibt, wie der Datenbankmanager diese Codes interpretiert. Wenn ein Rückkehrcode in dieser Tabelle nicht aufgeführt ist, wird er so behandelt, als hätte er den Wert 32.

Tabelle 8. Rückkehrcodes von Benutzerexitprogrammen

Rückkehrcode	Erläuterung
0	Erfolgreich beendet.
4	Temporärer Ressourcenfehler trat auf.[a]
8	Bedienereingriff ist erforderlich.[a]
12	Hardwarefehler.[b]
16	Fehler bei Benutzerexitprogramm oder einer vom Programm verwendeten Softwarefunktion.[b]
20	Fehler bei mindestens einem Parameter, der an das Benutzerexitprogramm übergeben wurde. Prüfen Sie, ob das Benutzerexitprogramm die angegebenen Parameter korrekt verarbeitet.[b]
24	Das Benutzerexitprogramm wurde nicht gefunden. [b]
28	Ein Fehler trat auf, der durch einen E/A-Fehler oder durch das Betriebssystem verursacht wurde.[b]
32	Das Benutzerexitprogramm wurde vom Benutzer beendet.[b]
255	Das Benutzerexitprogramm verursachte einen Fehler, da es die Bibliotheksdatei für die ausführbare Datei nicht laden konnte.[c]

Tabelle 8. Rückkehrcodes von Benutzerexitprogrammen (Forts.)

Rückkehrcode	Erläuterung
[a] Bei Archivierungs- und Abrufanforderungen bewirkt ein Rückkehrcode 4 oder 8 eine Wiederholung nach fünf Minuten. Wenn das Benutzerexitprogramm auf Abrufanforderungen für dieselbe Protokolldatei weiterhin 4 oder 8 zurückgibt, führt DB2 so lange Wiederholungen aus, bis die Operation erfolgreich ist. (Dies gilt für aktualisierende Recoverys oder für Aufrufe der Anwendungsprogrammierschnittstelle **db2ReadLog**, die vom Replikationsdienstprogramm verwendet wird.) [b] Aufrufe für Benutzerexitprogramme werden fünf Minuten lang ausgesetzt. Während dieser Zeit werden alle Anforderungen ignoriert, einschließlich der Anforderung, die die Fehlerbedingung verursachte. Nach dieser fünfminütigen Aussetzung wird die nächste Anforderung verarbeitet. Wenn diese Anforderung ohne Fehler verarbeitet wurde, wird die Verarbeitung neuer Benutzerexitanforderungen fortgesetzt, und DB2 setzt die Archivierungsanforderung, die fehlgeschlagen oder vorher ausgesetzt worden war, erneut ab. Wenn während der Wiederholung ein Rückkehrcode größer als acht generiert wird, werden Anforderungen für weitere fünf Minuten ausgesetzt. Die fünfminütigen Aussetzungen werden solange fortgesetzt, bis der Fehler behoben ist oder bis die Datenbank gestoppt und erneut gestartet wurde. Nachdem alle Anwendungen von der Datenbank getrennt wurden, setzt DB2 eine Archivierungsanforderung für alle Protokolldateien ab, die vorher möglicherweise nicht erfolgreich archiviert wurden. Wenn das Benutzerexitprogramm Protokolldateien nicht archivieren kann, füllt sich die Platte möglicherweise mit Protokolldateien an, und die Leistung verschlechtert sich. Wenn die Platte voll ist, nimmt der Datenbankmanager keine weiteren Anwendungsanforderungen für Datenbankaktualisierungen mehr entgegen. Wenn das Benutzerexitprogramm zum Abrufen von Protokolldateien aufgerufen wurde, wird die aktualisierende Recovery ausgesetzt, jedoch nicht gestoppt, es sei denn, die Option ROLLFORWARD STOP wurde angegeben. Wenn Sie die Option STOP nicht angegeben haben, können Sie den Fehler beheben und die Recovery wieder aufnehmen. [c] Wenn das Benutzerexitprogramm den Fehlercode 255 zurückgibt, kann das Programm die Bibliotheksdatei für die ausführbare Datei wahrscheinlich nicht laden. Rufen Sie das Benutzerexitprogramm manuell auf, um dies zu prüfen. Weitere Informationen werden angezeigt. **Anmerkung:** Bei Archivierungs- und Abrufoperationen wird ein Alert für alle Rückkehrcodes außer 0 und 4 ausgegeben. Der Alert enthält den Rückkehrcode vom Benutzerexitprogramm und eine Kopie der Eingabeparameter, die an das Benutzerexitprogramm übergeben wurden.	

Zuordnen und Entfernen von Protokolldateien

Eine für die Recovery nach einem Systemabsturz benötigte Protokolldatei wird als aktive Protokolldatei bezeichnet. Sofern nicht die Endlosprotokollierung aktiviert ist, werden die im Datenbankprotokollverzeichnis vorhandenen Protokolldateien nicht entfernt, wenn sie eventuell zur Recovery nach einem Systemabsturz benötigt werden.

Wurde die Endlosprotokollierung aktiviert und muss Speicherbereich für weitere aktive Protokolldateien verfügbar sein, archiviert der Datenbankmanager eine aktive Protokolldatei und benennt sie um, um eine neue aktive Protokolldatei zu erstellen. Falls eine Recovery nach Systemabsturz notwendig ist, wenn die Endlosprotokollierung aktiviert ist, müssen die Protokolldateien möglicherweise aus dem Archivprotokollpfad abgerufen werden, um die Recovery nach Systemabsturz abzuschließen.

Wenn der Datenbankkonfigurationsparameter **logarchmeth1** nicht auf OFF gesetzt ist, wird eine volle Protokolldatei nur dann entfernt, wenn sie nicht länger zur Re-

covery nach einem Systemabsturz benötigt wird, es sei denn, die Endlosprotokollierung ist aktiviert. In diesem Fall können die Protokolldateien stattdessen in den Archivprotokollpfad verschoben werden.

Wenn **logarchmeth1** oder **logarchmeth2** auf einen anderen Wert als OFF, LOGRETAIN oder USEREXIT gesetzt ist, kann die Komprimierung archivierter Protokolldateien aktiviert werden, um die Menge des erforderlichen Plattenspeicherplatzes für archivierte Protokolldateien zu reduzieren.

Der Prozess der Zuordnung neuer Protokolldateien und des Entfernens alter Protokolldateien ist von den Einstellungen der Datenbankkonfigurationsparameter **logarchmeth1** und **logarchmeth2** abhängig:

logarchmeth1 und logarchmeth2 sind auf OFF gesetzt
: Es wird Umlaufprotokollierung verwendet. Bei der Umlaufprotokollierung wird die Recovery nach einem Systemabsturz unterstützt, die aktualisierende Recovery jedoch nicht.

  Während der Umlaufprotokollierung werden außer sekundären Protokollen keine neuen Protokolldateien erstellt, und alte Protokolldateien werden nicht gelöscht. Die Protokolldateien werden umlaufend verwendet. Das heißt, wenn die letzte Protokolldatei voll ist, schreibt der Datenbankmanager wieder in die erste Protokolldatei.

  Wenn alle Protokolldateien aktiv sind und der Prozess der Umlaufprotokollierung nicht erneut in die erste Protokolldatei schreiben kann, gelten alle Protokolldateien als voll. Wenn alle primären Protokolldateien aktiv und voll sind, werden sekundäre Protokolldateien erstellt. Sekundäre Protokolldateien werden gelöscht, wenn der von ihnen belegte Speicherbereich für die aktiven Protokolldateien benötigt wird.

logarchmeth1 oder logarchmeth2 wird auf LOGRETAIN gesetzt
: Es wird Archivprotokollierung verwendet. Die Datenbank ist eine wiederherstellbare Datenbank. Sowohl die Recovery nach einem Systemabsturz als auch die aktualisierende Recovery sind aktiviert. Der Datenbankmanager verwaltet nicht die Protokolldateien. Nachdem Sie die Protokolldateien archiviert haben, müssen Sie sie aus dem Pfad für aktive Protokolldateien löschen, sodass der Plattenspeicherplatz für neue Protokolldateien verwendet werden kann. Welche der Protokolldateien archivierte Protokolldateien sind, können Sie am Wert des Datenbankkonfigurationsparameters **loghead** erkennen. Dieser Parameter gibt das Protokoll mit der niedrigsten Nummer an, das aktiv ist. Bei Protokollen mit Folgenummern, die niedriger als der Wert von **loghead** sind, handelt es sich um nicht aktive Protokolldateien, die archiviert und entfernt werden können.

logarchmeth1 oder logarchmeth2 ist auf einen anderen Wert als OFF oder LOGRETAIN gesetzt
: Es wird Archivprotokollierung verwendet. Die Datenbank ist eine wiederherstellbare Datenbank. Sowohl die Recovery nach einem Systemabsturz als auch die aktualisierende Recovery sind aktiviert. Wenn eine Protokolldatei voll ist, wird sie automatisch vom Datenbankmanager archiviert.

  Protokolldateien werden nicht gelöscht. Wenn eine neue Protokolldatei benötigt wird, aber keine verfügbar ist, wird stattdessen eine Archivprotokolldatei umbenannt und erneut verwendet. Eine Archivprotokolldatei wird nicht umbenannt oder gelöscht, nachdem sie geschlossen und in das Verzeichnis für archivierte Protokolldateien kopiert wurde. Der Datenbankmanager wartet, bis eine neue Protokolldatei benötigt wird, und benennt dann die älteste Archivprotokolldatei um. Eine Protokolldatei, die während

der Recovery in das Datenbankverzeichnis versetzt wurde, wird während des Recoveryprozesses entfernt, sobald sie nicht mehr benötigt wird.

Wenn beim Archivieren der Protokolldateien ein Fehler auftritt, wird die Archivierung für den vom Datenbankkonfigurationsparameter **archretrydelay** angegebenen Zeitraum ausgesetzt. Sie können auch mit dem Datenbankkonfigurationsparameter **numarchretry** angeben, wie oft der Datenbankmanager versuchen soll, eine Protokolldatei im primären oder sekundären Archivverzeichnis zu archivieren, bevor es versucht, die Protokolldateien im Funktionsübernahmeverzeichnis (das vom Datenbankkonfigurationsparameter **failarchpath** angegeben wird) zu archivieren. **Numarchretry** wird nur verwendet, wenn der Datenbankkonfigurationsparameter **failarchpath** definiert ist. Wenn **numarchretry** auf `0` gesetzt ist, wird der Datenbankmanager kontinuierlich versuchen, die Protokolldateien im primären oder sekundären Protokollpfad zu archivieren.

Die einfachste Möglichkeit, alte Protokolldateien zu entfernen, ist ein Neustart der Datenbank. Nach dem erneuten Start der Datenbank befinden sich im Datenbankverzeichnis nur noch neue Protokolldateien und Protokolldateien, die vom Datenbankmanager nicht archiviert werden konnten.

Nach einem erneuten Start der Datenbank entspricht die Mindestanzahl Protokolle im Datenbankprotokollverzeichnis der Anzahl primärer Protokolle, die mit dem Datenbankkonfigurationsparameter **logprimary** konfiguriert werden kann. Im Protokollverzeichnis können sich mehr Protokolldateien als die maximale Anzahl primärer Protokolle befinden. Diese Bedingung tritt auf, wenn zum Zeitpunkt des Herunterfahrens der Datenbank die Anzahl leerer Protokolle im Protokollverzeichnis größer ist als der Wert des Konfigurationsparameters **logprimary** zum Zeitpunkt des erneuten Datenbankstarts. Dies ist der Fall, wenn der Wert des Konfigurationsparameters **logprimary** zwischen dem Herunterfahren und dem erneuten Start der Datenbank geändert wird oder wenn sekundäre Protokolle zugeordnet, aber nicht benutzt wurden.

Wenn beim erneuten Start einer Datenbank die Anzahl leerer Protokolle kleiner als die vom Konfigurationsparameter **logprimary** angegebene Anzahl primärer Protokolle ist, werden zusätzliche Protokolldateien zugeordnet, um die Differenz auszugleichen. Falls im Datenbankverzeichnis mehr leere Protokolle als primäre Protokolle verfügbar sind, kann die Datenbank mit der Anzahl der im Datenbankverzeichnis vorhandenen, leeren Protokolle erneut gestartet werden. Nach dem Herunterfahren der Datenbank verbleiben die erstellten, sekundären Protokolldateien bei einem erneuten Start der Datenbank im Pfad für aktive Protokolldateien.

Einschließen von Protokolldateien in ein Backup-Image

Wenn Sie eine Online-Backup-Operation ausführen, können Sie angeben, ob die für einen Restore und eine Recovery erforderlichen Protokolldateien in das Backup-Image aufgenommen werden.

Dies bedeutet, dass Sie die Protokolldateien nicht separat senden oder selbst packen müssen, wenn Sie Backup-Images an einen Standort senden, an dem eine Recovery nach einem Katastrophenfall durchgeführt werden muss. Außerdem müssen Sie so nicht selbst entscheiden, welche Protokolldateien erforderlich sind, um die Konsistenz eines Online-Backups zu gewährleisten. Dies bietet einen gewissen Schutz vor dem Löschen von Protokolldateien, die für eine erfolgreiche Recovery erforderlich sind.

Geben Sie zur Verwendung dieser Funktion die Option **INCLUDE LOGS** des Befehls **BACKUP DATABASE** an. Wenn Sie diese Option angeben, schneidet das Backup-Dienstprogramm die aktive Protokolldatei ab und kopiert die erforderlichen Protokollspeicherbereiche in das Backup-Image.

Verwenden Sie zum Wiederherstellen der Protokolldateien aus einem Backup-Image die Option **LOGTARGET** des Befehls **RESTORE DATABASE**, und geben Sie einen vollständig qualifizierten, auf dem DB2-Server vorhandenen Pfad an. Das Datenbankrestoredienstprogramm schreibt anschließend die Protokolldateien aus dem Image in den Zielpfad. Wenn eine Protokolldatei desselben Namens im Zielpfad vorhanden ist, schlägt die Restoreoperation fehl, und es wird ein Fehler zurückgegeben. Wird die Option **LOGTARGET** nicht angegeben, werden keine Protokolldateien aus dem Backup-Image wiederhergestellt.

Wenn die Option **LOGTARGET** angegeben wird und das Backup-Image keine Protokolldateien enthält, wird ein Fehler zurückgegeben, bevor versucht wird, Tabellenbereichsdaten wiederherzustellen. Die Restoreoperation schlägt ebenfalls fehl, wenn ein ungültiger oder schreibgeschützter Pfad angegeben wird. Wenn während des Restores einer Datenbank oder eines Tabellenbereichs, für den die Option **LOGTARGET** angegeben wurde, mindestens eine Protokolldatei nicht extrahiert werden kann, schlägt die Restoreoperation fehl, und es wird ein Fehler zurückgegeben.

Sie können auch nur die in einem Backup-Image gespeicherten Protokolldateien wiederherstellen. Geben Sie dazu die Option **LOGS** mit der Option **LOGTARGET** des Befehls **RESTORE DATABASE** an. Wenn beim Restore von Protokolldateien in diesem Modus Fehler auftreten, schlägt die Restoreoperation fehl, und es wird ein Fehler zurückgegeben.

Bei einer Operation zum automatischen inkrementellen Restore werden nur die Protokolle aus dem Backup-Image abgerufen, die sich im Zielimage der Restoreoperation befinden. Protokolle, die sich in temporären Images befinden, auf die während des inkrementellen Restores verwiesen wird, werden nicht aus diesen Backup-Images extrahiert. Wenn Sie beim manuellen inkrementellen Restore eines Backup-Images, das Protokolldateien enthält, ein Protokollzielverzeichnis angeben, werden die in diesem Backup-Image enthaltenen Protokolldateien wiederhergestellt.

Wenn Sie eine aktualisierende Recovery einer Datenbank ausführen, die aus einem Online-Backup-Image, das Protokolldateien enthält, mit Restore wiederherstellt wurde, empfangen Sie möglicherweise den Fehler SQL1268N. Dieser Fehler gibt an, dass die aktualisierende Recovery aufgrund eines Fehlers beim Abruf eines Protokolls gestoppt wurde. Dieser Fehler wird generiert, wenn das Zielsystem, auf dem Sie den Restore des Backup-Images versuchen, keinen Zugriff auf die Einrichtung hat, die vom Quellensystem zum Archivieren der Transaktionsprotokolle verwendet wird.

Wenn Sie die Option **INCLUDE LOGS** des Befehls **BACKUP DATABASE** beim Backup einer Datenbank angeben und dann eine Restoreoperation und eine aktualisierende Recoveryoperation durchführen, die dieses Backup-Image verwenden, sucht DB2 bei der aktualisierenden Recovery auch dann noch nach weiteren Transaktionsprotokollen, wenn das Backup-Image Protokolle enthält. Es gehört zur Standardfunktionsweise der aktualisierenden Recovery, dass die Suche nach weiteren Transaktionsprotokollen fortgesetzt wird, bis keine Protokolle mehr gefunden werden. Es ist möglich, mehr als eine Protokolldatei mit der gleichen Zeitmarke zu haben. Infolgedessen wie DB2 nicht gestoppt, sobald die erste Zeitmarke angetroffen wird, die mit dem für die aktualisierende Recovery der Datenbank angegebenen Zeitpunkt

übereinstimmt, da es noch weitere Protokolldateien geben könnte, die ebenfalls diese Zeitmarke haben. Vielmehr setzt DB2 die Suche im Transaktionsprotokoll fort, bis eine Zeitmarke gefunden wird, die größer ist als der angegebene Zeitpunkt.

Wenn keine weiteren Protokolle zu finden sind, wird die aktualisierende Recoveryoperation erfolgreich beendet. Wenn während der Suche nach weiteren Transaktionsprotokolldateien jedoch ein Fehler auftritt, wird die Fehlernachricht SQL1268N zurückgegeben. Der Fehler SQL1268N kann dadurch verursacht werden, dass beim anfänglichen Restore bestimmte Konfigurationsparameter der Datenbank zurückgesetzt oder überschrieben wurden. Drei dieser Datenbankkonfigurationsparameter sind die TSM-Parameter **tsm_nodename**, **tsm_owner** und **tsm_password**. Sie werden alle auf NULL zurückgesetzt. Zur Durchführung einer aktualisierenden Recovery bis zum Ende der Protokolle müssen Sie vor der aktualisierenden Recoveryoperation die Datenbankkonfigurationsparameter so ändern, dass sie dem Quellensystem entsprechen. Alternativ können Sie auch die Option **NORETRIEVE** im Befehl **ROLLFORWARD DATABASE** angeben. Dadurch wird verhindert, dass das DB2-Datenbanksystem versucht, potenziell fehlende Transaktionsprotokolle von anderen Positionen abzurufen.

Anmerkung:

1. Diese Funktion wird für Offline-Backups nicht unterstützt.
2. Wenn in ein Online-Backup-Image Protokolldateien eingeschlossen werden, kann das daraus resultierende Image nicht in Releases der DB2-Datenbank wiederhergestellt werden, die älter als Version 8.2 sind.

Zufälligen Verlust von Protokolldateien verhindern

In Situationen, in denen Sie eine Datenbank löschen oder eine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt durchführen müssen, verlieren Sie möglicherweise Protokolldateien, die für zukünftige Recoveryoperationen erforderlich sind. In solchen Fällen ist es wichtig, Kopien aller im aktuellen Protokollverzeichnispfad der Datenbank vorhandenen Protokolle zu erstellen.

Betrachten Sie die folgenden Szenarien:

- Wenn Sie vor einer RESTORE-Operation eine Datenbank löschen wollen, müssen Sie die im Pfad für aktive Protokolldateien vorhandenen Protokolldateien speichern, bevor Sie den Befehl **DROP DATABASE** absetzen. Nach dem Wiederherstellen der Datenbank werden diese Protokolldateien möglicherweise für die aktualisierende Recovery benötigt, da einige dieser Protokolldateien eventuell vor dem Löschen der Datenbank nicht archiviert wurden. In der Regel ist es nicht erforderlich, eine Datenbank vor dem Absetzen des Befehls **RESTORE** zu löschen. Es ist jedoch möglich, dass Sie die Datenbank löschen müssen (oder die Datenbank in einer Datenbankpartition löschen müssen, indem Sie den Parameter **AT DBPARTITIONNUM** des Befehls **DROP DATABASE** angeben), da sie beschädigt wurde, sodass der Befehl **RESTORE** fehlschlägt. Sie können eine Datenbank auch löschen, um vor der Restoreoperation einen Neuanfang zu machen.
- Wenn Sie eine Datenbank bis zu einem bestimmten Zeitpunkt aktualisierend wiederherstellen, werden alle Protokolldaten nach der von Ihnen angegebenen Zeitmarke überschrieben. Wenn Sie nach Beendigung der aktualisierenden Recovery bis zu einem bestimmten Zeitpunkt und der erneuten Verbindungsherstellung zur Datenbank feststellen, dass Sie die Datenbank eigentlich bis zu einem späteren Zeitpunkt wiederherstellen müssen, können Sie dies nicht mehr nach-

holen, da die bereits Protokolle überschrieben wurden. Zwar ist es möglich, dass die ursprünglichen Protokolldateien archiviert wurden; DB2 könnte jedoch ein Benutzerexitprogramm aufrufen, um die neu generierten Protokolldateien automatisch zu archivieren. Je nachdem, wie das Benutzerexitprogramm geschrieben ist, könnten hierbei die ursprünglichen Protokolldateien im Archivprotokollverzeichnis überschrieben werden. Auch wenn sich die ursprünglichen Protokolldateien zusammen mit den neuen Protokolldateien im Archivprotokollverzeichnis befinden (als verschiedene Versionen derselben Dateien), müssen Sie möglicherweise bestimmen, welche Protokolle nun für spätere Recoveryoperationen verwendet werden sollen.

Beeinträchtigung der Verfügbarkeit durch Verwaltungs- und Wartungsaktivitäten minimieren

In Ihrer DB2-Datenbanklösung werden Sie für Ihre Geschäftsaktivitäten auch Verwaltungs- und Wartungsaktivitäten wie Software- oder Hardware-Upgrades, Optimierung der Datenbankleistung, Datenbankbackups, Statistikerfassung und Überwachung durchführen müssen.

Sollen die Auswirkungen dieser Verwaltungs- und Wartungsaktivitäten auf die Verfügbarkeit Ihrer Lösung minimiert werden, ist sorgfältige Terminplanung für die Offlineverwaltung notwendig, sowie der Einsatz von DB2-Komponenten und -Funktionalitäten, um die Beeinträchtigung der Verfügbarkeit durch die Offlineverwaltung reduzieren.

Vorbereitende Schritte

Die beiden folgenden Aufgaben sind zu erledigen, bevor Sie die anschließenden Schritte ausführen können, um die Beeinträchtigung der Verfügbarkeit Ihrer DB2-Datenbanklösung durch Verwaltungs- und Wartungsaktivitäten zu minimieren:

- Konfigurieren der automatischen Verwaltung und
- Installieren der High Availability Disaster Recovery-Funktion (HADR-Funktion).

Vorgehensweise

1. Die automatische Verwaltung erledigt die Verwaltungsaktivitäten für Sie.

   Die DB2-Datenbank kann viele Datenbankverwaltungsaktivitäten automatisieren. Sobald die automatische Verwaltung konfiguriert wurde, finden die Verwaltungsaktivitäten statt, ohne dass Sie dafür zusätzliche Schritte unternehmen müssten.

2. Verwenden Sie das schrittweise Upgrade von DB2 High Availability Disaster Recovery (HADR), um eine Beeinträchtigung anderer Verwaltungsaktivitäten zu minimieren.

   Wenn Sie für Software oder Hardware ein Upgrade durchführen oder einige Konfigurationsparameter des Datenbankmanagers modifizieren möchten, können Sie diese Änderungen mit der HADR-Funktion bei nur minimaler Unterbrechung der Verfügbarkeit ausführen. Diese durch HADR ermöglichte reibungslose Änderung wird als schrittweises Upgrade bezeichnet.

   Bei einigen Verwaltungsaktivitäten ist es jedoch selbst in der HADR-Umgebung notwendig, dass Sie eine Datenbank beenden, bevor Sie die Verwaltungsarbeit durchführen können. Unter bestimmten Umständen unterscheidet sich die Prozedur zum Beenden einer HADR-Datenbank geringfügig von der Prozedur zum Beenden einer Standarddatenbank: Wenn eine HADR-Datenbank von einer Clientanwendung gestartet wird, die eine Verbindung zu ihr herstellt, müssen Sie den Befehl **DEACTIVATE DATABASE** verwenden.

Stoppen von DB2 High Availability Disaster Recovery (HADR)

Wenn Sie die DB2 High Availability Disaster Recovery-Funktion (HADR-Funktion) verwenden, kann es erforderlich sein, HADR-Operationen zur Durchführung von Pflegemaßnahmen in der Primärdatenbank und den Bereitschaftsdatenbanken zu stoppen. Die HADR-Operationen dürfen nur in der Datenbank gestoppt werden, auf der die Pflegemaßnahmen durchgeführt wird. Wenn Sie die Verwendung von HADR vollständig einstellen möchten, dann stoppen Sie HADR auf beiden Datenbanken.

Informationen zu diesem Vorgang

Warnung: Wenn Sie die angegebene Datenbank stoppen möchten, diese jedoch ihre Rolle als Primär- oder Bereitschaftsdatenbank in einer HADR-Konfiguration behalten soll, verwenden Sie den Befehl STOP HADR nicht. Wenn Sie den Befehl **STOP HADR** ausführen, wird die Datenbank zu einer Standarddatenbank und erfordert möglicherweise eine Reinitialisierung, wenn sie den Betrieb als HADR-Datenbank wieder aufnehmen soll. Führen Sie stattdessen den Befehl **DEACTIVATE DATABASE** aus.

Wenn Sie den Befehl **STOP HADR** für eine Standarddatenbank absetzen, wird ein Fehler zurückgegeben.

Vorgehensweise

Gehen Sie wie folgt vor, um HADR-Operationen in der Primärdatenbank oder einer Bereitschaftsdatenbank zu stoppen:

- Setzen Sie über den Befehlszeilenprozessor den Befehl **STOP HADR** in der Datenbank ab, in der die Ausführung von HADR-Operationen gestoppt werden soll.

  Im folgenden Beispiel werden HADR-Operationen in der Datenbank SOCKS gestoppt.

  ```
  STOP HADR ON DATABASE SOCKS
  ```

  Wenn Sie diesen Befehl für eine inaktive Primärdatenbank absetzen, wird die Datenbank zu einer Standarddatenbank und bleibt offline.

  Wenn Sie diesen Befehl für eine inaktive Bereitschaftsdatenbank absetzen, wird die Datenbank zu einer Standarddatenbank, erhält den Status *Aktualisierende Recovery anstehend* und bleibt offline.

  Wenn Sie diesen Befehl für eine aktive Primärdatenbank absetzen, werden keine Protokolle mehr an die Bereitschaftsdatenbank übertragen, und in der Bereitschaftsdatenbank werden alle HADR-EDUs (EDU - Engine Dispatchable Unit) heruntergefahren. Die Datenbank wird zu einer Standarddatenbank und bleibt offline. Die Transaktionsverarbeitung kann fortgesetzt werden. Sie können den Befehl START HADR mit der Option AS PRIMARY verwenden, um der Datenbank wieder die Rolle einer Primärdatenbank zuzuweisen.

  Wenn Sie diesen Befehl für eine aktive Bereitschaftsdatenbank ausführen, wird eine Fehlernachricht zurückgegeben, die angibt, dass Sie die Bereitschaftsdatenbank inaktivieren müssen, bevor Sie sie in eine Standarddatenbank konvertieren können.
- Rufen Sie über eine Anwendung die Anwendungsprogrammierschnittstelle (API) **db2HADRStop** auf.
- Öffnen Sie über IBM Data Studio den Taskassistenten für den Befehl **STOP HADR**.

Aktivierung und Inaktivierung von Datenbanken in einer HA-DR-Umgebung

Wenn eine Standarddatenbank durch eine Clientverbindung gestartet wird, wird die Datenbank gestoppt, wenn der letzte Client die Verbindung trennt. Wenn eine HADR-Primärdatenbank durch eine Clientverbindung gestartet wird, ist dieser Vorgang äquivalent zum Starten der Datenbank über den Befehl **ACTIVATE DATABASE**.

Zum Stoppen einer HADR-Primärdatenbank, die durch eine Clientverbindung gestartet wurde, müssen Sie explizit den Befehl **DEACTIVATE DATABASE** ausführen.

In einer Standarddatenbank im Status 'aktualisierende Recovery anstehend' sind die Befehle **ACTIVATE DATABASE** und **DEACTIVATE DATABASE** nicht anwendbar. Sie können nur die aktualisierende Recovery (ROLLFORWARD) fortsetzen, die aktualisierende Recovery stoppen oder die Datenbank mit dem Befehl **START HADR** als HADR-Bereitschaftsdatenbank starten. Wenn eine Datenbank als HADR-Bereitschaftsdatenbank gestartet ist, können Sie die Befehle **ACTIVATE DATABASE** und **DEACTIVATE DATABASE** zum Starten und Stoppen der Datenbank verwenden.

Zum Aktivieren einer Primärdatenbank stehen die folgenden Methoden zur Verfügung:

- Herstellen einer Clientverbindung
- Befehl **ACTIVATE DATABASE**
- Taskassistent für Befehl **ACTIVATE DATABASE** in IBM Data Studio
- Befehl **START HADR** mit der Option `AS PRIMARY`

Zum Inaktivieren einer Primärdatenbank stehen die folgenden Methoden zur Verfügung:

- Befehl **DEACTIVATE DATABASE**

  Anmerkung: Wenn Sie eine HADR-Primärdatenbank im Status 'Unterbrochener Peer' mit dem Befehl **DEACTIVATE DATABASE** oder der API 'sqle_deactivate_db' inaktivieren, befindet sich die Datenbank in einem inkonsistenten Status. Beim Neustart der Datenbank muss eine Recovery nach einem Systemabsturz durchgeführt werden und vor dem Neustart können keine Offline-Backups dieser Datenbank erstellt werden.
- Taskassistent für Befehl **DEACTIVATE DATABASE** in IBM Data Studio
- Befehl **db2stop** mit dem Parameter **FORCE**

Zum Aktivieren einer Bereitschaftsdatenbank stehen die folgenden Methoden zur Verfügung:

- Befehl **ACTIVATE DATABASE**
- Taskassistent für Befehl **ACTIVATE DATABASE** in IBM Data Studio
- Befehl **START HADR** mit der Option `AS STANDBY`

Zum Inaktivieren einer Bereitschaftsdatenbank stehen die folgenden Methoden zur Verfügung:

- Befehl **DEACTIVATE DATABASE**
- Taskassistent für Befehl **DEACTIVATE DATABASE** in IBM Data Studio
- Befehl **db2stop** mit dem Parameter **FORCE**

Empfohlene Reihenfolge zum Herunterfahren eines HADR-Paars

n

Warnung: Obwohl der Befehl **STOP HADR** zum Stoppen von HADR auf einer Primär- und/oder Bereitschaftsdatenbank verwendet werden kann, sollten Sie den Befehl mit Vorsicht einsetzen. Wenn Sie die angegebene Datenbank stoppen möchten, diese jedoch ihre Rolle als HADR-Primärdatenbank oder -Bereitschaftsdatenbank beibehalten soll, verwenden Sie den Befehl **STOP HADR** nicht. Wenn Sie den Befehl **STOP HADR** absetzen, wird die Datenbank zu einer Standarddatenbank und erfordert möglicherweise eine Reinitialisierung, wenn sie den Betrieb als HADR-Datenbank wieder aufnehmen soll. Führen Sie stattdessen den Befehl **DEACTIVATE DATABASE** aus.

Wenn Sie lediglich den HADR-Betrieb beenden wollen, dann sollten Sie folgendermaßen vorgehen, um das HADR-Paar zu beenden:

1. Inaktivieren Sie die Primärdatenbank.
2. Stoppen Sie DB2 in der Primärdatenbank.
3. Inaktivieren Sie die Bereitschaftsdatenbank.
4. Stoppen Sie DB2 in der Bereitschaftsdatenbank.

Aspekte des Neuausgleichs für Tabellenbereiche in einer DB2-HADR-Umgebung

Mit den Anweisungen ALTER TABLESPACE REBALANCE oder ALTER TABLESPACE USING STOGROUP können Sie eine Neuausgleichsoperation für eine Primärdatenbank starten. Die Anweisung wird für die Bereitschaftsdatenbank wiederholt und eine entsprechende Neuausgleichsoperation wird gestartet.

Während der Neuausgleichsoperation können Sie die Anweisung ALTER TABLESPACE mit der Klausel REBALANCE SUSPEND angeben, um die Neuausgleichsoperation für die Primärdatenbank auszusetzen. Zum Fortsetzen der ausgesetzten Neuausgleichsoperation geben Sie die Anweisung ALTER TABLESPACE mit der Klausel REBALANCE RESUME an.

Die Bereitschaftsdatenbank bleibt im Status 'aktiv', wenn eine Anweisung ALTER TABLESPACE REBALANCE SUSPEND wiederholt wird. Da der Neuausgleich für die Primärdatenbank ausgesetzt wird, wenn die Bereitschaftsdatenbank die Rolle als neue Primärdatenbank übernimmt, wird die Neuausgleichsoperation für die neue Primärdatenbank ausgesetzt und die Neuausgleichsoperation für die neue Bereitschaftsdatenbank wird implizit fortgesetzt.

Wenn Sie einen Restore einer Datenbank mithilfe einer geteilten Spiegeldatenbank als Klondatenbank oder als Bereitschaftsdatenbank durchführen, werden alle ausgesetzten Neuausgleichsoperationen für Tabellenbereiche automatisch beim Start der Datenbank fortgesetzt.

Ausführen von schrittweisen Aktualisierungen und Upgrades in einer DB2-HADR-Umgebung

Verwenden Sie diese Prozedur in einer HADR-Umgebung (High Availability Disaster Recovery), wenn Sie ein Upgrade für Software oder Hardware durchführen, Ihr DB2-Datenbanksystem aktualisieren oder Änderungen an Datenbankkonfigurationsparametern vornehmen.

Durch diese Prozedur bleibt der Datenbankservice während des Upgradeprozesses verfügbar, wobei nur beim Wechsel von einer Datenbank zur nächsten eine kurze Serviceunterbrechung auftritt. Mit mehreren Bereitschaftsdatenbanken kann über den gesamten Aktualisierungs- oder Upgradeprozess hinweg kontinuierlicher HA- und DR-Schutz gewährleistet werden.

Vorbereitende Schritte

Überprüfen Sie die Systemvoraussetzungen für HADR. Siehe hierzu „Systemvoraussetzungen für High Availability Disaster Recovery (HADR)" auf Seite 73.

Das HADR-Paar sollte sich im Peerstatus befinden, bevor der schrittweise Upgrade gestartet wird.

Anmerkung: Alle Fixpacks und Upgrades für das DB2-Datenbanksystem sollten zuerst in einer Testumgebung implementiert werden, bevor Sie sie auf Ihr Produktionssystem anwenden.

Informationen zu diesem Vorgang

Mit dieser Prozedur können Sie kein Upgrade von einer früheren auf eine spätere Version eines DB2-Datenbanksystems durchführen; Sie können diese Prozedur beispielsweise nicht für das Upgrade eines Datenbanksystems von Version 8 auf Version 9 verwenden. Mit dieser Prozedur können Sie für Ihr Datenbanksystem lediglich eine schrittweise Aktualisierung von einer Modifikationsstufe auf eine andere durchführen, z. B. indem Sie ein Fixpack anwenden. Bei schrittweisen Updates kann die Modifikationsstufe (z. B. die Fixpackstufe) der Bereitschaftsdatenbank kurzzeitig zum Testen der neuen Stufe neuer als die der Primärdatenbank sein. Allerdings sollte diese Konfiguration nicht über einen längeren Zeitraum beibehalten werden, um das Risiko zu reduzieren, dass Funktionen verwendet werden, die bei den Stufen nicht kompatibel sind. Die Primär- und die Bereitschaftsdatenbank stellen keine Verbindung zueinander her, wenn die Modifikationsstufe des Datenbanksystems für eine Primärdatenbank neuer als die für die Bereitschaftsdatenbank ist.

Diese Prozedur funktioniert nicht, wenn Sie die DB2-HADR-Konfigurationsparameter aktualisieren. Aktualisierungen an HADR-Konfigurationsparametern sollten getrennt vorgenommen werden. Da HADR voraussetzt, dass die Parameter in der Primär- und der Bereitschaftsdatenbank identisch sind, kann dies erfordern, dass die Primär- und die Bereitschaftsdatenbank zur gleichen Zeit inaktiviert und aktualisiert werden.

Vorgehensweise

Gehen Sie wie folgt vor, um einen schrittweisen Upgrade in einer HADR-Umgebung auszuführen:

1. Führen Sie einen Upgrade für das System aus, auf dem sich die Bereitschaftsdatenbank befindet:
   a. Inaktivieren Sie die Bereitschaftsdatenbank mit dem Befehl **DEACTIVATE DATABASE**.
   b. Falls erforderlich, fahren Sie die Instanz in der Bereitschaftsdatenbank herunter.
   c. Nehmen Sie Änderungen an der Software, Hardware und/oder den DB2-Konfigurationsparametern vor.

Anmerkung: Sie können keine HADR-Konfigurationsparameter ändern, wenn Sie einen schrittweisen Upgrade ausführen.

d. Falls erforderlich, starten Sie die Instanz in der Bereitschaftsdatenbank erneut.
e. Starten Sie die Bereitschaftsdatenbank mit dem Befehl **db2pd** erneut.
f. Stellen Sie sicher, dass die Bereitschaftsdatenbank den Peerstatus erhält. Überprüfen Sie dies mit dem Befehl **GET SNAPSHOT**.

2. Tauschen Sie die Rollen der Primärdatenbank und die Bereitschaftsdatenbank:
   a. Geben Sie auf der Bereitschaftsdatenbank den Befehl **TAKEOVER HADR** ein.
   b. Leiten Sie Clients an die neue Primärdatenbank um. Sie können dazu die automatische Clientweiterleitung verwenden.

      Anmerkung: Da die Bereitschaftsdatenbank die Rolle der Primärdatenbank übernimmt, wird nun ein Upgrade für die neue Primärdatenbank ausgeführt. Wenn Sie ein Fixpack für ein DB2-Datenbanksystem anwenden, ändert der Befehl **TAKEOVER HADR** die Rolle der ursprünglichen Primärdatenbank in die der Bereitschaftsdatenbank. Der Befehl erstellt jedoch keine Verbindung zwischen der neuen Bereitschaftsdatenbank und der neu aktualisierten Primärdatenbank. Da die neue Bereitschaftsdatenbank eine ältere Version des DB2-Datenbanksystems verwendet, kann sie möglicherweise die neuen, von der aktualisierten Primärdatenbank generierten Protokollsätze nicht verstehen und wird heruntergefahren. Damit die neue Bereitschaftsdatenbank wieder eine Verbindung zur neuen Primärdatenbank herstellen kann (d. h., damit das HADR-Paar erneut gebildet werden kann), muss auch die neue Bereitschaftsdatenbank aktualisiert werden.
3. Führen Sie einen Upgrade für die ursprüngliche Primärdatenbank (die aktuelle Bereitschaftsdatenbank) aus, und verwenden Sie dazu die oben in Schritt 1 beschriebene Prozedur. Wenn Sie dies ausgeführt haben, sind beide Datenbanken aktualisiert und stellen im HADR-Peerstatus eine Verbindung zueinander her. Das HADR-System stellt den vollständigen Datenbankservice und einen vollständigen Hochverfügbarkeitsschutz zur Verfügung.
4. Optional: Wenn Sie zu Ihrer ursprünglichen Konfiguration zurückkehren wollen, tauschen Sie die Rollen der Primärdatenbank und die Bereitschaftsdatenbank, wie in Schritt 2 beschrieben.

   Wenn Sie die HADR-Funktion für Leseoperationen in der Bereitschaftsdatenbank während des schrittweisen Upgrades aktivieren möchten, verschieben Sie den optionalen Schritt 4 auf einen späteren Zeitpunkt und führen Sie die folgenden Schritte aus. Das Binden von internen DB2-Paketen findet zum Zeitpunkt der ersten Verbindungsherstellung statt und kann nur auf der Primärdatenbank erfolgreich ausgeführt werden. Diese Schritte sind erforderlich, um die Konsistenz der internen DB2-Pakete in der Bereitschaftsdatenbank sicherzustellen, bevor Leseoperationen vorgenommen werden.
5. Aktivieren Sie die HADR-Funktion für Leseoperationen in der Bereitschaftsdatenbank wie folgt für die Bereitschaftsdatenbank:
   a. Setzen Sie die Registrierdatenbankvariable **DB2_HADR_ROS** in der Bereitschaftsdatenbank auf ON.
   b. Inaktivieren Sie die Bereitschaftsdatenbank mit dem Befehl **DEACTIVATE DATABASE**.
   c. Starten Sie die Instanz in der Bereitschaftsdatenbank neu.
   d. Starten Sie die Bereitschaftsdatenbank mit dem Befehl **ACTIVATE DATABASE** erneut.

e. Überprüfen Sie mit dem Befehl **GET SNAPSHOT**, ob die Bereitschaftsdatenbank in den Peerstatus (PEER) versetzt wird.

6. Tauschen Sie die Rollen der Primärdatenbank und der Bereitschaftsdatenbank wie folgt:
   a. Geben Sie auf der Bereitschaftsdatenbank den Befehl **TAKEOVER HADR** ein.
   b. Leiten Sie Clients an die neue Primärdatenbank um.
7. Wiederholen Sie die Prozedur in Schritt 5, um die HADR-Funktion für Leseoperationen in der Bereitschaftsdatenbank für die neue Bereitschaftsdatenbank zu aktivieren.
8. Optional: Wenn Sie zu Ihrer ursprünglichen Konfiguration zurückkehren wollen, tauschen Sie die Rollen der Primärdatenbank und die Bereitschaftsdatenbank, wie in Schritt 2 beschrieben.

Schrittweises Upgrade in einer automatisierten HADR-Umgebung (HADR = High Availability Disaster Recovery)

Wenn Sie das integrierte High Availability Feature (HA) zur Automatisierung von HADR verwenden, sind zusätzliche Schritte für das Upgrade der Software (Betriebssystem oder DB2-Datenbanksystem), für das Upgrade der Hardware oder für Änderungen der Datenbankkonfigurationsparameter erforderlich. Gehen Sie wie folgt vor, um ein schrittweises Upgrade in einer automatisierten HADR-Umgebung durchzuführen.

Vorbereitende Schritte

Die folgenden Voraussetzungen müssen erfüllt sein, damit die unter 'Vorgehensweise' beschriebenen Schritte ausgeführt werden können:

- Es müssen zwei DB2-Instanzen vorhanden sein (in diesem Fall haben sie den Namen 'stevera' auf jedem Knoten).
- Es müssen zwei Knoten vorhanden sein ('grom04' und 'grom03'). Auf der 'grom04'-Maschine befindet sich zu Beginn die HADR-Primärdatenbank.
- Die Instanzen müssen ursprünglich mit dem Code von DB2 Version 9.8 GA ausgeführt werden.
- Die Instanzen müssen mit der integrierten HA-Steuerung der HADR-Funktionsübernahme konfiguriert sein. Die Clusterdomäne heißt 'test'.

Anmerkung: Alle Fixpacks und Upgrades für das DB2-Datenbanksystem sollten zuerst in einer Testumgebung implementiert werden, bevor Sie sie auf Ihr Produktionssystem anwenden.

Das HADR-Paar sollte sich im Peerstatus befinden, bevor der schrittweise Upgrade gestartet wird.

Einschränkungen

Mit dieser Prozedur können Sie keine Migration von einer früheren auf eine spätere Version eines DB2-Datenbanksystems durchführen; Sie können diese Prozedur beispielsweise nicht zum Migrieren eines Datenbanksystems von Version 8 auf Version 9 verwenden. Mit dieser Prozedur können Sie Ihr Datenbanksystem nur von einer Modifikationsstufe auf eine andere aktualisieren, z. B. indem Sie ein Fixpack anwenden.

Diese Prozedur funktioniert nicht, wenn Sie die DB2-HADR-Konfigurationsparameter aktualisieren. Aktualisierungen an HADR-Konfigurationsparametern sollten getrennt vorgenommen werden. Da HADR voraussetzt, dass die Parameter in der

Primär- und der Bereitschaftsdatenbank identisch sind, kann dies erfordern, dass die Primär- und die Bereitschaftsdatenbank zur gleichen Zeit inaktiviert und aktualisiert werden.

Vorgehensweise

1. Anfangsstatus des Systems anzeigen:

```
root@grom03:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  ------------------  -------------   -------  -------
test   Online    2.4.7.1             No              12347    12348

root@grom03:# lssam
Online IBM.ResourceGroup:db2_stevera_grom03_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom03_0-rs
                '- Online IBM.Application:db2_stevera_grom03_0-rs:grom03
Online IBM.ResourceGroup:db2_stevera_grom04_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom04_0-rs
                '- Online IBM.Application:db2_stevera_grom04_0-rs:grom04
Online IBM.ResourceGroup:db2_stevera_stevera_SVTDB-rg Nominal=Online
        |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs
                |- Offline IBM.Application:db2_stevera_stevera_SVTDB-rs:grom03
                '- Online IBM.Application:db2_stevera_stevera_SVTDB-rs:grom04
        '- Online IBM.ServiceIP:db2ip_9_26_124_22-rs
                |- Offline IBM.ServiceIP:db2ip_9_26_124_22-rs:grom03
                '- Online IBM.ServiceIP:db2ip_9_26_124_22-rs:grom04

root@grom03:# lsrpnode
Name     OpState   RSCTVersion
-----    -------   ------------
grom03   Online    2.4.7.1
grom04   Online    2.4.7.1
```

   Dieses Beispiel zeigt, dass Sie ein Upgrade für die Bereitschaftsinstanz auf 'grom03' durchführen müssen. Stoppen Sie hierzu alle Ressourcengruppen, die sich auf 'grom03' befinden.

2. Stoppen aller Ressourcengruppen auf dem Bereitschaftsknoten und Bestätigen der Änderung:

```
root@grom03:# chrg -o Offline db2_stevera_grom03_0-rg

root@grom03:# lssam g db2_stevera_grom03_0-rg
Offline IBM.ResourceGroup:db2_stevera_grom03_0-rg Nominal=Offline
        '- Offline IBM.Application:db2_stevera_grom03_0-rs
                '- Offline IBM.Application:db2_stevera_grom03_0-rs:grom03
```

3. Stoppen des Clusterknotens (Bereitschaftsknotens) und Bestätigen der Änderung:

```
root@grom03:# stoprpnode grom03
root@grom03:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
----   -------   -----------------   -------------   ------   ------
test   Offline   2.4.7.1             No              12347    12348
```

4. Installieren des DB2-Fixpacks auf dem Bereitschaftsknoten:

   Optional: Unter AIX ist möglicherweise die Installation der RSCT-Voraussetzungen für das betreffende Fixpack erforderlich.

```
root@grom03:# ./installFixPack -b /opt/ibm/db2/V9.8
DBI1017I  installFixPack aktualisiert das bzw. die an der Speicherposition
'/opt/ibm/db2/V9.8' installierte(n) Produkt(e).
```

   Die Installation des DB2-Fixpacks wird gestartet.

5. Starten des Knotens und Versetzen der Ressourcengruppe in den Onlinemodus:

   Wenn die Installation erfolgreich abgeschlossen ist.

```
root@grom03:# startrpdomain test

root@grom03:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  ------------------  --------------  -------  -------
test   Online    2.4.7.1             Yes             12347    12348

root@grom03:# chrg -o Online db2_stevera_grom03_0-rg
```

6. Prüfen, ob das Fixpack angewendet wurde und sich HADR wieder im Peerzustand befindet:

```
stevera@grom03% db2level

stevera@grom03% db2pd hadr db SVTDB
```

7. TAKEOVER-Operation durchführen:

   Für ein Upgrade des anderen Knotens (in diesem Fall 'grom04') führen Sie die TAKEOVER-Operation so aus, dass sich die HADR-Primärdatenbank auf dem Knoten 'grom03' befindet.

```
root@grom03:# su - stevera

stevera@grom03% db2 takeover hadr on db SVTDB
DB20000I  Der Befehl TAKEOVER HADR ON DATABASE wurde erfolgreich ausgeführt.

root@grom03:# lssam
Online IBM.ResourceGroup:db2_stevera_grom03_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom03_0-rs
                '- Online IBM.Application:db2_stevera_grom03_0-rs:grom03
Online IBM.ResourceGroup:db2_stevera_grom04_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom04_0-rs
                '- Online IBM.Application:db2_stevera_grom04_0-rs:grom04
Online IBM.ResourceGroup:db2_stevera_stevera_SVTDB-rg Nominal=Online
        |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs
                |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs:grom03
                '- Offline IBM.Application:db2_stevera_stevera_SVTDB-rs:grom04
        '- Online IBM.ServiceIP:db2ip_9_26_124_22-rs
                |- Online IBM.ServiceIP:db2ip_9_26_124_22-rs:grom03
                '- Offline IBM.ServiceIP:db2ip_9_26_124_22-rs:grom04
```

8. Durchführen eines Upgrades auf dem Knoten 'grom04':

```
root@grom03:# ssh root@grom04

root@grom04:# chrg -o Offline db2_stevera_grom04_0-rg

root@grom04:# lssam g db2_stevera_grom04_0-rg
Offline IBM.ResourceGroup:db2_stevera_grom04_0-rg Nominal=Offline
        '- Offline IBM.Application:db2_stevera_grom04_0-rs
                '- Offline IBM.Application:db2_stevera_grom04_0-rs:grom04

root@grom04:# stoprpnode grom04
```

   Optional: Unter AIX ist möglicherweise die Installation der RSCT-Voraussetzungen für das betreffende Fixpack erforderlich.

```
root@grom04:# ./installFixPack -b /opt/ibm/db2/V9.8
DBI1017I  installFixPack aktualisiert das bzw. die an der Speicherposition
'/opt/ibm/db2/V9.8' installierte(n) DB2-Produkt(e).
```

   Die Installation des DB2-Fixpacks wird gestartet.

   Wenn die Installation erfolgreich abgeschlossen ist.

```
root@grom04:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  ------------------  --------------  -------  -------
test   Offline   2.4.7.1             Yes             12347    12348

root@grom04:# startrpdomain test
```

```
root@grom04:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  ------------------  --------------  -------  -------
test   Online    2.4.7.1             Yes             12347    12348

root@grom04:# chrg -o Online db2_stevera_grom04_0-rg
```

9. Prüfen, ob das Fixpack angewendet wurde (durch Ausführen von **db2level**) und sich HADR im Peerzustand befindet (durch Ausführen von db2pd hadr db svtdb):

```
root@grom04:# su - stevera

stevera@grom04% db2pd -hadr -db svtdb

Database Partition 0 -- Database SVTDB -- Active -- Up 0 days 00:00:05

HADR-Information:
Role     State    SyncMode    HeartBeatsMissed    LogGapRunAvg (bytes)
-----    ------   ---------   ----------------    --------------------
Standby  Peer     Sync        0                   0

ConnectStatus   ConnectTime                            Timeout
--------------  ------------                           --------
Connected       Tue May 5 13:20:58 2009 (1241544058)   120

PeerWindowEnd                          PeerWindow
--------------                         -----------
Tue May  5 13:25:58 2009 (1241544358)  300

LocalHost                              LocalService
----------                             -------------
grom04                                 55555

RemoteHost                             RemoteService      RemoteInstance
----------                             --------------     --------------
grom03                                 55555              stevera

PrimaryFile    PrimaryPg    PrimaryLSN
------------   ----------   -----------
S0000001.LOG   1            0x0000000003389487

StandByFile    StandByPg    StandByLSN          StandByRcvBufUsed
------------   ----------   -----------         ------------------
S0000001.LOG   1            0x0000000003389487  0%

root@grom04:# lssam
Online IBM.ResourceGroup:db2_stevera_grom03_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom03_0-rs
                '- Online IBM.Application:db2_stevera_grom03_0-rs:grom03
Online IBM.ResourceGroup:db2_stevera_grom04_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom04_0-rs
                '- Online IBM.Application:db2_stevera_grom04_0-rs:grom04
Online IBM.ResourceGroup:db2_stevera_stevera_SVTDB-rg Nominal=Online
        |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs
                |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs:grom03
                '- Offline IBM.Application:db2_stevera_stevera_SVTDB-rs:grom04
        '- Online IBM.ServiceIP:db2ip_9_26_124_22-rs
                |- Online IBM.ServiceIP:db2ip_9_26_124_22-rs:grom03
                '- Offline IBM.ServiceIP:db2ip_9_26_124_22-rs:grom04
```

10. Migration der TSA-Domäne:

```
root@grom04:# export CT_MANAGEMENT_SCOPE=2

root@grom04:# runact -c IBM.PeerDomain CompleteMigration Options=0
Resource Class Action Response for CompleteMigration

root@grom04:# samctrl -m
```

```
Ready to Migrate! Are you Sure? [Y|N]:.

Y
```

11. Sicherstellen, dass der Wert für **MixedVersions** nicht mehr Yes für die Clusterkomponente lautet:

```
root@grom04:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  ------------------  --------------  -------  -------
test   Online    2.5.1.2             No              12347    12348
```

12. Sicherstellen, dass die AVN (Active Version Number) mit der IVN (Installed Version Number) für den HA-Manager übereinstimmt:

```
root@grom04:# lssrc ls IBM.RecoveryRM |grep VN
   Our IVN         : 2.2.0.7
   Our AVN         : 2.2.0.7
```

13. Optional: Durchführen einer TAKEOVER-Operation als Instanzeigner 'stevera' auf der 'grom04'-Maschine, um 'grom04' zur HADR-Primärdatenbank zu machen (wie ursprünglich).

Verwenden einer geteilten Spiegeldatenbank zum Klonen einer Datenbank

Verwenden Sie die folgende Prozedur, um eine Klondatenbank in einer Umgebung außerhalb einer DB2 pureScale-Umgebung zu erstellen. Klondaten werden allgemein für Aktivitäten verwendet, bei denen nur Lesezugriff erforderlich ist (z. B. das Ausführen von Berichten), es ist jedoch auch möglich, in Klondatenbanken zu schreiben.

Informationen zu diesem Vorgang

Wenn die primäre Datenbank für die Protokollarchivierung konfiguriert wurde, wird dieselbe Protokollarchivierungskonfiguration von der Klondatenbank gemeinsam genutzt. Wenn die Position der Archivprotokolle für die Klondatenbank zugänglich ist, könnte dies dazu führen, dass die Klondatenbank Protokolldateien an derselben Position wie die primäre Datenbank archiviert. Dies kann Auswirkungen auf die Wiederherstellbarkeit beider Datenbanken haben. Während die Klondatenbank zunächst eine andere Protokollkette als die Primärdatenbank verwendet, könnte die Primärdatenbank mit der Zeit denselben Wert für die Protokollkette verwenden wie die Klondatenbank. Bevor Sie den Befehl **db2inidb** ausführen, sollten Sie daher das Ziel der Protokollarchivierung für die Klondatenbank auf ein anderes Ziel ändern, damit Probleme mit der Wiederherstellbarkeit vermieden werden.

Sie können eine Klondatenbank weder sichern, noch ihr Image auf dem Originalsystem wiederherstellen oder sie mithilfe der Protokolldateien, die auf dem Originalsystem generiert wurden, aktualisierend wiederherstellen. Die Klondatenbank stellt nur eine unmittelbare Kopie der Datenbank bereit, wenn die Ein-/Ausgabe ausgesetzt wird. Sämtliche weitere ausstehende und nicht festgeschriebene Arbeit wird rückgängig gemacht, nachdem der Befehl **db2inidb** ausgeführt wurde.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Datenbank zu klonen:

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zur Primärdatenbank her:

```
db2 connect to datenbankname
```

2. Setzen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl aus:

   ```
   db2 set write suspend for database
   ```

   Anmerkung: Führen Sie keine weiteren Dienstprogramme oder Tools aus, während sich die Datenbank im ausgesetzten Status befindet. Das Erstellen einer Datenbankkopie sollte die einzige Aktivität sein. Optional können Sie vor Eingabe des Befehls **SET WRITE SUSPEND** alle Pufferpools löschen, um das Recoveryfenster zu minimieren. Hierfür können Sie die Anweisung FLUSH BUFFERPOOLS ALL verwenden.
3. Erstellen Sie aus der Primärdatenbank mithilfe der entsprechenden Befehle auf Betriebssystem- bzw. Speicherebene mindestens eine geteilte Spiegeldatenbank.

   Anmerkung:

   - Stellen Sie sicher, dass Sie das gesamte Datenbankverzeichnis einschließlich des Datenträgerverzeichnisses kopieren. Sie müssen auch das Protokollverzeichnis und alle Containerverzeichnisse kopieren, die außerhalb des Datenbankverzeichnisses vorhanden sind. Eine Zusammenstellung dieser Informationen finden Sie in der Verwaltungssicht DBPATHS, in der alle Dateien und Verzeichnisse der Datenbank angezeigt werden, die geteilt werden müssen.
   - Wenn Sie `EXCLUDE LOGS` im Befehl **SET WRITE** angegeben haben, dürfen Sie die Protokolldateien nicht in die Kopie einbeziehen.
4. Nehmen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl wieder auf:

   ```
   db2 set write resume for database
   ```
5. Katalogisieren Sie die gespiegelte Datenbank auf dem sekundären System.

   Anmerkung: Eine gespiegelte Datenbank kann standardmäßig nicht in demselben System vorhanden sein wie die Primärdatenbank. Sie muss sich auf einem sekundären System mit der gleichen Verzeichnisstruktur befinden und denselben Instanznamen als primäre Datenbank verwenden. Wenn die gespiegelte Datenbank auf demselben System wie die Primärdatenbank vorhanden sein muss, können Sie das Dienstprogramm **db2relocatedb** oder die Option **RELOCATE USING** des Befehls **db2inidb** für diese Aufgabe verwenden.
6. Starten Sie die Datenbankinstanz auf dem sekundären System mit folgendem Befehl:

   ```
   db2start
   ```
7. Initialisieren Sie die gespiegelte Datenbank auf dem sekundären System:

   ```
   db2inidb aliasname-der-datenbank as snapshot
   ```

   Falls erforderlich, geben Sie die Option `RELOCATE USING` des Befehls **db2inidb** an, um die Klondatenbank zu verlagern:

   ```
   db2inidb aliasname-der-datenbank as snapshot relocate using relocatedbcfg.txt
   ```

   Dabei ist `relocatedbcfg.txt` die Datei, die die zum Verlagern der Datenbank erforderlichen Informationen enthält.

   Anmerkung:

   - Mit diesem Befehl werden Transaktionen rückgängig gemacht, die während des Teilens gerade verarbeitet werden, und es wird eine neue Protokollkettenreihenfolge begonnen, sodass Protokolle aus der primären Datenbank nicht für die Klondatenbank wiedergegeben werden können.
   - Wenn die Primärdatenbank für die Protokollarchivierung konfiguriert wurde, wird dieselbe Protokollarchivierungskonfiguration von der Klondatenbank

gemeinsam genutzt. Dies bedeutet, dass die Klondatenbank versucht, Protokolldateien an derselben Position wie die Primärdatenbank zu archivieren, wenn diese Position für die Klondatenbank zugänglich ist. Obwohl die Klondatenbank zunächst eine andere Protokollkette als die Primärdatenbank verwendet, könnte die Primärdatenbank mit der Zeit denselben Wert für die Protokollkette verwenden wie die Klondatenbank. Dies kann dazu führen, dass die Primärdatenbank Protokolldateien über die Protokolldateien archiviert, die von der Klondatenbank archiviert wurden, oder umgekehrt. Dies kann Auswirkungen auf die Wiederherstellbarkeit beider Datenbanken haben. Zum Vermeiden dieser Probleme sollten Sie das Ziel der Protokollarchivierung für die Klondatenbank ändern, so dass es sich von dem Ziel der Primärdatenbank unterscheidet.

Verwenden einer geteilten Spiegeldatenbank zum Klonen einer Datenbank in einer DB2 pureScale-Umgebung

Verwenden Sie die folgende Prozedur, um eine Klondatenbank in einer in a DB2 pureScale-Umgebung zu erstellen. Klondaten werden allgemein für Aktivitäten verwendet, bei denen nur Lesezugriff erforderlich ist (z. B. das Ausführen von Berichten), es ist jedoch auch möglich, in Klondatenbanken zu schreiben.

Informationen zu diesem Vorgang

Wenn die primäre Datenbank für die Protokollarchivierung konfiguriert wurde, wird dieselbe Protokollarchivierungskonfiguration von der Klondatenbank gemeinsam genutzt. Wenn die Position der Archivprotokolle für die Klondatenbank zugänglich ist, könnte dies dazu führen, dass die Klondatenbank Protokolldateien an derselben Position wie die primäre Datenbank archiviert. Dies kann Auswirkungen auf die Wiederherstellbarkeit beider Datenbanken haben. Während die Klondatenbank zunächst eine andere Protokollkette als die Primärdatenbank verwendet, könnte die Primärdatenbank mit der Zeit denselben Wert für die Protokollkette verwenden wie die Klondatenbank. Bevor Sie den Befehl **db2inidb** ausführen, sollten Sie daher das Ziel der Protokollarchivierung für die Klondatenbank auf ein anderes Ziel ändern, damit Probleme mit der Wiederherstellbarkeit vermieden werden.

Sie können eine Klondatenbank weder sichern, noch ihr Image auf dem Originalsystem wiederherstellen oder sie mithilfe der Protokolldateien, die auf dem Originalsystem generiert wurden, aktualisierend wiederherstellen. Die Klondatenbank stellt nur eine unmittelbare Kopie der Datenbank bereit, wenn die Ein-/Ausgabe ausgesetzt wird. Sämtliche weitere ausstehende und nicht festgeschriebene Arbeit wird rückgängig gemacht, nachdem der Befehl **db2inidb** ausgeführt wurde.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Datenbank zu klonen:

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zur Primärdatenbank her:

   ```
   db2 connect to <datenbankname>
   ```

2. Konfigurieren Sie das General Parallel File System (GPFS) auf dem sekundären Cluster durch Extrahieren und Importieren der Einstellungen des primären Clusters. Führen Sie auf dem primären Cluster den folgenden GPFS-Befehl aus:

   ```
   mmfsctl dateisystem syncFSconfig -n datei-des-fernen-knotens
   ```

Dabei steht *datei-des-fernen-knotens* für die Liste der Hosts im sekundären Cluster.

3. Listen Sie die Cluster-Manager-Domäne mit folgendem Befehl auf:

   ```
   db2cluster -cm -list -domain
   ```

4. Stoppen Sie den Cluster-Manager auf jedem Host im Cluster mit folgendem Befehl:

   ```
   db2cluster
   -cm -stop -host host -force
   ```

   Anmerkung: Sie müssen den Host, von dem aus Sie diesen Befehl absetzen, als Letztes beenden.

5. Stoppen Sie den GPFS-Cluster auf dem sekundären System mit dem folgenden Befehl:

   ```
   db2cluster -cfs -stop -all
   ```

6. Setzen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl aus:

   ```
   db2 set write suspend for database
   ```

   Anmerkung: Führen Sie keine weiteren Dienstprogramme oder Tools aus, während sich die Datenbank im ausgesetzten Status befindet. Das Erstellen einer Datenbankkopie sollte die einzige Aktivität sein. Optional können Sie vor Eingabe des Befehls **SET WRITE SUSPEND** alle Pufferpools löschen, um das Recoveryfenster zu minimieren. Hierfür können Sie die Anweisung FLUSH BUFFERPOOLS ALL verwenden.

7. Ermitteln Sie mithilfe des folgenden Befehls, welche Dateisysteme ausgesetzt und kopiert werden müssen:

   ```
   db2cluster -cfs -list -filesystem
   ```

8. Setzen Sie mit folgendem Befehl alle GPFS-Dateisysteme aus, die Daten oder Protokolldaten enthalten:

   ```
   /usr/lpp/mmfs/bin/mmfsctl dateisystem suspend-write
   ```

   Dabei steht *dateisystem* für ein Dateisystem, das Daten oder Protokolldaten enthält.

   Anmerkung: Wenn die GPFS-Dateisysteme ausgesetzt sind, sind nur Schreiboperationen blockiert.

9. Erstellen Sie aus der Primärdatenbank mithilfe der entsprechenden Befehle auf Betriebssystem- bzw. Speicherebene mindestens eine geteilte Spiegeldatenbank.

   Anmerkung:

   - Stellen Sie sicher, dass Sie das gesamte Datenbankverzeichnis einschließlich des Datenträgerverzeichnisses kopieren. Sie müssen auch das Protokollverzeichnis und alle Containerverzeichnisse kopieren, die außerhalb des Datenbankverzeichnisses vorhanden sind. Eine Zusammenstellung dieser Informationen finden Sie in der Verwaltungssicht DBPATHS, in der alle Dateien und Verzeichnisse der Datenbank angezeigt werden, die geteilt werden müssen.
   - Wenn Sie `EXCLUDE LOGS` im Befehl **SET WRITE** angegeben haben, dürfen Sie die Protokolldateien nicht in die Kopie einbeziehen.

10. Setzen Sie den Betrieb der ausgesetzten GPFS-Dateisysteme fort. Verwenden Sie dazu den folgenden Befehl für jedes ausgesetzte Dateisystem:

```
/usr/lpp/mmfs/bin/mmfsctl dateisystem resume
```

Dabei steht *dateisystem* für ein ausgesetztes Dateisystem, das Daten oder Protokolldaten enthält.

11. Nehmen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank wieder auf:

    ```
    db2 set write resume for database
    ```

12. Starten Sie den GPFS-Cluster auf dem sekundären System mit dem folgenden Befehl:

    ```
    db2cluster -cfs -start -all
    ```

13. Starten Sie den Cluster-Manager mit folgendem Befehl:

    ```
    db2cluster -cm -start -domain domäne
    ```

14. Katalogisieren Sie die Spiegeldatenbank auf dem sekundären System:

    Anmerkung: Eine gespiegelte Datenbank kann standardmäßig nicht in demselben System vorhanden sein wie die Primärdatenbank. Sie muss sich auf einem sekundären System mit der gleichen Verzeichnisstruktur befinden und denselben Instanznamen als primäre Datenbank verwenden. Wenn die gespiegelte Datenbank auf demselben System wie die Primärdatenbank vorhanden sein muss, können Sie das Dienstprogramm **db2relocatedb** oder die Option **RELOCATE USING** des Befehls **db2inidb** für diese Aufgabe verwenden.

15. Starten Sie die Datenbankinstanz auf dem sekundären System mit folgendem Befehl:

    ```
    db2start
    ```

16. Initialisieren Sie die gespiegelte Datenbank auf dem sekundären System mit folgendem Befehl:

    ```
    db2inidb aliasname-der-datenbank as snapshot
    ```

    Falls erforderlich, geben Sie die Option **RELOCATE USING** des Befehls **db2inidb** an, um die Klondatenbank zu verlagern:

    ```
    db2inidb aliasname-der-datenbank as snapshot relocate using relocatedbcfg.txt
    ```

    Dabei ist `relocatedbcfg.txt` die Datei, die die zum Verlagern der Datenbank erforderlichen Informationen enthält.

    Anmerkung:

    - Mit diesem Befehl werden Transaktionen rückgängig gemacht, die während des Teilens gerade verarbeitet werden, und es wird eine neue Protokollkettenreihenfolge begonnen, sodass Protokolle aus der primären Datenbank nicht für die Klondatenbank wiedergegeben werden können.
    - Wenn die Primärdatenbank für die Protokollarchivierung konfiguriert wurde, wird dieselbe Protokollarchivierungskonfiguration von der Klondatenbank gemeinsam genutzt. Wenn das Ziel der Protokollarchivierung von der Klondatenbank aus zugänglich ist, ruft die Bereitschaftsdatenbank während Operationen zur aktualisierenden Recovery aus diesem Ziel automatisch Protokolldateien ab. Sobald jedoch die Datenbank den Status 'Aktualisierende Recovery anstehend' verlassen hat, versucht die Klondatenbank, Protokolldateien an derselben Position zu archivieren, die auch die Primärdatenbank verwendet hat. Obwohl die Bereitschaftsdatenbank zunächst eine andere Protokollkette als die Primärdatenbank verwendet, könnte die Primärdatenbank mit der Zeit denselben Wert für die Protokollkette verwenden wie die Klondatenbank. Dies kann dazu führen, dass die Primärdatenbank Protokolldateien über die Protokolldateien archiviert, die von der Klondatenbank archiviert wurden, oder umgekehrt. Dies kann Auswirkungen auf die Wiederherstellbarkeit beider Datenbanken haben. Zum Vermei-

den dieser Probleme sollten Sie das Ziel der Protokollarchivierung für die Klondatenbank ändern, so dass es sich von dem Ziel der Primärdatenbank unterscheidet.

Szenario: Ändern der Systemuhr

Wenn Sie die Systemuhr anpassen oder ändern wollen, müssen Sie den DB2-Datenbankmanager nicht stoppen. DB2 for Linux, UNIX and Windows kann den Wechsel zwischen Sommer- und Winterzeit zweimal pro Jahr weltweit erfolgreich ändern, ohne dass dabei Probleme auftreten.

Konfigurationen, die NTP verwenden, um Uhren systemübergreifend zu synchronisieren, werden ebenfalls vollständig unterstützt.

Informationen zu diesem Vorgang

Bei der Änderung der Systemzeit müssen verschiedene bewährte Verfahren beachtet werden.

Einschränkungen

Die Änderung der Systemuhr hat in der überwiegenden Anzahl der Szenarios absolut keine Auswirkungen.

Bei Auftreten erheblicher Zeitverschiebungen müssen Sie jedoch zwei Faktoren berücksichtigen.

- Bei der Durchführung der punktuellen Recovery müssen erhebliche Zeitverschiebungen berücksichtigt werden.
- Funktionsdefinitionen umfassen die Uhrzeit und das Datum der Erstellung in Form einer Zeitmarke. Beim Funktionsaufruf versucht DB2 for Linux, UNIX and Windows, die Funktionsdefinition aufzulösen. Im Rahmen der Funktionsauflösung wird der Zeitmarkenwert, der in der Funktionsdefinition zum Erstellungszeitpunkt protokolliert wurde, überprüft. Wenn Sie die Systemuhr auf eine Uhrzeit zurückstellen, die vor dem Zeitpunkt der Funktionserstellung liegt, dann löst DB2 for Linux, UNIX and Windows Referenzen auf diese Funktionen nicht auf.

Vorgehensweise

Zur Vermeidung dieser beiden Situationen stehen die folgenden bewährten Verfahren zur Verfügung:

1. Wenn Sie die Uhrzeit vorstellen, dann fahren Sie mit Schritt 3 fort.
2. Wenn Sie die Uhrzeit um *X* Minuten zurückstellen, dann gehen Sie wie folgt vor:
   a. Wählen Sie den Zeitpunkt zur Durchführung der Änderung so aus, dass während der letzten *X* Minuten keine neuen Funktionen erstellt wurden und innerhalb der nächsten *X* Minuten keine Aktualisierungstransaktionen durchgeführt werden.
   b. Wenn es nicht möglich ist, einen geeigneten Zeitpunkt gemäß den Anforderungen in Schritt a zu finden, dann kann die Systemuhr trotzdem um *X* Minuten zurückgestellt werden, während DB2 for Linux, UNIX and Windows online ist. Dabei müssen Sie jedoch die folgenden Auswirkungen hinnehmen:
      - Möglicherweise können Sie die punktuelle Recovery nicht verwenden, um eine Wiederherstellung bis zu einem Zeitpunkt durchzuführen, der

innerhalb dieser *X* Minuten liegt. Dies bedeutet, dass Sie möglicherweise nicht in der Lage sind, eine Untergruppe der Aktualisierungstransaktionen wiederherzustellen, die innerhalb dieser X Minuten ausgeführt wurden.

- Funktionen, die innerhalb von *X* Minuten vor der Änderung erstellt wurden, können möglicherweise bis zu X Minuten nach der Änderung nicht aufgelöst werden.

3. Ändern Sie die Systemuhr.

Ergebnisse

Indem Sie die beschriebenen bewährten Verfahren befolgen, können Sie das Auftreten von Problemen bei der Änderung der Systemuhr in Bezug auf die punktuelle Recovery oder die Funktionsauflösung vermeiden.

Synchronisation der Primär- und der Bereitschaftsdatenbank

Eine der Strategien für hohe Verfügbarkeit ist es, über eine Primärdatenbank und eine sekundäre oder Bereitschaftsdatenbank zu verfügen; letztere übernimmt die anfallenden Operationen, falls die Primärdatenbank ausfällt.

Muss die Bereitschaftsdatenbank die Datenbankoperationen für eine ausgefallene Primärdatenbank übernehmen, müssen in ihr exakt die gleichen Daten enthalten sein, sämtliche unvollständigen Transaktionen müssen ihr bekannt sein, und sie muss ansonsten auf exakt die gleiche Art die Datenbankverarbeitung fortführen, als ob der primäre Datenbankserver nicht ausgefallen wäre. Der fortlaufende Prozess der Aktualisierung der Bereitschaftsdatenbank, sodass sie eine Kopie der Primärdatenbank darstellt, wird als Synchronisation bezeichnet.

Vorbereitende Schritte

Sie müssen zunächst die folgenden Aufgaben erledigen, bevor Sie die primäre und die Bereitschaftsdatenbank synchronisieren können:

- Erstellen und Konfigurieren der primären und der Bereitschaftsdatenbank.
- Konfigurieren der Datenübertragung zwischen der primären und der Bereitschaftsdatenbank.
-

  Auswählen einer Synchronisationsstrategie (z. B. Protokollübertragung, Protokollspiegelung, ausgesetzte E/A und Plattenspiegelung oder HADR).

  Es gibt verschiedene Strategien, um den primären Datenbankserver und den Bereitschaftsdatenbankserver synchron zu halten:

  – Das Übertragen von Protokollen von der Primärdatenbank an die Bereitschaftsdatenbank sowie anschließend eine aktualisierende Recovery der Protokolle in der Bereitschaftsdatenbank
  – Das gleichzeitige Schreiben von Datenbankprotokollen in die Primär- und die Bereitschaftsdatenbank, auch als Protokollspiegelung bezeichnet
  – Verwenden der Unterstützung für ausgesetzte E/A zusammen mit der Plattenspiegelung, um regelmäßig eine Kopie der Primärdatenbank zu erstellen, den Spiegel zu teilen und die Kopie als neuen Bereitschaftsdatenbankserver zu initialisieren
  – Das Verwenden einer Funktion für die Verfügbarkeit, wie z. B. DB2 High Availability Disaster Recovery (HADR), sodass die Primär- und die Bereitschaftsdatenbank immer synchronisiert sind.

Vorgehensweise

1. Wenn Sie Protokolle verwenden, um die Primärdatenbank und die sekundäre bzw. die Bereitschaftsdatenbank zu synchronisieren, konfigurieren Sie die DB2-Datenbank so, dass sie die erforderliche Protokollierungsverwaltung für Sie ausführt. Beispiel: Wenn Sie möchten, dass die DB2-Datenbank die Protokolle spiegelt, setzen Sie den Konfigurationsparameter **mirrorlogpath** auf die Speicherposition, in der die Kopie der Protokolle gespeichert werden soll.
2. Wenn Sie die Funktionalität für ausgesetzte E/A Ihrer DB2-Datenbank verwenden, um eine gespiegelte Platte der Primärdatenbank zu teilen, müssen Sie wie folgt vorgehen:
   a. Initialisieren Sie die Plattenspiegelung für die Primärdatenbank.
   b. Wenn Sie den Spiegel der Primärdatenbank teilen müssen, befolgen Sie die Anweisungen im Abschnitt „Verwenden einer geteilten Spiegeldatenbank als Bereitschaftsdatenbank".
3. Wenn Sie die HADR-Funktion verwenden, um die Synchronisation der Primär- und der Bereitschaftsdatenbank zu verwalten, konfigurieren Sie die DB2-Datenbank für HADR und richten Sie es so ein, dass die DB2-Datenbank das Synchronisieren der Primär- und der Bereitschaftsdatenbank für Sie übernimmt.

Auflösen von Protokollanwendungsfehlern beim Erstellen eines Tabellenbereichs

Auflösen von Protokollanwendungsfehlern beim Erstellen eines Tabellenbereichs

Wenn Sie einen Tabellenbereich für die Primärdatenbank erstellen und die Anwendung von Protokollen in der Bereitschaftsdatenbank fehlschlägt, weil die entsprechenden Container nicht verfügbar sind, empfängt die Primärdatenbank keine Fehlernachricht darüber, dass die Anwendung der Protokolle fehlgeschlagen ist.

Zur Überprüfung, ob beim Anwenden der Protokolle Fehler auftreten, müssen Sie für die Bereitschaftsdatenbank die Protokolldatei **db2diag** und die Protokolldatei mit Benachrichtigungen für die Systemverwaltung überwachen, wenn Sie neue Tabellenbereiche erstellen.

Wenn eine Übernahmeoperation stattfindet, steht dieser Tabellenbereich für die neue Primärdatenbank nicht zur Verfügung. Um dies zu beheben, führen Sie einen Restore des Tabellenbereichs in der neuen Primärdatenbank von einem Backup-Image durch.

Im folgenden Beispiel wird der Tabellenbereich **MEIN_TABELLENBEREICH** für die Datenbank **MEINE_DATENBANK** mit Restore wiederhergestellt, bevor er als Teil der neuen Primärdatenbank eingesetzt wird:

1. **db2 connect to meine_datenbank**
2. **db2 list tablespaces show detail**

   Anmerkung: Führen Sie den Befehl **db2 list tablespaces show detail** aus, um den Status aller Tabellenbereiche anzuzeigen und die für Schritt 5 erforderliche Tabellenbereichs-ID abzurufen.
3. db2 stop hadr on database meine_datenbank
4. db2 "restore database meine_datenbank tablespace (mein_tabellenbereich) online redirect"
5. db2 "set tablespace containers for ID_nr_meines_tabellenbereichs ignore rollforward container operations using (path '/mein_neuer_containerpfad/')"

6. db2 "restore database meine_datenbank continue"
7. db2 rollforward database meine_datenbank to end of logs and stop tablespace "(mein_tabellenbereich)"
8. db2 start hadr on database meine_datenbank as primary

Replizierte Operationen von DB2 High Availability Disaster Recovery (HADR)

DB2 High Availability Disaster Recovery (HADR) verwendet Datenbankprotokolle, um Daten aus der Primärdatenbank in der Bereitschaftsdatenbank zu replizieren. Einige Aktivitäten können dazu führen, dass die Bereitschaftsdatenbank nicht auf dem aktuellen Stand der Primärdatenbank bleibt, da die Protokolle auf der Bereitschaftsdatenbank angewendet werden.

Einige Aktivitäten werden so ausführlich protokolliert, dass die dadurch generierte umfangreiche Zahl an Protokolldateien Speicherprobleme verursachen kann. Obwohl das Replizieren von Daten in der Bereitschaftsdatenbank mithilfe von Protokollen zentraler Bestandteil von Verfügbarkeitsstrategien ist, kann die Protokollierung an sich potenziell negative Auswirkungen auf die Verfügbarkeit Ihrer Lösung haben. Gestalten Sie Ihre Verwaltungsstrategie so optimal wie möglich, und konfigurieren Sie Ihr System so, dass negative Auswirkungen der Protokollierung minimiert werden und die Protokollierung Ihre Transaktionsdaten schützt.

Wenn Sie HADR verwenden, werden die folgenden Operationen aus der Primärdatenbank in die Bereitschaftsdatenbank repliziert:

- Datendefinitionssprache (DDL - Data Definition Language)
- Datenbearbeitungssprache (DML - Data Manipulation Language)
- Pufferpooloperationen
- Tabellenbereichsoperationen
- Onlinereorganisation
- Offlinereorganisation
- Metadaten für gespeicherte Prozeduren und benutzerdefinierte Funktionen (UDF - User-defined Function), jedoch ohne zugehörige Objekt- oder Bibliotheksdateien

Während einer Onlinereorganisation werden alle Operationen detailliert protokolliert. Daher kann HADR die Operation replizieren, ohne dass die Bereitschaftsdatenbank weiter in Rückstand geraten würde, als dies bei herkömmlichen Datenbankaktualisierungen der Fall wäre. Dieses Verhalten kann potenziell jedoch große Auswirkungen auf das System haben, da eine große Anzahl Protokollsätze generiert wird.

Während Offlinereorganisationen nicht so umfassend protokolliert werden wie Onlinereorganisationen, werden Operationen in der Regel für hunderte oder tausende von betroffenen Zeilen protokolliert. Dies bedeutet, dass die Bereitschaftsdatenbank in Rückstand geraten kann, da sie auf jeden Protokollsatz wartet und anschließend gleichzeitig zahlreiche Aktualisierungen wiederholt. Wenn es sich bei Offlinereorganisation nicht um eine Clusterreorganisation handelt, wird nach der gesamten Reorganisation ein einzelner Protokollsatz generiert. Dieser Modus hat die größte Auswirkung darauf, ob und wie die Bereitschaftsdatenbank auf dem Stand der Primärdatenbank gehalten werden kann. Die Bereitschaftsdatenbank führt die gesamte Reorganisation aus, nachdem sie den Protokollsatz von der Primärdatenbank erhalten hat.

HADR repliziert weder gespeicherte Prozeduren noch UDF-Objektdateien oder Bibliotheksdateien. Sie müssen die Dateien in der Primärdatenbank und in der Bereitschaftsdatenbank in identischen Pfaden erstellen. Wenn die Bereitschaftsdatenbank das Referenzobjekt oder die Bibliotheksdatei nicht finden kann, wird der Aufruf der gespeicherten Prozedur oder UDF in der Bereitschaftsdatenbank fehlschlagen.

Nicht replizierte Operationen von DB2 High Availability Disaster Recovery (HADR)

DB2 High Availability Disaster Recovery (HADR) verwendet Datenbankprotokolle, um Daten aus der Primärdatenbank in die Bereitschaftsdatenbank zu replizieren. Nicht protokollierte Operationen sind in der Primärdatenbank zwar zulässig, sie werden aber nicht in die Bereitschaftsdatenbank repliziert.

Wenn Sie möchten, dass nicht protokollierte Operationen wie z. B. Aktualisierungen der Verlaufsdatei in der Bereitschaftsdatenbank abgebildet werden, sind dafür zusätzliche Schritte erforderlich.

Die folgenden Beispiele beschreiben Fälle, in denen Operationen in der Primärdatenbank nicht in die Bereitschaftsdatenbank repliziert werden:

- Tabellen, die mit der Option NOT LOGGED INITIALLY erstellt werden, werden nicht repliziert. Versuche, auf solche Tabellen zuzugreifen, nachdem eine HADR-Bereitschaftsdatenbank die Funktion als primäre Datenbank übernommen hat, enden mit einem Fehler.
- Alle protokollierten LOB-Spalten werden repliziert. Nicht protokollierte LOB-Spalten werden nicht repliziert. Der Speicherbereich für sie wird in der Bereitschaftsdatenbank jedoch zugeordnet, indem binäre Nullen als Werte für die Spalte verwendet werden.
- Aktualisierungen an der Datenbankkonfiguration, die mit den Befehlen **UPDATE DATABASE CONFIGURATION** und **UPDATE DATABASE MANAGER CONFIGURATION** ausgeführt werden, werden nicht repliziert.
- Datenbankkonfigurationsparameter und Datenbankmanagerkonfigurationsparameter werden nicht repliziert.
- Bei benutzerdefinierten Funktionen (UDFs) werden Änderungen an Objekten, die für die Datenbank extern sind (z. B. zugehörige Objekte und Bibliotheksdateien) nicht repliziert. Solchen Änderungen muss in der Bereitschaftsdatenbank mit anderen Mitteln Rechnung getragen werden.
- Die Datei des Recoveryprotokolls (`db2rhist.asc`) und daran vorgenommene Änderungen werden nicht automatisch von der Primärdatenbank an die Bereitschaftsdatenbank übertragen.

  Sie können eine Anfangskopie der Datei des Recoveryprotokolls (die Sie aus dem Backup-Image der Primärdatenbank abrufen) in die Bereitschaftsdatenbank kopieren, indem Sie den Befehl **RESTORE DATABASE** mit dem Parameter **REPLACE HISTORY FILE** absetzen:

  ```
  RESTORE DB KELLY REPLACE HISTORY FILE
  ```

  Nachdem HADR initialisiert wurde und nachfolgend in der Primärdatenbank Backup-Aktivitäten erfolgt sind, befindet sich die Datei des Recoveryprotokolls in der Bereitschaftsdatenbank nicht mehr auf dem aktuellen Stand. Eine Kopie der Datei des Recoveryprotokolls wird jedoch in jedem Backup-Image gespeichert.

Sie können die Datei des Recoveryprotokolls in der Bereitschaftsdatenbank aktualisieren, indem Sie die Datei des Recoveryprotokolls mithilfe des folgenden Befehls aus einem Backup-Image extrahieren:

```
RESTORE DB KELLY HISTORY FILE
```

Verwenden Sie nicht die regulären Betriebssystembefehle, um die Datei des Recoveryprotokolls im Datenbankverzeichnis aus der Primärdatenbank in die Bereitschaftsdatenbank zu kopieren. Die Datei des Recoveryprotokolls kann beschädigt werden, wenn die Primärdatenbank die Datei gerade ändert, wenn die Kopie ausgeführt wird.

Wenn eine Übernahmeoperation ausgeführt wird und die Bereitschaftsdatenbank über eine aktuelle Datei des Recoveryprotokolls verfügt, generieren Backup- und Restoreoperationen in der neuen Primärdatenbank neue Sätze in der Datei des Recoveryprotokolls, die sich nahtlos in die von der ursprünglichen Primärdatenbank generierten Sätze einfügen. Wenn die Datei des Recoveryprotokolls nicht auf dem neuesten Stand ist oder Einträge fehlen, kann eventuell kein automatischer inkrementeller Restore ausgeführt werden. In diesem Fall ist ein manueller inkrementeller Restore erforderlich.

Status von DB2-HADR-Bereitschaftsdatenbanken (High Availability Disaster Recovery)

Eine HADR-Bereitschaftsdatenbank befindet sich immer in einem von fünf möglichen Status: Lokales Catch-up, Fernes Catch-up anstehend, Fernes Catch-up, Peer und Unterbrochener Peer. Die Status werden über den Protokollübertragungsstatus definiert. Unabhängig vom Status werden gleichzeitig alle verfügbaren Protokolle wiedergegeben.

Die Protokollposition der Primärdatenbank, die Protokollempfangsposition der Bereitschaftsdatenbank und auch die Protokollwiedergabeposition der Bereitschaftsdatenbank werden von den Standardüberwachungsschnittstellen für HADR ausgegeben: der Tabellenfunktion MON_GET_HADR und dem Befehl **db2pd** mit dem Parameter **-hadr**. Der Status einer Bereitschaftsdatenbank wird im Feld HADR_STATE ausgegeben. Wenn eine Primärdatenbank mit einer Bereitschaftsdatenbank verbunden ist, gibt die Überwachungsschnittstelle den Status der Bereitschaftsdatenbank als HADR_STATE aus; andernfalls wird DISCONNECTED ausgegeben.

Abb. 10 auf Seite 208 zeigt den Verlauf über die unterschiedlichen Statuszustände der Bereitschaftsdatenbank.

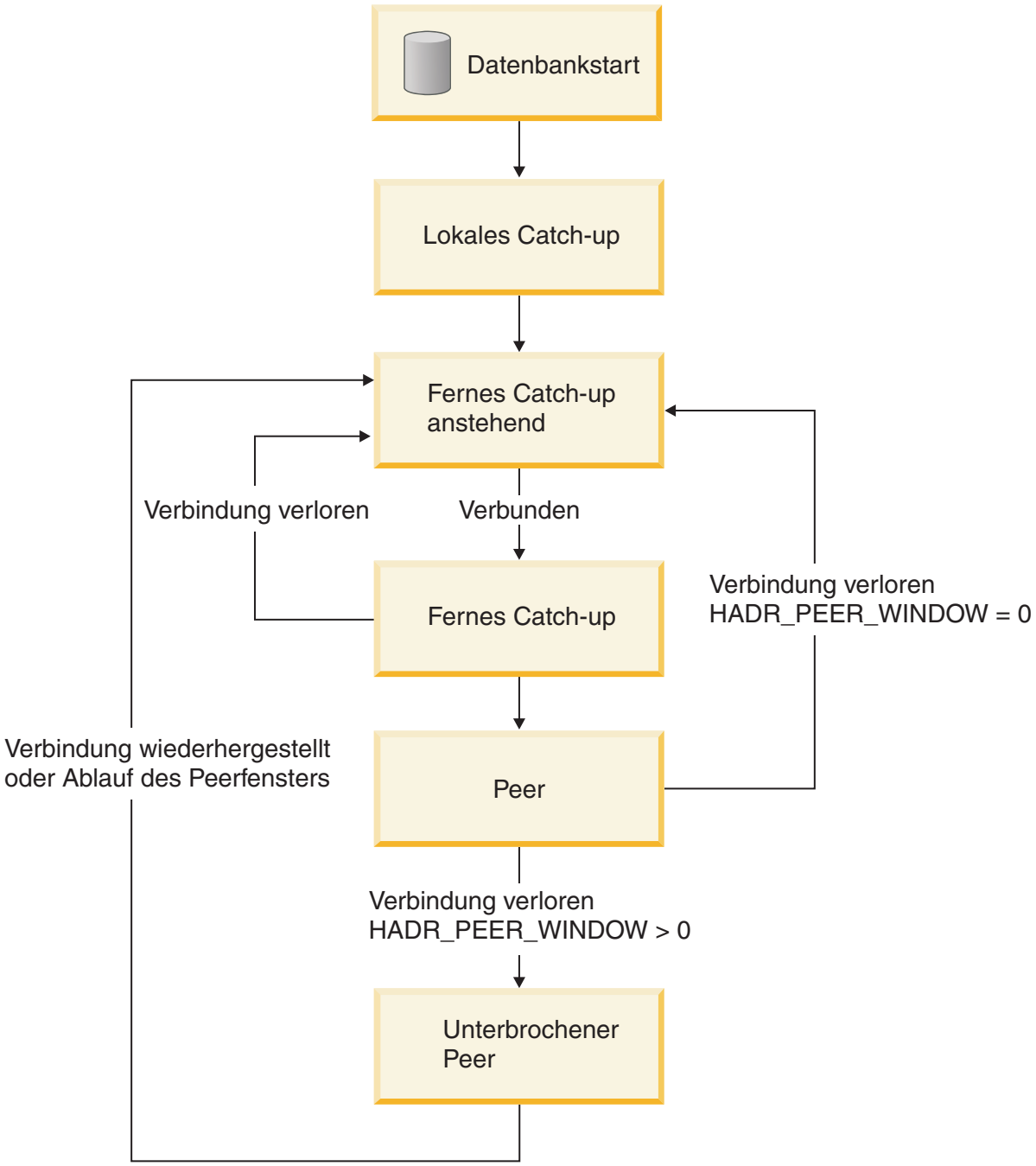


Abbildung 10. Status der Bereitschaftsdatenbank

Status 'Lokales Catch-up'

Wenn Sie die HADR-Funktion verwenden und eine Bereitschaftsdatenbank gestartet wird, erhält sie den Status 'Lokales Catch-up'. Die Protokolldateien im lokalen Protokollpfad werden gelesen, um festzustellen, welche Protokolle lokal verfügbar sind. In diesem Status werden auch dann keine Protokolle aus dem Archiv abgerufen, wenn eine Protokollarchivierungsmethode konfiguriert wurde. Ferner ist in diesem Status keine Verbindung zur Primärdatenbank erforderlich. Wenn jedoch keine Verbindung vorhanden ist, versucht die Bereitschaftsdatenbank, eine Verbindung zur Primärdatenbank herzustellen. Wenn das Ende der lokalen Protokollda-

teien erreicht wird, wechselt die Bereitschaftsdatenbank in den Status "Fernes Catch-up anstehend'.

Status 'Fernes Catch-up anstehend'

Wenn die Bereitschaftsdatenbank in den Status 'Fernes Catch-up anstehend' wechselt und keine Verbindung zur Primärdatenbank hergestellt wurde, dann wartet die Bereitschaftsdatenbank auf eine Verbindung. Nachdem eine Verbindung hergestellt wurde, ruft die Bereitschaftsdatenbank die aktuellen Protokollketteninformationen der Primärdatenbank ab. Sofern ein Protokollarchiv konfiguriert ist, wird die Bereitschaftsdatenbank dadurch in die Lage versetzt, Protokolldateien aus dem Archiv abzurufen und sicherzustellen, dass die Protokolldateien gültig sind.

Im Status 'Fernes Catch-up' und im Status 'Peer' wechselt die Bereitschaftsdatenbank wieder in den Status 'Fernes Catch-up anstehend', wenn die Verbindung zur Primärdatenbank unterbrochen wird. Nach erneuter Herstellung der Verbindung versucht die Bereitschaftsdatenbank, die Protokolle aus dem Archiv abzurufen. Wenn Sie also eine gemeinsam genutzte Archivierungseinheit konfigurieren, findet die Bereitschaftsdatenbank möglicherweise mehr Protokolle als bei Verwendung einer separaten Archivierungseinheit. Dieses Verhalten zieht das Abrufen von Protokollen aus dem Archiv der Protokollübertragung aus der Primärdatenbank über die HADR-Verbindung vor, um die Auswirkungen auf die Primärdatenbank möglichst gering zu halten.

Status 'Fernes Catch-up'

Im Status 'Fernes Catch-up' werden Protokolldaten von der Primärdatenbank aus dem Protokollpfad oder mithilfe einer Protokollarchivierungsmethode gelesen und an die Bereitschaftsdatenbank gesendet. Die Primär- und die Bereitschaftsdatenbank wechseln in den Peerstatus, wenn die Bereitschaftsdatenbank alle Protokolldaten der Primärdatenbank empfangen hat, die sich auf Platte befinden. Bei Verwendung des Synchronisationsmodus SUPERASYNC wechseln die Primär- und die Primärdatenbank nie in den Peerstatus. Sie verbleiben dauerhaft im Status 'Fernes Catch-up'. Dadurch wird verhindert, dass das Schreiben von Protokollen der Primärdatenbank im Peerstatus blockiert wird.

Geht die Verbindung zwischen der Primärdatenbank und der Bereitschaftsdatenbank verloren, wenn sich die Datenbanken im Status 'Fernes Catch-up' befinden, wechselt die Bereitschaftsdatenbank in den Status 'Fernes Catch-up anstehend'.

Peerstatus

Im Peerstatus werden Protokolldaten direkt aus dem Puffer für Protokollschreibvorgänge der Primärdatenbank in die Bereitschaftsdatenbank übertragen, sobald die Primärdatenbank ihre Protokollseiten auf Platte schreibt. Der HADR-Synchronisationsmodus legt fest, ob die Primärdatenbank darauf wartet, dass die Bereitschaftsdatenbank eine Bestätigungsnachricht bezüglich des Empfangs der Protokolldaten sendet. Die Protokollseiten werden immer in die lokalen Protokolldateien in der Bereitschaftsdatenbank geschrieben. Dieses Verhalten schützt vor Ausfällen und bietet die Möglichkeit, eine Datei im Falle einer Übernahme auf der neuen Primärdatenbank zu archivieren, wenn sie nicht auf der alten Primärdatenbank archiviert wurde. Nachdem sie auf eine lokale Platte geschrieben wurden, können die empfangenen Protokollseiten in der Bereitschaftsdatenbank wiedergegeben werden. Wenn das Protokollspooling inaktiviert ist (Standardeinstellung), werden Protokolle von der Wiedergabe nur aus dem Protokollempfangspuffer gelesen.

Wenn die Protokollwiederholung langsam ist, füllt sich möglicherweise der Protokollempfangspuffer und die Bereitschaftsdatenbank empfängt keine neuen Protokolle mehr. In diesem Fall werden Protokollschreibvorgänge der Primärdatenbank blockiert. Wenn Sie das Protokollspooling aktivieren, wird ein Teil des Protokollpuffers freigegeben, auch wenn er noch nicht wiedergegeben wurde. Die Protokolldaten werden von der Protokollwiedergabe zu einen späteren Zeitpunkt von Platte gelesen. Wenn die Spoolingeinheit voll ist oder das konfigurierte Spoollimit erreicht wurde, empfängt die Bereitschaftsdatenbank weiterhin keine Protokolle und die Primärdatenbank kann weiterhin blockiert sein.

Geht die Verbindung zwischen der Primär- und der Bereitschaftsdatenbank verloren, wenn sich die Datenbanken im Peerstatus befinden und der Datenbankkonfigurationsparameter **hadr_peer_window** auf `0` (Standardeinstellung) gesetzt ist, wechselt die Bereitschaftsdatenbank in den Status 'Fernes Catch-up anstehend'. Geht die Verbindung zwischen der Primär- und der Bereitschaftsdatenbank jedoch verloren, wenn sich die Datenbanken im Peerstatus befinden und der Datenbankkonfigurationsparameter **hadr_peer_window** auf einen Wert ungleich null gesetzt ist (was bedeutet, dass ein *Peerfenster* konfiguriert wurde), wechselt die Bereitschaftsdatenbank in den Status 'Unterbrochener Peer'.

Status 'Unterbrochener Peer'

Wenn Sie ein Peerfenster konfiguriert haben und die Primärdatenbank die Verbindung mit der Bereitschaftsdatenbank im Peerstatus verliert, verhält sich die Primärdatenbank weiterhin so, als ob sich die Primär- und die Bereitschaftsdatenbank im Peerstatus befinden würden. Dies dauert entweder bis zum Ablauf des konfigurierten Zeitraums (des so genannten *Peerfensters*) oder bis die Bereitschaftsdatenbank die Verbindung wiederherstellt - je nachdem, welches Ereignis zuerst eintritt. Wenn die Verbindung zwischen der Primär- und der Bereitschaftsdatenbank getrennt wird und sich die Datenbanken weiterhin so verhalten, als ob sie sich im Peerstatus befinden würden, wird dieser Status als *Unterbrochener Peer* bezeichnet.

Die Konfiguration eines Peerfensters hat den Vorteil, dass das Risiko eines Transaktionsverlusts bei mehreren oder kaskadierenden Ausfällen niedriger ist. Wenn die Primärdatenbank die Verbindung mit der Bereitschaftsdatenbank verliert und kein Peerfenster definiert ist, verlässt die Primärdatenbank umgehend den Peerstatus und setzt die Transaktionsverarbeitung fort. Diese Transaktionen werden nicht in die Bereitschaftsdatenbank repliziert. Wenn der Primärserver ausfällt, kurz nachdem die Verbindung mit der Bereitschaftsdatenbank unterbrochen wurde, ist das Risiko von Transaktionsverlusten bei einer Funktionsübernahme hoch. Wenn das Peerfenster aktiviert ist, blockiert die Primärdatenbank die Transaktionsverarbeitung für einem bestimmten Zeitraum, nachdem die Verbindung mit der Bereitschaftsdatenbank im Peerstatus unterbrochen wurde, um kaskadierende Ausfälle zu vermeiden. Darüber hinaus ist innerhalb des Peerfensters eine Übernahme durch die Bereitschaftsdatenbank ohne das Risiko von Datenverlusten möglich.

Die Konfiguration eines Peerfensters hat den Nachteil, dass Transaktionen auf der Primärdatenbank länger dauern oder sogar das Zeitlimit überschreiten können, während die Primärdatenbank innerhalb des Peerfensters auf eine Wiederherstellung der Verbindung mit der Bereitschaftsdatenbank oder den Ablauf des Peerfensters wartet. Ferner können zeitweilig auftretende Netzstörungen negative Auswirkungen auf die Transaktionsverarbeitung in der Primärdatenbank haben.

Unter Verwendung der Tabellenfunktion MON_GET_HADR oder des Befehls **db2pd** mit dem Parameter **-hadr** können Sie die Größe des Peerfensters festlegen, die dem Wert des Datenbankkonfigurationsparameters **hadr_peer_window** entspricht.

Manuelles Kopieren von Protokolldateien aus der Primär- in die Bereitschaftsdatenbank

Eine Methode der Synchronisation von Primär- und Bereitschaftsdatenbank besteht darin, die Protokolldateien der Primärdatenbank manuell in den Protokollpfad oder den Überlaufprotokollpfad (sofern konfiguriert) der Bereitschaftsdatenbank zu kopieren. Diese Methode kann insbesondere hilfreich sein, wenn zwischen der Primär- und der Bereitschaftsdatenbank eine große Protokollabstimmungsdiskrepanz herrscht (z. B. weil die Bereitschaftsdatenbank lange Zeit inaktiv war). So können die Verzögerungen reduziert werden, die sich daraus ergeben, dass die Bereitschaftsdatenbank die Protokolle aus dem Archiv abrufen muss. Möglicherweise sind auch die Auswirkungen auf die Primärdatenbank geringer, die sich daraus ergeben, dass diese Protokolldateien (die die Primärdatenbank wahrscheinlich aus dem Archiv abrufen muss) übertragen werden müssen. Es ist wichtig, dass dieser Schritt vor der Aktivierung der Bereitschaftsdatenbank ausgeführt wird. Nach der Aktivierung der Bereitschaftsdatenbank fährt diese mit der Suche lokaler Protokolldateien fort. Dabei versucht sie, Dateien aus dem Archiv abzurufen, und sie veranlasst die Primärdatenbank zur Protokollübertragung (siehe oben). Das Kopieren der Protokolldateien auf die Bereitschaftsdatenbank nach der Aktivierung stört den normalen Betrieb der Datenbank.

Bestimmen des Status von HADR-Bereitschaftsdatenbanken

Der Status einer DB2-HADR-Bereitschaftsdatenbank bestimmt, welche Operationen diese Datenbank ausführen kann. Es gibt zwei Optionen, die zur Bestimmung des Status einer Bereitschaftsdatenbank geeignet sind: der Befehl **db2pd** und die Tabellenfunktion MON_GET_HADR.

Vorgehensweise

Gehen Sie wie folgt vor, um den Status einer HADR-Bereitschaftsdatenbank in einem HADR-Datenbankpaar (Primär- und Bereitschaftsdatenbank) zu bestimmen:

- Setzen Sie den Befehl **db2pd** mit dem Parameter **-hadr** auf der Primär- oder der Bereitschaftsdatenbank ab.
  - Wenn Sie den Befehl in der Primärdatenbank absetzen, gibt er für jede Bereitschaftsdatenbank in der HADR-Konfiguration eine Gruppe von Daten zurück.
  - Wenn Sie den Befehl in der Bereitschaftsdatenbank absetzen, gibt der Befehl nur eine einzige Gruppe von Daten zurück, da der Bereitschaftsdatenbank andere Bereitschaftsdatenbanken nicht bekannt sind, auch wenn sich die HADR-Konfiguration im Modus für mehrere Bereitschaftsdatenbanken befindet.
- Führen Sie auf der Primär- oder der Bereitschaftsdatenbank eine Abfrage mit der Tabellenfunktion MON_GET_HADR aus:

  ```
  db2 "select STANDBY_ID, HADR_STATE, from table (mon_get_hadr(NULL))"
  ```

  Die folgenden Informationen werden zurückgegeben:

  ```
  STANDBY_ID HADR_STATE
  ---------- -----------------------
           1 PEER
           2 REMOTE_CATCHUP
           3 REMOTE_CATCHUP

   3 record(s) selected.
  ```

  - Wenn Sie die Abfrage für die Primärdatenbank ausführen, gibt die Tabellenfunktion eine Zeile mit Informationen für jede Bereitschaftsdatenbank in der HADR-Konfiguration zurück.

– Wenn Sie die Abfrage für die Bereitschaftsdatenbank ausführen, gibt die Tabellenfunktion nur eine Zeile mit Informationen zurück, da der Bereitschaftsdatenbank keine anderen Bereitschaftsdatenbanken bekannt sind, auch wenn sich die HADR-Konfiguration im Modus für mehrere Bereitschaftsdatenbanken befindet.

Recovery nach Tabellenbereichsfehlern in einer HADR-Bereitschaftsdatenbank

Wenn bei der HADR-Bereitschaftsdatenbank während der Anwendung der Protokolle ein Fehler bei einem bestimmten Tabellenbereich auftritt, setzt die Bereitschaftsdatenbank die Anwendung der Protokolle auf andere Tabellenbereiche fort, stoppt jedoch die Anwendung der Protokolle auf den betroffenen Tabellenbereich.

Informationen zu diesem Vorgang

Der Tabellenbereichsstatus des betroffenen Tabellenbereichs wird in 'Restore anstehend', 'aktualisierende Recovery anstehend' oder 'offline' geändert. Sie müssen den Tabellenbereich in der Bereitschaftsdatenbank wiederherstellen, da Daten in diesem Tabellenbereich nicht verfügbar sein werden, wenn diese Datenbank die primäre Rolle übernimmt.

Vorgehensweise

1. Korrigieren Sie die dem Fehler zugrunde liegende Ursache. Folgendes können mögliche Ursachen sein:
   - Nicht genügend freier Speicherbereich
   - Das Dateisystem ist nicht angehängt
   - Es konnte keine Ladekopie gefunden werden
2. Reparieren Sie den betroffenen Tabellenbereich. Dazu müssen Sie die Bereitschaftsdatenbank vollständig reinitialisieren, indem Sie einen Restore eines Backup-Images der Primärdatenbank durchführen.

HADR-Rollenwechsel und Tabellenbereiche im Quiescemodus

In einer Umgebung mit HADR (High Availability Disaster Recovery) bleibt während eines Rollenwechsels der Quiescemodus für einen Tabellenbereich nicht erhalten.

Wenn ein Tabellenbereich bei der Primärdatenbank in den Quiescemodus versetzt wird, werden keine Protokollsätze generiert. Die Operation wirkt sich daher nicht auf die Bereitschaftsdatenbank aus. Falls die Bereitschaftsdatenbank vor der Freigabe des Quiescemodus die Rolle der Primärdatenbank übernehmen muss, ist der entsprechende Tabellenbereich in der neuen Primärdatenbank voll verfügbar. Sie müssen beachten, dass bei einer Fortsetzung des Jobs, für den der Tabellenbereich in der ursprünglichen Primärdatenbank in den Quiescemodus versetzt werden musste, der Job in der neuen Primärdatenbank nicht mehr durch den Quiescemodus geschützt ist.

Falls ein Rollenwechsel stattgefunden hat (die alte Primärdatenbank nun also die neue Bereitschaftsdatenbank ist), werden Änderungen am Tabellenbereich in der neuen Primärdatenbank in der neuen Bereitschaftsdatenbank wiedergegeben. Wird die Primärrolle zurück zur alten Primärdatenbank vesetzt, bleibt der Quiescestatus jedoch für diesen Tabellenbereich weiterhin wirksam.

Verzögerte Wiedergabe bei HADR

Die verzögerte Wiedergabe bei HADR trägt dazu bei, Datenverluste aufgrund von fehlerhaften Transaktionen zu vermeiden. Zur Implementierung der verzögerten Wiedergabe bei HADR müssen Sie den Datenbankkonfigurationsparameter **hadr_replay_delay** auf der HADR-Bereitschaftsdatenbank definieren.

Die verzögerte Wiedergabe setzt für die Bereitschaftsdatenbank absichtlich eine frühere Zeit an als für die Primärdatenbank, indem die Wiedergabe von Protokollen auf dieser Bereitschaftsdatenbank verzögert wird. Wenn in der Primärdatenbank eine fehlerhafte Transaktion ausgeführt wird, können Sie die konfigurierte Zeitverzögerung nutzen, um Maßnahmen zu ergreifen, die verhindern, dass die fehlerhafte Transaktion in der Bereitschaftsdatenbank wiedergegeben wird. Zur Wiederherstellung der verloren gegangenen Daten können Sie diese Daten entweder zurück in die Primärdatenbank kopieren oder angeben, dass die Bereitschaftsdatenbank die Rolle als neue Primärdatenbank übernehmen soll.

Die verzögerte Wiedergabe vergleicht Zeitmarken innerhalb des Protokolldatenstroms, der in der Primärdatenbank generiert wird, und die aktuelle Uhrzeit der Bereitschaftsdatenbank. Daher ist es wichtig, dass die Uhren der Primärdatenbank und der Bereitschaftsdatenbanken synchron bleiben. Das Commit der Transaktion wird in der Bereitschaftsdatenbank gemäß der folgenden Gleichung wiedergegeben:

```
(Aktuelle Zeit auf Bereitschaftsdatenbank  - Wert des Konfigurationsparameters
hadr_replay_delay) >= Zeitmarke des festgeschriebenen Protokolldatensatzes
```

Für den Datenbankkonfigurationsparameter **hadr_replay_delay** sollte ein ausreichend hoher Wert angegeben werden, um genügend Zeit zur Erkennung fehlerhafter Transaktionen in der Primärdatenbank und zur Ergreifung entsprechender Maßnahmen zu erhalten.

Sie können diese Funktion entweder im Modus für eine Bereitschaftsdatenbank oder im Modus für mehrere Bereitschaftsdatenbanken verwenden. Im Modus für mehrere Bereitschaftsdatenbanken bleibt normalerweise mindestens eine Bereitschaftsdatenbank zu HA- oder DR-Zwecken mit der Primärdatenbank synchronisiert. Eine Bereitschaftsdatenbank wird zum Schutz vor fehlerhaften Transaktionen für die verzögerte Wiederhabe konfiguriert. Wenn Sie diese Funktion im Modus für eine Bereitschaftsdatenbank verwenden, sollte IBM Tivoli System Automation for Multiplatforms nicht aktiviert werden, da die Übernahme dann fehlschlägt.

Für die verzögerte Wiedergabe gelten die folgenden wichtigen Einschränkungen:

- Der Konfigurationsparameter **hadr_replay_delay** kann nur auf der Bereitschaftsdatenbank definiert werden.
- Wenn der Befehl **TAKEOVER** auf einer Bereitschaftsdatenbank abgesetzt wird, auf der die verzögerte Wiedergabe aktiviert ist, schlägt dieser Befehl fehl. Sie müssen zuerst den Konfigurationsparameter **hadr_replay_delay** auf 0 setzen und anschließend die Bereitschaftsdatenbank inaktivieren und erneut aktivieren, damit der neue Wert übernommen wird. Dann können Sie den Befehl **TAKEOVER** absetzen.
- Die Funktion für die verzögerte Wiedergabe wird nur im Modus SUPERASYNC unterstützt. Da die Protokollwiedergabe verzögert ist, kann sich auf der Bereitschaftsdatenbank eine große Menge nicht wiedergegebener Protokolldaten ansammeln, die Empfangspuffer und Spool (sofern konfiguriert) füllen. In anderen Synchronisationsmodi würde dies dazu führen, dass die Primärdatenbank blockiert wird.

Diese Funktion dient zum Schutz gegen Anwendungsfehler. Wenn Sie diese Funktion benutzen und dabei sicherstellen wollen, dass es im Falle eines Ausfalls der Primärdatenbank zu keinen Datenverlusten kommt, sollten Sie in Erwägung ziehen, eine Konfiguration mit mehreren Bereitschaftsdatenbanken zu implementieren, die ein höheres Maß an Synchronisation in Bezug auf die Hauptbereitschaftsdatenbank aufweist.

Empfehlungen

Verzögerte Wiedergabe und Disaster-Recovery
Wenn Sie eine Bereitschaftsdatenbank für DR-Zwecke und zum Schutz vor fehlerhaften Transaktionen nutzen, dann sollten Sie mit einer kurzen Zeitverzögerung arbeiten.

Funktion für verzögerte Wiedergabe und HADR-Funktion für Leseoperationen in der Bereitschaftsdatenbank
Wenn Sie eine Bereitschaftsdatenbank für Leseoperationen in der Bereitschaftsdatenbank nutzen wollen, dann sollten Sie mit einer kurzen Zeitverzögerung arbeiten, um aktuellere Daten für Lesersitzungen bereitstellen zu können. Da Leseoperationen in der Bereitschaftsdatenbank mit der Isolationsstufe UR (Uncommitted Read) ausgeführt werden, können angewendete, aber noch nicht festgeschriebene Änderungen festgestellt werden, die technisch gesehen noch nicht wiedergegeben wurden. Diese nicht festgeschriebenen Transaktionen können im Rahmen einer Recoveryprozedur bei fehlerhaften Transaktionen rückgängig gemacht werden, wenn Sie für die Bereitschaftsdatenbank eine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt durchführen und dann stoppen.

Verzögerte Wiedergabe und Protokollspooling
Bei Aktivierung der verzögerten Wiedergabe wird empfohlen, das Protokollspooling ebenfalls zu aktivieren, indem Sie den Konfigurationsparameter **hadr_spool_limit** definieren. Aufgrund der beabsichtigten Verzögerung kann die Wiedergabeposition weit hinter der Protokollempfangsposition in der Bereitschaftsdatenbank liegen. Ohne Spooling kann der Protokollempfang nur um die Menge des Empfangspuffers über die Wiedergabe hinausgehen. Mit Spooling kann die Bereitschaftsdatenbank über die Wiedergabeposition hinaus deutlich mehr Protokolle empfangen, was einen höheren Schutz vor Datenverlusten im Falle eines Ausfalls der Primärdatenbank bietet. Beachten Sie, dass die Primärdatenbank aufgrund des obligatorischen Modus SUPERASYNC in beiden Fällen von der verzögerten Wiedergabe nicht blockiert wird.

Datenrecovery mit verzögerter Wiedergabe bei HADR

Mit der Funktion für zeitverzögerte Wiedergabe bei HADR kann eine Recovery für Daten durchgeführt werden, die aufgrund einer fehlerhaften Transaktion auf der Primärdatenbank verloren gegangen sind. Dazu wird HADR auf einer Bereitschaftsdatenbank gestoppt, bevor diese Transaktion wiedergegeben wird.

Vorbereitende Schritte

Die verzögerte Wiedergabe muss für die Bereitschaftsdatenbank bereits aktiviert sein.

Wenn die Protokollwiedergabe auf der Bereitschaftsdatenbank gemäß STANDBY_REPLAY_LOG_TIME den Commitzeitpunkt für die fehlerhafte Transaktion auf der Bereitschaftsdatenbank überschritten hat, kann mit der folgenden Prozedur keine Datenrecovery durchgeführt werden. Sie können den Wert STANDBY_REPLAY-

_LOG_TIME bestimmen, indem Sie den Befehl **db2pd** mit dem Parameter **-hadr** oder die Tabellenfunktion MON_GET_HADR verwenden.

Einschränkung: Eine Bereitschaftsdatenbank, für die der Konfigurationsparameter **hadr_replay_delay** definiert wird, kann die Funktion einer Primärdatenbank erst dann übernehmen, nachdem die verzögerte Wiedergabe auf dieser Bereitschaftsdatenbank inaktiviert wurde.

Vorgehensweise

Führen Sie die folgenden Schritte auf der Bereitschaftsdatenbank aus, auf der Sie die verzögerte Wiedergabe aktiviert haben, um eine Recovery aufgrund einer fehlerhaften Transaktion durchzuführen:

1. Überprüfen Sie den zeitlichen Ablauf:
   a. Stellen Sie sicher, dass die Bereitschaftsdatenbank die Transaktion noch nicht wiedergegeben hat. Der Wert für STANDBY_REPLAY_LOG_TIME darf den Commitzeitpunkt der fehlerhaften Transaktion noch nicht erreicht haben.
   b. Stellen Sie sicher, dass die Bereitschaftsdatenbank die relevanten Protokolle empfangen hat. Der Wert für STANDBY_LOG_TIME, der die empfangenen Protokolle angibt, muss einen Zeitpunkt vor, aber nahe dem Commitzeitpunkt der fehlerhaften Transaktion erreicht haben. Dieser Zeitpunkt ist der in Schritt 3 verwendete Zeitpunkt der aktualisierenden Recovery. Wenn die Bereitschaftsdatenbank noch nicht genügend Protokolldateien empfangen hat, können Sie warten, bis weitere Protokolle übertragen werden, haben aber das Risiko, dass die Wiedergabezeit den Zeitpunkt der fehlerhaften Transaktionen erreicht. Wenn die Verzögerung beispielsweise 1 Stunde beträgt, sollte HADR spätestens 50 Minuten nach dem Zeitpunkt der fehlerhaften Transaktion gestoppt werden (einschließlich einer zehnminütigen Sicherheitszeitspanne), auch wenn die Protokollübertragung den gewünschten Zeitpunkt noch nicht erreicht hat.

      Wenn alternativ ein gemeinsam genutztes Protokollarchiv verfügbar ist und die Protokolle bereits archiviert sind, ist keine Wartezeit erforderlich. Wenn die Protokolle noch nicht archiviert sind, können diese mit dem Befehl **ARCHIVE LOG** archiviert werden. Andernfalls können fertiggestellte Protokolldateien vom Benutzer manuell aus der Primärdatenbank in die zeitverzögerte Bereitschaftsdatenbank kopiert werden. Verwenden Sie dazu vorzugsweise den Überlaufprotokollpfad oder ansonsten den Protokollpfad. Für diese alternativen Methoden muss die Bereitschaftsdatenbank zuerst inaktiviert werden, um Interferenzen mit der Protokollübertragung und -wiedergabe der Bereitschaftsdatenbank zu vermeiden.

   Sie können diese Zeiten festlegen, indem Sie den Befehl `db2pd -db dbname -hadr` absetzen oder die Funktion für Leseoperationen in der Bereitschaftsdatenbank aktivieren und dann die folgende Abfrage ausführen, die die Tabellenfunktion MON_GET_HADR verwendet:

   ```
   DB2 "select HADR_ROLE, STANDBY_ID, STANDBY_LOG_TIME, STANDBY_REPLAY_LOG_TIME,
   varchar(PRIMARY_MEMBER_HOST,20) as PRIMARY_MEMBER_HOST,
   varchar(STANDBY_MEMBER_HOST,20) as STANDBY_MEMBER_HOST
   from table (mon_get_hadr(NULL))"
   ```

2. Stoppen Sie HADR in der Bereitschaftsdatenbank:

   ```
   DB2 STOP HADR ON DATABASE dbname
   ```

3. Führen Sie eine aktualisierende Recovery der Bereitschaftsdatenbank bis zum gewünschten Zeitpunkt durch und stoppen Sie dann die Datenbank:

   ```
   DB2 ROLLFORWARD DB dbname to time-stamp and STOP
   ```

4. Verwenden Sie eine der folgenden Methoden:
   - Führen Sie ein Restore der verloren gegangenen Daten auf der Primärdatenbank durch:
     a. Kopieren Sie die betroffenen Daten aus der Bereitschaftsdatenbank und senden Sie diese zurück zur Primärdatenbank.

        Wenn die fehlerhafte Transaktion eine Tabelle gelöscht hat, können Sie diese auf der Bereitschaftsdatenbank exportieren und in die Primärdatenbank importieren. Wenn die fehlerhafte Transaktion Zeilen aus einer Tabelle gelöscht hat, können Sie die Tabelle auf der Bereitschaftsdatenbank exportieren und auf der Primärdatenbank eine IMPORT REPLACE-Operation verwenden.
     b. Die Bereitschaftsdatenbank mit verzögerter Wiedergabe muss reinitialisiert werden, weil der Protokolldatenstrom dieser Datenbank vom Protokolldatenstrom der Primärdatenbank abweicht. Auf allen anderen Bereitschaftsdatenbanken sind keine Aktionen erforderlich, da diese Datenbanken weiterhin der Primärdatenbank folgen und behobene Datenfehler auf diese Datenbanken repliziert werden.
     c. Stellen Sie die Datenbank mit einem auf der Primärdatenbank erstellten Backup-Image wieder her. Dabei kann es sich um ein zu einem beliebigen Zeitpunkt erstelltes Image handeln.
     d. Entfernen Sie alle Protokolldateien im Protokollpfad der Bereitschaftsdatenbank. Dieser Schritt ist wichtig. Der in Schritt 3 abgesetzte Befehl **ROLLFORWARD**... STOP hat dazu geführt, dass der Protokolldatenstrom dieser Datenbank vom Protokolldatenstrom der Primärdatenbank abweicht. Wenn die Dateien nicht entfernt werden, folgt die neu wiederhergestellte Datenbank diesem Protokolldatenstrom und weicht ebenfalls von der Primärdatenbank ab. Alternativ kann die Datenbank vor dem Restore für einen kompletten Neustart gelöscht werden. In diesem Fall geht allerdings auch die aktuelle Konfiguration einschließlich der HADR-Konfiguration verloren.
     e. Setzen Sie den Befehl **START HADR** mit der Option AS STANDBY auf der Datenbank ab. Daraufhin wird die Datenbank aktiviert und stellt eine Verbindung zur Primärdatenbank her.
   - Legen Sie die Bereitschaftsdatenbank mit den unbeschädigten Daten als Primärdatenbank fest:
     a. Beenden Sie die alte Primärdatenbank, um eine Split-Brain-Situation auszuschließen.
     b. Setzen Sie den Konfigurationsparameter **hadr_replay_delay** auf der Datenbank mit verzögerter Wiedergabe auf 0. Rekonfigurieren Sie gegebenenfalls weitere Parameter wie beispielsweise **hadr_target_list**. Setzen Sie dann auf der Datenbank den Befehl **START HADR** mit der Option AS PRIMARY BY FORCE ab, um die Datenbank in die neue Primärdatenbank zu konvertieren. Verwenden Sie die Option BY FORCE, weil nicht sichergestellt ist, dass die konfigurierte Hauptbereitschaftsdatenbank (die möglicherweise die alte Primärdatenbank ist) eine Verbindung herstellen kann.
     c. Leiten Sie die Clients auf die neue Primärdatenbank um.
     d. Die anderen Bereitschaftsdatenbanken werden automatisch auf die neue Primärdatenbank umgeleitet. Wenn eine Bereitschaftsdatenbank jedoch über den Abweichungspunkt zwischen alter und neuer Primärdatenbank (siehe Zeitpunkt in Schritt 3) hinaus Protokolle von der alten Primärdatenbank empfangen hat, wird diese von der neuen Primärdatenbank zurückgewiesen. In diesem Fall muss die Bereitschaftsdatenbank auf dieselbe Weise reinitialisiert werden wie die alte Primärdatenbank.

e. Reinitialisieren Sie die alte Primärdatenbank, weil der Protokolldatenstrom dieser Datenbank vom Protokolldatenstrom der neuen Primärdatenbank abweicht.
f. Stellen Sie die Datenbank mit einem Backup-Image wieder her, das auf der neuen Primärdatenbank oder vor dem in Schritt 3 angegebenen Zeitpunkt auf der alten Primärdatenbank erstellt wurde.
g. Entfernen Sie alle Protokolldateien im Protokollpfad. Andernfalls folgt die neu wiederhergestellte Datenbank dem Protokolldatenstrom der alten Primärdatenbank und weicht dann von der neuen Primärdatenbank ab. Alternativ kann die Datenbank vor dem Restore gelöscht werden, um einen kompletten Neustart durchzuführen. In diesem Fall geht allerdings die aktuelle Konfiguration einschließlich der HADR-Konfiguration verloren.
h. Setzen Sie den Befehl **START HADR** mit der Option AS STANDBY auf der Datenbank ab. Daraufhin wird die Datenbank aktiviert und stellt eine Verbindung zur Primärdatenbank her.

Verwaltung von DB2 High Availability Disaster Recovery (HADR)

Zur Verwaltung von DB2 High Availability Disaster Recovery (HADR) gehört das Konfigurieren und Verwalten des Status Ihres HADR-Systems.

Die HADR-Verwaltung umfasst u. a. die folgenden Tasks:

- Katalogisieren einer HADR-Datenbank.
- „Initialisieren von High Availability Disaster Recovery (HADR)" auf Seite 38
- Überprüfen oder Ändern der HADR-bezogenen Datenbankkonfigurationsparameter
- „Wechseln von Datenbankrollen bei High Availability Disaster Recovery (HADR)" auf Seite 266
- „Ausführen einer HADR-Funktionsübernahmeoperation" auf Seite 263
- „HADR-Überwachung" auf Seite 257
- „Stoppen von DB2 High Availability Disaster Recovery (HADR)" auf Seite 188

Zur Verwaltung von HADR stehen die folgenden Möglichkeiten zur Verfügung:

- Befehlszeilenprozessor
- DB2-Administrator-API
- Taskassistenten zur Verwaltung von HADR in IBM Data Studio ab Version 3.1

Zugehörige Informationen:

Datenbanken mit Taskassistenten verwalten

DB2-HADR-Befehle

Die Funktion DB2 High Availability Disaster Recovery (HADR) umfasst u. a. komplexe Protokollierung, die Funktionsübernahme (Failover) und eine Recoveryfunktionalität für hoch verfügbare DB2-Datenbanklösungen.

Trotz der Komplexität der Funktionalität, die HADR bietet, gibt es nur einige wenige Aktionen, deren Ausführung Sie HADR direkt befehlen müssen: HADR starten und stoppen sowie die Bereitschaftsdatenbank veranlassen, die Operationen als Primärdatenbank zu übernehmen.

Zur Verwaltung von High Availability Disaster Recover (HADR) werden drei Befehle zur Verfügung gestellt:

- **START HADR**
- **STOP HADR**
- **TAKEOVER HADR**

Zum Aufrufen dieser Befehle können Sie den Befehlszeilenprozessor oder die Administrator-API verwenden.

Durch Absetzen des Befehls **START HADR** mit der Option AS PRIMARY oder AS STANDBY wird die Rolle der Datenbank in die angegebene Rolle geändert, sofern die Datenbank diese Rolle nicht bereits hat. Darüber hinaus aktiviert dieser Befehl die Datenbank, wenn sie nicht bereits aktiviert ist.

Der Befehl **STOP HADR** ändert eine HADR-Datenbank (Primär- oder Bereitschaftsdatenbank) in eine Standarddatenbank. Alle auf HADR bezogenen Datenbankkonfigurationsparameter bleiben unverändert, sodass sich die Datenbank problemlos als HADR-Datenbank reaktivieren lässt.

Der Befehl **TAKEOVER HADR**, der nur in der Bereitschaftsdatenbank abgesetzt werden kann, ändert die Bereitschaftsdatenbank in eine Primärdatenbank. Wenn Sie die Option BY FORCE nicht angeben, tauschen die Primärdatenbank und die Bereitschaftsdatenbank ihre Rollen. Wenn Sie die Option BY FORCE angeben, wechselt die Bereitschaftsdatenbank einseitig ihre Rolle und wird zur Primärdatenbank. In diesem Fall versucht die Bereitschaftsdatenbank die Transaktionsverarbeitung in der alten Primärdatenbank zu stoppen. Es gibt jedoch keine Garantie, dass die Transaktionsverarbeitung tatsächlich gestoppt wird. Verwenden Sie die Option BY FORCE zur Erzwingung einer Übernahmeoperation nur unter Bedingungen, in denen die Funktionsübernahme erforderlich ist. Stellen Sie so weit wie möglich sicher, dass die aktuelle Primärdatenbank definitiv ausgefallen ist, oder fahren Sie sie selbst herunter, bevor Sie den Befehl **TAKEOVER HADR** mit der Option BY FORCE absetzen.

Rollenwechsel einer HADR-Datenbank

Eine Datenbank kann dynamisch und wiederholt zwischen der Rolle der Primärdatenbank und einer Standarddatenbank umgeschaltet werden. Wenn die Datenbank online oder offline ist, können Sie sowohl den Befehl **START HADR** mit der Option AS PRIMARY als auch den Befehl **STOP HADR** ausführen.

Sie können eine Datenbank zwischen der Rolle als Bereitschaftsdatenbank und der Rolle als Standarddatenbank statisch umschalten. Mehrfach ist dies jedoch nur möglich, wenn die Datenbank im Status *aktualisierende Recovery anstehend* verbleibt. Sie können den Befehl **START HADR** mit der Option AS STANDBY ausführen, um eine Standarddatenbank in eine Bereitschaftsdatenbank umzuwandeln, während die Datenbank offline ist und sich im Status 'aktualisierende Recovery anstehend' befindet. Verwenden Sie den Befehl **STOP HADR**, um eine Bereitschaftsdatenbank in eine Standarddatenbank umzuwandeln, während die Datenbank offline ist. Die Datenbank bleibt nach der Ausführung des Befehls **STOP HADR** im Status 'aktualisierende Recovery anstehend'. Durch einen nachfolgend ausgeführten Befehl **START HADR** mit der Option AS STANDBY wird die Datenbank wieder in eine Bereitschaftsdatenbank geändert. Wenn Sie den Befehl **ROLLFORWARD DATABASE** mit der Option STOP ausführen, nachdem HADR in einer Bereitschaftsdatenbank gestoppt wurde, können Sie die Datenbank nicht wieder zu einer Bereitschaftsdatenbank machen. Da sich die Datenbank nicht mehr im Status *aktualisierende Recovery anstehend* befindet, können Sie sie wieder als Standarddatenbank verwenden. Dies wird als Erfassung einer Momentaufnahme der Bereitschaftsdatenbank bezeichnet. Nach der Umwandlung

einer vorhandenen Bereitschaftsdatenbank in eine Standarddatenbank sollten Sie aus Gründen der hohen Verfügbarkeit in Betracht ziehen, eine neue Bereitschaftsdatenbank zu erstellen.

Zum Vertauschen der Rollen von Primär- und Bereitschaftsdatenbank führen Sie eine TAKEOVER-Operation ohne die Option BY FORCE aus.

Zur einseitigen Änderung der Bereitschaftsdatenbank in eine Primärdatenbank (d. h. ohne die Primärdatenbank zur Bereitschaftsdatenbank zu machen), verwenden Sie den Befehl TAKEOVER mit der Option BY FORCE. Nachfolgend kann die alte Primärdatenbank möglicherweise als neue Bereitschaftsdatenbank wieder integriert werden.

Die HADR-Rolle ist persistent. Wenn eine HADR-Rolle eingerichtet ist, bleibt sie als Eigenschaft der Datenbank bestehen, selbst wenn die DB2-Instanz zwischendurch wiederholt gestoppt und erneut gestartet bzw. die DB2-Datenbank inaktiviert und aktiviert wird.

Starten der Bereitschaftsdatenbank erfolgt asynchron

Wenn Sie den Befehl **START HADR** mit der Option AS STANDBY ausführen, gibt der Befehl die Steuerung zurück, sobald die relevanten EDUs (Engine Dispatchable Units, zuteilbare Einheiten der Steuerkomponente) erfolgreich gestartet wurden. Der Befehl wartet nicht darauf, dass die Bereitschaftsdatenbank eine Verbindung zur Primärdatenbank herstellt. Vielmehr wird die Primärdatenbank erst dann als gestartet betrachtet, wenn sie eine Verbindung zu einer Bereitschaftsdatenbank herstellt (sofern der Befehl **START HADR** in der Primärdatenbank nicht mit der Option BY FORCE ausgeführt wird). Wenn die Bereitschaftsdatenbank einen Fehler feststellt, zum Beispiel wenn die Verbindung von der Primärdatenbank verweigert wird, wurde der Befehl **START HADR** mit der Option AS STANDBY möglicherweise bereits erfolgreich beendet. Infolgedessen ist keine Benutzereingabeaufforderung vorhanden, an die HADR einen Fehleranzeiger zurückmelden könnte. Die HADR-Bereitschaftsdatenbank schreibt in solchen Fällen eine Nachricht in das DB2-Diagnoseprotokoll und fährt sich selbst herunter. Sie sollten den Status der HADR-Bereitschaftsdatenbank überwachen, um sicherzustellen, dass sie erfolgreich eine Verbindung zur HADR-Primärdatenbank herstellt.

Protokollanwendungsfehler, d. h. Fehler, die von der Bereitschaftsdatenbank bei der nachvollziehenden Anwendung von Protokollsätzen festgestellt werden, setzen die Bereitschaftsdatenbank ebenfalls außer Gefecht. Solche Fehler können zum Beispiel auftreten, wenn zur Erstellung eines Pufferpools nicht genügend Speicher verfügbar ist oder wenn bei der Erstellung eines Tabellenbereichs der Pfad nicht gefunden wird. Daher sollten Sie den Status der Bereitschaftsdatenbank kontinuierlich überwachen.

Führen Sie HADR-Befehle nicht über einen Client aus, der mit einem zur Clientweiterleitung eingerichteten Datenbankaliasnamen arbeitet

Wenn die automatische Clientweiterleitung eingerichtet ist, ist für den Datenbankserver ein alternativer Server vordefiniert, sodass Clientanwendungen bei minimaler Unterbrechung zwischen der Arbeit mit dem ursprünglichen Datenbankserver und dem alternativen Server umschalten können. Wenn ein Client in einer solchen Umgebung eine Verbindung über TCP herstellt, kann die tatsächliche Verbindung entweder zur ursprünglichen Datenbank oder zur alternativen Datenbank erfolgen. HADR-Befehle sind so implementiert, dass sie die Zieldatenbank durch die regulä-

re Verbindungslogik des Clients ermitteln. Wenn für die Zieldatenbank eine alternative Datenbank definiert ist, ist es daher schwierig, die Datenbank zu bestimmen, an der der Befehl tatsächlich ausgeführt wird. Während es für einen SQL-Client keinen Unterschied macht, mit welcher Datenbank er verbunden wird, müssen HADR-Befehl auf eine bestimmte Datenbank angewendet werden. Um dieser Einschränkung Rechnung zu tragen, sollten HADR-Befehle lokal auf der Servermaschine ausgeführt werden, sodass die Clientweiterleitung umgangen wird (die Clientweiterleitung betrifft nur TCP/IP-Verbindungen).

HADR-Befehle müssen auf einem Server mit einer gültigen Lizenz ausgeführt werden

Die Befehle **START HADR**, **STOP HADR** und **TAKEOVER HADR** setzen voraus, dass eine gültige HADR-Lizenz auf dem Server installiert ist, auf dem sie ausgeführt werden. Wenn die Lizenz nicht vorhanden ist, schlagen diese Befehle fehl und geben einen befehlsspezifischen Fehlercode (SQL1767N, SQL1769N bzw. SQL1770N) zusammen mit dem Ursachencode 98 zurück. Zur Behebung des Problems müssen Sie entweder eine gültige HADR-Lizenz mithilfe des Befehls **db2licm** installieren oder eine Version des Servers installieren, deren Lieferumfang eine gültige HADR-Lizenz enthält.

Mehrere HADR-Bereitschaftsdatenbanken

Die HADR-Funktion (HADR = High Availability Disaster Recovery) unterstützt die Verwendung mehrerer Bereitschaftsdatenbanken. Dadurch können Sie Ihre Daten an mehr als zwei Standorten speichern und somit einen verbesserten Datenschutz unter Anwendung nur einer Technologie bereitstellen.

Wenn Sie die HADR-Funktion im Modus für mehrere Bereitschaftsdatenbanken implementieren, können Sie bis zu drei Bereitschaftsdatenbanken konfigurieren. Eine dieser Datenbanken muss als *HADR-Hauptbereitschaftsdatenbank* und die anderen Bereitschaftsdatenbanken müssen als *HADR-Nebenbereitschaftsdatenbanken* definiert werden. Beide Typen von HADR-Bereitschaftsdatenbanken werden mit der HADR-Primärdatenbank über eine direkte TCP/IP-Verbindung synchronisiert. Beide Typen unterstützen Leseoperationen in der Bereitschaftsdatenbank und beide Typen können für die zeitverzögerte Wiedergabe von Protokollen konfiguriert werden. Darüber hinaus können Sie auf allen Bereitschaftsdatenbanken eine erzwungene oder nicht erzwungene Übernahme angeben. Allerdings bestehen zwischen der Haupt- und den Nebenbereitschaftsdatenbanken einige wichtige Unterschiede:

- Die automatisierte Funktionsübernahme von IBM Tivoli System Automation for Multiplatforms (SA MP) wird nur in der Hauptbereitschaftsdatenbank unterstützt. Sie müssen eine manuelle Übernahme in einer der Nebenbereitschaftsdatenbanken angeben, um eine dieser Datenbanken als Primärdatenbank zu definieren.
- Alle HADR-Synchronisationsmodi werden in der Hauptbereitschaftsdatenbank unterstützt. Die Nebenbereitschaftsdatenbanken können nur im Modus SUPERASYNC ausgeführt werden.

Es gibt eine Reihe von Vorteilen bei der Verwendung einer Konfiguration mit mehreren HADR-Bereitschaftsdatenbanken. Anstatt die HADR-Funktion anzuwenden, um die Hochverfügbarkeitsanforderungen zu erfüllen und für die Disaster-Recovery auf eine andere Technologie zurückzugreifen, können Sie HADR zur Erfüllung beider Anforderungen verwenden. Die Hauptbereitschaftsdatenbank kann am selben Standort wie die Primärdatenbank implementiert werden. Wenn nun in der Primärdatenbank ein Ausfall auftritt, kann die Hauptbereitschaftsdatenbank inner-

halb der für die Recovery definierten Zeitvorgabe die Rolle der Primärdatenbank übernehmen. Sie können auch Nebenbereitschaftsdatenbanken an einem fernen Standort implementieren. Diese Vorgehensweise bietet Schutz gegen einen flächendeckenden Katastrophenfall, der sowohl die Primärdatenbank als auch die Hauptbereitschaftsdatenbank betrifft. Die Entfernung und die potenziellen Verzögerungen bei der Netzübertragung zwischen der Primärdatenbank und den Nebenbereitschaftsdatenbanken haben keine Auswirkungen auf die Aktivität der Primärdatenbank, da die Nebenbereitschaftsdatenbanken den Modus SUPERASYNC verwenden. Sollten in einem Katastrophenfall die Primärdatenbank und die Hauptbereitschaftsdatenbank betroffen sein, können Sie in einer der Nebenbereitschaftsdatenbanken eine Übernahme angeben. Unter Verwendung des Datenbankkonfigurationsparameters **hadr_target_list** kann die Nebenbereitschaftsdatenbank als neue Hauptbereitschaftsdatenbank konfiguriert werden. Eine Nebenbereitschaftsdatenbank kann jedoch auch dann die Rolle der Primärdatenbank übernehmen, wenn keine verfügbare Bereitschaftsdatenbank vorhanden ist. Wenn es beispielsweise in der Primärdatenbank und in der Hauptbereitschaftsdatenbank zu einem Ausfall kommt, kann eine Nebenbereitschaftsdatenbank die Rolle der Primärdatenbank übernehmen, auch wenn keine entsprechende Bereitschaftsdatenbank verfügbar ist. Wenn Sie diese Datenbank jedoch stoppen, nachdem sie die Rolle der neuen Primärdatenbank übernommen hat, kann sie nur dann erneut als HADR-Primärdatenbank gestartet werden, wenn die zugehörige Hauptbereitschaftsdatenbank ebenfalls gestartet wird.

Einschränkungen für mehrere Bereitschaftsdatenbanken

Es gibt eine Reihe von Einschränkungen, die Sie berücksichtigen sollten, wenn Sie die Implementierung der HADR-Funktion im Modus für mehrere Bereitschaftsdatenbanken planen.

Es bestehen die folgenden Einschränkungen:

- Sie können maximal drei Bereitschaftsdatenbanken einrichten. Hierbei übernimmt eine die Rolle der Hauptbereitschaftsdatenbank und die andere(n) können Sie als Nebenbereitschaftsdatenbank nutzen.
- Nur die Hauptbereitschaftsdatenbank unterstützt alle HADR-Synchronisationsmodi, wohingegen die Nebenbereitschaftsdatenbanken im Modus SUPERASYNC ausgeführt werden.
- Die Unterstützung durch IBM Tivoli System Automation for Multiplatforms (SA MP) hat nur zwischen der HADR-Primärdatenbank und der zugehörigen Hauptbereitschaftsdatenbank Gültigkeit.
- Der Datenbankkonfigurationsparameter **hadr_target_list** muss auf allen Datenbanken innerhalb der Konfiguration mit mehreren Bereitschaftsdatenbanken konfiguriert sein. Jede Bereitschaftsdatenbank muss in der Einstellung für **hadr_target_list** die Primärdatenbank enthalten.

Initialisierung von HADR im Modus für mehrere Bereitschaftsdatenbanken

Die Initialisierung eines HADR-Systems im Modus für mehrere Bereitschaftsdatenbanken ist mit der Vorgehensweise im Modus für eine Bereitschaftsdatenbank vergleichbar. Der Hauptunterschied besteht darin, dass Sie den Modus für mehrere Bereitschaftsdatenbanken aktivieren müssen, indem Sie den Datenbankkonfigurationsparameter **hadr_target_list** für alle Datenbanken in Ihrer Konfiguration setzen.

Informationen zu diesem Vorgang

In dieser Task werden die Schritte beschrieben, die zur Initialisierung von HADR im Modus für mehrere Bereitschaftsdatenbanken erforderlich sind. Wenn Sie eine Konfiguration mit einer Bereitschaftsdatenbank in eine Konfiguration mit mehreren Bereitschaftsdatenbanken ändern möchten, lesen Sie die Informationen in „Aktivieren des Modus für mehrere Bereitschaftsdatenbanken in einer vorhandenen HADR-Konfiguration" auf Seite 224.

Der Modus für mehrere Bereitschaftsdatenbanken setzt voraus, dass der Konfigurationsparameter **hadr_target_list** für alle beteiligten Datenbanken definiert ist. Dieser Parameter listet die Bereitschaftsdatenbanken für den Fall auf, dass die Datenbank die Funktion einer Primärdatenbank übernimmt. Er muss auch auf einer Bereitschaftsdatenbank definiert werden. Gegenseitige Einbeziehung ist erforderlich (d. h., wenn in der Zielliste von Datenbank A Datenbank B enthalten ist, muss in der Zielliste von Datenbank B auch Datenbank A enthalten sein). Dadurch wird sichergestellt, dass die neue Primärdatenbank nach der Übernahme von einer Bereitschaftsdatenbank die alte Primärdatenbank als Bereitschaftsdatenbank behalten kann. Die erste in der Zielliste angegebene Bereitschaftsdatenbank wird als *HADR-Hauptbereitschaftsdatenbank* bezeichnet. Weitere Bereitschaftsdatenbanken sind so genannte *HADR-Nebenbereitschaftsdatenbanken*. Die Zielliste muss nicht immer alle beteiligten Datenbanken enthalten. Ferner besteht bei Vorhandensein mehrerer Bereitschaftsdatenbanken keine Notwendigkeit für Symmetrie oder Gegenseitigkeit. Selbst dann, wenn Sie Datenbank A als Hauptbereitschaftsdatenbank Datenbank B zuordnen, muss Datenbank B nicht unbedingt Datenbank A als Hauptbereitschaftsdatenbank festlegen. Jede in der Zielliste von Datenbank A angegebene Bereitschaftsdatenbank muss wiederum in ihrer Zielliste Datenbank A enthalten. Das Definieren der Ziellisten für die einzelnen Datenbanken ist ein wichtiger Schritt.

Ein Sonderfall ist die Konfiguration des Modus für mehrere Bereitschaftsdatenbanken mit nur einer Bereitschaftsdatenbank. Im Modus für mehrere Bereitschaftsdatenbanken können beispielsweise zwei Datenbanken jeweils als Primärdatenbank und als Bereitschaftsdatenbank konfiguriert werden. Das Verhalten entspricht in diesem Fall nicht der Konfiguration mit einer Bereitschaftsdatenbank, weil Funktionen wie die automatisierte Konfiguration wirksam sind, die nur im Modus für mehrere Bereitschaftsdatenbanken ausgeführt werden, und weil Bereitschaftsdatenbankziele dynamisch hinzugefügt oder entfernt werden können.

Tipp: Sie können die Schritte 2 und 4 auf allen Datenbanken in einem einzigen Aktualisierungsschritt ausführen.

Vorgehensweise

Gehen Sie wie folgt vor, um HADR im Modus für mehrere Bereitschaftsdatenbanken zu initialisieren:

1. Erstellen Sie Ihre Bereitschaftsdatenbank(en) entweder anhand eines wiederhergestellten Backups oder mithilfe einer geteilten Spiegeldatenbank. Anweisungen zur Vorgehensweise finden Sie in „Initialisieren einer Bereitschaftsdatenbank" auf Seite 60 oder in Schritt 2 von „Initialisieren von High Availability Disaster Recovery (HADR)" auf Seite 38.
   - Setzen Sie in der Primärdatenbank den folgenden Befehl ab:

     ```
     BACKUP DB dbname
     ```
   - Setzen Sie in den Bereitschaftsdatenbanken den folgenden Befehl ab:

     ```
     RESTORE DB dbname
     ```

2. Definieren Sie auf allen Datenbanken die Parameter **hadr_local_host**, **hadr_local_svc**, **hadr_local_svc** und **hadr_sync_mode**.

```
"UPDATE DB CFG FOR dbname USING
    HADR_LOCAL_HOST   hostname
    HADR_LOCAL_SVC    servicename
    HADR_SYNCMODE     syncmodus"
```

3. Definieren Sie den Konfigurationsparameter **hadr_target_list** auf allen Bereitschaftsdatenbanken und auf der Primärdatenbank.

```
DB2 UPDATE DB CFG FOR dbname USING
     HADR_TARGET_LIST  haupt_hostname:haupt_servicename|
neben_hostname1:neben_servicename1|neben_hostname2:neben_servicename2
```

4. Definieren Sie auf allen Datenbanken die Konfigurationsparameter **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst**.

   Dieser Schritt ist nicht erforderlich, weil diese Werte im Modus für mehrere Bereitschaftsdatenbanken automatisch definiert bzw. neu definiert werden, wenn Sie diese nicht oder nicht ordnungsgemäß definieren. Wenn Sie diese Parameter jedoch explizit auf die richtigen Werte setzen, können sie umgehend verwendet werden. Diese Werte sind hilfreich für die IBM Tivoli System Automation for Multiplatforms-Software (SA MP), für die möglicherweise der Wert **hadr_remote_inst** zur Erstellung des Ressourcennamens erforderlich ist.

   - Setzen Sie die Parameter in der Primärdatenbank auf die entsprechenden Werte der Hauptbereitschaftsdatenbank, indem Sie den folgenden Befehl absetzen:

     ```
     DB2 "UPDATE DB CFG FOR  dbname USING
          HADR_REMOTE_HOST   haupt_hostname
          HADR_REMOTE_SVC    haupt_servicename
          HADR_REMOTE_INST   haupt_instname"
     ```

   - Setzen Sie die Parameter in jeder Bereitschaftsdatenbank auf die entsprechenden Werte der Primärdatenbank, indem Sie den folgenden Befehl absetzen:

     ```
     DB2 "UPDATE DB CFG FOR  dbname USING
          HADR_REMOTE_HOST   primär_hostname
          HADR_REMOTE_SVC    primär_servicename
          HADR_REMOTE_INST   primär_instname"
     ```

5. Stellen Sie eine Verbindung zu allen Bereitschaftsdatenbankinstanzen her.
6. Setzen Sie in der Bereitschaftsdatenbankinstanz den Befehl **START HADR** mit dem Parameter **AS STANDBY** ab:

   ```
   START HADR ON DB dbname AS STANDBY
   ```

7. Stellen Sie eine Verbindung zur Primärinstanz her.
8. Setzen Sie in der Primärinstanz den Befehl **START HADR** mit dem Parameter **AS PRIMARY** ab:

   ```
   START HADR ON DB dbname AS PRIMARY
   ```

Ergebnisse

Die Bereitschaftsdatenbanken werden im Status 'Lokales Catch-up' gestartet, in dem lokal verfügbare Protokolldateien gelesen und wiedergegeben werden. Nach der Wiedergabe aller Protokolle wechseln die Datenbanken in den Status 'Fernes Catch-up anstehend'. Nach dem Start der Primärdatenbank wechseln die Bereitschaftsdatenbanken in den Status 'Fernes Catch-up', in dem Protokollseiten von der Primärdatenbank empfangen und wiedergegeben werden. Nachdem auf den Bereitschaftsdatenbanken alle Protokolldateien wiedergegeben wurden, die sich auf der Festplatte der Primärdatenbank befinden, hängt die weitere Vorgehensweise vom verwendeten Synchronisationsmodus ab. Eine Hauptbereitschaftsdatenbank im Modus SUPERASYNC und alle Nebenbereitschaftsdatenbanken verbleiben im Modus 'Fernes Catch-up'. Eine Hauptbereitschaftsdatenbank im Modus SYNC, NEARSYNC oder ASYNC wechselt in den Peermodus.

Aktivieren des Modus für mehrere Bereitschaftsdatenbanken in einer vorhandenen HADR-Konfiguration

Die Initialisierung eines HADR-Systems im Modus für mehrere Bereitschaftsdatenbanken ist mit der Vorgehensweise im Modus für eine Bereitschaftsdatenbank vergleichbar. Der Hauptunterschied besteht darin, dass Sie den Modus für mehrere Bereitschaftsdatenbanken aktivieren müssen, indem Sie den Datenbankkonfigurationsparameter **hadr_target_list** für alle Datenbanken in Ihrer Konfiguration setzen.

Vorbereitende Schritte

- Legen Sie den Hostnamen oder die Host-IP-Adresse (für die Einstellung **hadr_local_host**) sowie den Servicenamen oder die Portnummer (für die Einstellung **hadr_local_svc**) für alle beteiligten Datenbanken fest.
- Definieren Sie die Zielliste für die einzelnen Datenbanken.
- Legen Sie den Synchronisationsmodus und das Peerfenster für die Hauptbereitschaftsdatenbanken der einzelnen Datenbanken für den Fall fest, dass die Datenbank die Funktion der Primärdatenbank übernimmt.
- Legen Sie die Einstellung für den Konfigurationsparameter **hadr_timeout** fest; die Einstellung für diesen Parameter muss auf allen Datenbanken gleich sein.
- Stellen Sie fest, ob die Netzbandbreite zwischen der Primärdatenbank und den einzelnen Bereitschaftsdatenbanken ausreichend ist. Führen Sie gegebenenfalls eine Aktualisierung durch.
- Stellen Sie fest, ob die Netzschnittstelle der Primärdatenbank abgehende Datenflüsse der zusätzlichen Bereitschaftsdatenbanken unterstützen kann. Führen Sie gegebenenfalls eine Aktualisierung durch.

Informationen zu diesem Vorgang

Der Modus für mehrere Bereitschaftsdatenbanken setzt voraus, dass der Konfigurationsparameter **hadr_target_list** für alle beteiligten Datenbanken definiert ist. Dieser Parameter listet die Bereitschaftsdatenbanken für den Fall auf, dass die Datenbank die Funktion einer Primärdatenbank übernimmt. Er muss auch auf einer Bereitschaftsdatenbank definiert werden. Gegenseitige Einbeziehung ist erforderlich (d. h., wenn in der Zielliste von Datenbank A Datenbank B enthalten ist, muss in der Zielliste von Datenbank B auch Datenbank A enthalten sein). Dadurch wird sichergestellt, dass die neue Primärdatenbank nach der Übernahme von einer Bereitschaftsdatenbank die alte Primärdatenbank als Bereitschaftsdatenbank behalten kann. Die erste in der Zielliste angegebene Bereitschaftsdatenbank wird als *HADR-Hauptbereitschaftsdatenbank* bezeichnet. Weitere Bereitschaftsdatenbanken sind so genannte *HADR-Nebenbereitschaftsdatenbanken*. Die Zielliste muss nicht immer alle beteiligten Datenbanken enthalten. Ferner besteht bei Vorhandensein mehrerer Bereitschaftsdatenbanken keine Notwendigkeit für Symmetrie oder Gegenseitigkeit. Selbst dann, wenn Sie Datenbank A als Hauptbereitschaftsdatenbank Datenbank B zuordnen, muss Datenbank B nicht unbedingt Datenbank A als Hauptbereitschaftsdatenbank festlegen. Jede in der Zielliste von Datenbank A angegebene Bereitschaftsdatenbank muss wiederum in ihrer Zielliste Datenbank A enthalten. Das Definieren der Ziellisten für die einzelnen Datenbanken ist ein wichtiger Schritt.

Ein Sonderfall ist die Konfiguration des Modus für mehrere Bereitschaftsdatenbanken mit nur einer Bereitschaftsdatenbank. Im Modus für mehrere Bereitschaftsdatenbanken können beispielsweise zwei Datenbanken jeweils als Primärdatenbank und als Bereitschaftsdatenbank konfiguriert werden. Das Verhalten entspricht in diesem Fall nicht der Konfiguration mit einer Bereitschaftsdatenbank, weil Funktio-

nen wie die automatisierte Konfiguration wirksam sind, die nur im Modus für mehrere Bereitschaftsdatenbanken ausgeführt werden, und weil Bereitschaftsdatenbankziele dynamisch hinzugefügt oder entfernt werden können.

In dieser Task werden zuerst nur die neuen Bereitschaftsdatenbanken erstellt und konfiguriert. Da die ursprüngliche Konfiguration bis kurz vor Ausführung der letzten Schritte beibehalten wird, bleibt das Datenbankpaar (Primär- und Bereitschaftsdatenbank) so lange wie möglich funktionsfähig. Wenn Sie die ursprüngliche Konfiguration der Bereitschaftsdatenbank in einem zu frühen Stadium ändern, kann die Verbindung zwischen dem alten HADR-Paar unterbrochen werden, wenn die Bereitschaftsdatenbank zur Übernahme der neuen Konfiguration versehentlich inaktiviert und erneut aktiviert wird.

Vorgehensweise

Gehen Sie wie folgt vor, um HADR im Modus für mehrere Bereitschaftsdatenbanken zu aktivieren:

1. Erstellen Sie die zusätzlich benötigten Bereitschaftsdatenbanken entweder anhand eines wiederhergestellten Backups oder mithilfe einer geteilten Spiegeldatenbank. Anweisungen zur Vorgehensweise finden Sie in „Initialisieren einer Bereitschaftsdatenbank" auf Seite 60 oder in Schritt 2 von „Initialisieren von High Availability Disaster Recovery (HADR)" auf Seite 38.
   - Für die Primärdatenbank:

     ```
     DB2 BACKUP DB dbname
     ```
   - Für die Bereitschaftsdatenbanken:

     ```
     DB2 RESTORE DB dbname
     ```
2. Konfigurieren Sie jede der neuen Bereitschaftsdatenbanken wie folgt:
   a. Geben Sie für **hadr_local_host** und **hadr_local_svc** die von der HADR-Verbindung verwendete TCP-Adresse an.
   b. Definieren Sie die Konfigurationsparameter **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** so, dass sie auf die Primärdatenbank verweisen.
   c. Geben Sie für den Konfigurationsparameter **hadr_timeout** auf allen Datenbanken dieselbe Einstellung an.
   d. Definieren Sie den Konfigurationsparameter **hadr_target_list** wie bereits geplant.
   e. Definieren Sie die Konfigurationsparameter **hadr_syncmode** und **hadr_peer_window** für die Hauptbereitschaftsdatenbank für den Fall, dass diese Datenbank die Funktion der Primärdatenbank übernimmt.
   f. Geben Sie je nach gewünschter Konfiguration weitere HADR-spezifische Parameter wie **hadr_spool_limit** oder **hadr_replay_delay** an.
3. Stellen Sie eine Verbindung zu allen neuen Bereitschaftsdatenbankinstanzen her und setzen Sie den Befehl **START HADR** mit der Option **AS STANDBY** ab.

   ```
   START HADR ON DB dbname AS STANDBY
   ```
4. Rekonfigurieren Sie die ursprüngliche Bereitschaftsdatenbank gemäß den Anweisungen in Schritt 2.
5. Rekonfigurieren Sie die Primärdatenbank wie folgt:
   a. Geben Sie für **hadr_local_host** und **hadr_local_svc** die von der HADR-Verbindung verwendete TCP-Adresse an. Wenn Sie eine neue Netzschnittstellenkarte verwenden, um für die zusätzlichen Bereitschaftsdatenbanken eine höhere Netzbandbreite bereitstellen zu können, müssen Sie möglicherweise eine Aktualisierung durchführen.

b. Definieren Sie die Konfigurationsparameter **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** so, dass sie auf die Hauptbereitschaftsdatenbank verweisen.
c. Geben Sie für den Konfigurationsparameter **hadr_timeout** auf allen Bereitschaftsdatenbanken dieselbe Einstellung an.
d. Definieren Sie den Konfigurationsparameter **hadr_target_list** wie bereits geplant.
e. Definieren Sie die Konfigurationsparameter **hadr_syncmode** und **hadr_peer_window**, die von der Hauptbereitschaftsdatenbank verwendet werden.
f. Geben Sie je nach gewünschter Konfiguration weitere HADR-spezifische Parameter wie **hadr_spool_limit** oder **hadr_replay_delay** an.

6. Inaktivieren Sie die ursprüngliche Bereitschaftsdatenbank und aktivieren Sie diese erneut, um die neue Konfiguration zu übernehmen.
7. Stoppen Sie HADR auf der Primärdatenbank und führen Sie dann einen Neustart durch, um die neue Konfiguration zu übernehmen.

Ergebnisse

Alle Bereitschaftsdatenbanken sollten innerhalb von Sekunden eine Verbindung zur Primärdatenbank herstellen. Sie können den Status der Datenbanken überwachen, indem Sie den Befehl **db2pd** mit der Option `-hadr` oder die Tabellenfunktion MON_GET_HADR verwenden.

Änderungen an einer Konfiguration mit mehreren Bereitschaftsdatenbanken

Nachdem Ihre Konfiguration mit mehreren HADR-Bereitschaftsdatenbanken betriebsbereit ist, möchten Sie möglicherweise weitere Änderungen (z. B. Hinzufügen oder Entfernen von Nebenbereitschaftsdatenbanken oder Ändern der Hauptbereitschaftsdatenbank) durchführen. Sie können diese Änderungen durchführen, ohne dass es zu einem Ausfall der Primärdatenbank kommt.

Hinzufügen von Nebenbereitschaftsdatenbanken

Unter bestimmten Umständen kann das Hinzufügen einer Nebenbereitschaftsdatenbank erforderlich sein:

- Sie möchten eine zusätzliche Bereitschaftsdatenbank zur Verarbeitung von schreibgeschützten Workloads implementieren.
- Sie möchten eine zusätzliche Bereitschaftsdatenbank für die zeitverzögerte Wiedergabe implementieren.
- Sie möchten eine zusätzliche Bereitschaftsdatenbank zu Disaster-Recovery-Zwecken implementieren.
- Sie möchten eine Bereitschaftsdatenbank hinzufügen, die zu einer zuvor aktiven HADR-Implementierung gehörte, dann jedoch *verwaist* war, weil die Bereitschaftsdatenbank im Konfigurationsparameter **hadr_target_list** der neuen Primärdatenbank nicht aufgeführt ist.

Sie können eine Nebenbereitschaftsdatenbank hinzufügen, wenn Ihre HADR-Implementierung im Modus für mehrere Bereitschaftsdatenbanken arbeitet. Dies bedeutet, dass der Konfigurationsparameter **hadr_target_list** in mindestens einer Bereitschaftsdatenbank bereits definiert sein muss.

Zum Hinzufügen einer Nebenbereitschaftsdatenbank zu Ihrer HADR-Implementierung müssen Sie die Zielliste der Primärdatenbank mit den Host- und Portangaben

der Bereitschaftsdatenbank aktualisieren. Diese Angaben entsprechen den Einstellungen für die Parameter **hadr_local_host** und **hadr_local_svc** in der Bereitschaftsdatenbank. Außerdem müssen Sie die Host- und Portangaben für die Primärdatenbank zur Zielliste der neuen Bereitschaftsdatenbank hinzufügen.

Tipp: Obwohl es nicht erforderlich ist, hat es sich bewährt, die Host- und Portangaben für die neue Bereitschaftsdatenbank auch zur Zielliste der anderen Bereitschaftsdatenbanken in der Implementierung hinzuzufügen. Ebenso sollten Sie die Host- und Portangaben für diese Bereitschaftsdatenbanken auch in der Zielliste der neuen Bereitschaftsdatenbank angeben. Wenn Sie diese zusätzlichen Aktualisierungen nicht durchführen und eine der anderen Bereitschaftsdatenbanken die Funktion der neuen Primärdatenbank übernimmt, dann wird die neue Bereitschaftsdatenbank als Bereitschaftsdatenbankziel zurückgewiesen und heruntergefahren.

Entfernen von Nebenbereitschaftsdatenbanken

Es können nur Nebenbereitschaftsdatenbanken dynamisch entfernt werden. Wenn Sie eine Nebenbereitschaftsdatenbank aus einer Implementierung mit mehreren Bereitschaftsdatenbanken dynamisch entfernen, hat dies keine Auswirkungen auf den normalen HADR-Betrieb in der Primärdatenbank und der Hauptbereitschaftsdatenbank. Zum Entfernen einer Nebenbereitschaftsdatenbank müssen Sie den Befehl **STOP HADR** in der Bereitschaftsdatenbank absetzen. Anschließend können Sie die Datenbank aus der Zielliste der Primärdatenbank und aller anderen Bereitschaftsdatenbanken entfernen.

Ändern der Hauptbereitschaftsdatenbank

Sie können die Hauptbereitschaftsdatenbank nur ändern, wenn Sie zuerst HADR in der Primärdatenbank stoppen. Dies führt allerdings nicht zu einem Ausfall, da die Primärdatenbank nicht inaktiviert werden muss.

Zur Änderung der Hauptbereitschaftsdatenbank müssen Sie HADR in der Primärdatenbank stoppen. Dann aktualisieren Sie die Zielliste der Primärdatenbank, damit die neue Hauptbereitschaftsdatenbank als erstes aufgeführt wird. Wenn die neue Hauptbereitschaftsdatenbank noch keine Bereitschaftsdatenbank ist, fügen Sie die Adresse der Primärdatenbank zur Zielliste hinzu, konfigurieren die anderen HADR-Parameter und aktivieren die Bereitschaftsdatenbank. Wenn diese bereits eine Bereitschaftsdatenbank ist, sind keine Maßnahmen erforderlich.

Tipp: Obwohl es nicht erforderlich ist, hat es sich bewährt, die Host- und Portangaben für die neue Hauptbereitschaftsdatenbank auch zur Zielliste der anderen Bereitschaftsdatenbank in der Implementierung hinzuzufügen. Ebenso sollten Sie die Host- und Portangaben für diese Bereitschaftsdatenbank auch in der Zielliste der neuen Hauptbereitschaftsdatenbank angeben. Wenn Sie diese zusätzlichen Aktualisierungen nicht durchführen und eine der Bereitschaftsdatenbanken die Funktion der neuen Primärdatenbank übernimmt, dann wird die andere Bereitschaftsdatenbank als Bereitschaftsdatenbankziel zurückgewiesen und heruntergefahren.

Datenbankkonfiguration für mehrere HADR-Bereitschaftsdatenbanken

Beachten Sie bei einer Datenbankkonfiguration in einer Konfiguration mit mehreren HADR-Bereitschaftsdatenbanken die folgenden Punkte.

Automatische Rekonfiguration von HADR-Parametern

Rekonfiguration nach dem Starten von HADR

Im Modus für mehrere Bereitschaftsdatenbanken werden die Konfigurationsparameter, die die Primärdatenbank für die Bereitschaftsdatenbanken und die Hauptbereitschaftsdatenbank für die Primärdatenbank identifizieren, automatisch rekonfiguriert, wenn HADR gestartet wird, falls sie nicht korrekt definiert sind. Dieses Verhalten gilt für die folgenden Konfigurationsparameter:

- **hadr_remote_host**
- **hadr_remote_inst**
- **hadr_remote_svc**

Tipp: Auch wenn diese automatische Rekonfiguration auftritt, sollten Sie immer versuchen, die korrekten Anfangswerte zu definieren, weil diese Rekonfiguration möglicherweise nicht wirksam wird, bis eine Verbindung zwischen einer Bereitschaftsdatenbank und der zugehörigen Primärdatenbank hergestellt wird. In bestimmten HADR-Implementierungen sind diese Anfangswerte möglicherweise erforderlich. Wenn Sie beispielsweise die IBM Tivoli System Automation for Multiplatforms-Software verwenden, dann ist der Wert für den Konfigurationsparameter **hadr_remote_inst** erforderlich, um einen Ressourcennamen zu erstellen.

Anmerkung: Wenn die Registrierdatenbankvariable **DB2_HADR_NO_IP_CHECK** auf ON gesetzt ist, werden **hadr_remote_host** und **hadr_remote_svc** nicht automatisch aktualisiert.

Bei der Rekonfiguration wird davon ausgegangen, dass die Werte des Konfigurationsparameters **hadr_target_list** korrekt sind. Wenn ein Zielisteneintrag Fehler enthält, dann müssen Sie ihn manuell korrigieren.

In der Primärdatenbank wird die Rekonfiguration folgendermaßen ausgeführt:

- Wenn die Werte für die Konfigurationsparameter **hadr_remote_host** und **hadr_remote_svc** nicht mit dem *host*:*port*-Paar übereinstimmen, das zuerst im Konfigurationsparameter **hadr_target_list** angegeben wurde (Hauptbereitschaftsdatenbank), dann werden die Konfigurationsparameter **hadr_remote_host** und **hadr_remote_svc** mit den Werten aus der Zielliste aktualisiert.
- Wenn der Wert für den Konfigurationsparameter **hadr_remote_inst** nicht mit dem Instanznamen der Hauptbereitschaftsdatenbank übereinstimmt, wird der korrekte Instanzname in den Konfigurationsparameter **hadr_remote_inst** der Primärdatenbank kopiert, nachdem die Hauptbereitschaftsdatenbank eine Verbindung zur Primärdatenbank hergestellt hat.

Auf einer Bereitschaftsdatenbank wird die Rekonfiguration folgendermaßen ausgeführt:

- Wenn die Bereitschaftsdatenbank gestartet wird, dann versucht sie, eine Verbindung zu der Datenbank herzustellen, die in ihren Konfigurationsparametern **hadr_remote_host**, **hadr_remote_inst** und **hadr_remote_svc** angegeben ist.
- Wenn die Bereitschaftsdatenbank keine Verbindung zur Primärdatenbank herstellen kann, dann wartet sie darauf, dass die Primärdatenbank eine Verbindung herstellt.
- Die Primärdatenbank versucht, anhand der im Parameter **hadr_target_list** aufgelisteten Adressen eine Verbindung zu den Bereitschaftsdatenbanken herzustellen. Nachdem die Primärdatenbank eine

Verbindung zur Bereitschaftsdatenbank hergestellt hat, werden die Konfigurationsparameter **hadr_remote_host**, **hadr_remote_inst** und **hadr_remote_svc** der Bereitschaftsdatenbank mit den korrekten Werten für die Primärdatenbank aktualisiert.

Rekonfiguration während und nach der Übernahme

Bei einer nicht erzwungenen Übernahme werden die Werte für die Konfigurationsparameter **hadr_remote_host**, **hadr_remote_inst** und **hadr_remote_svc** in der neuen Primärdatenbank automatisch für die Hauptbereitschaftsdatenbank aktualisiert. Die Parameter in den Bereitschaftsdatenbanken, die in **hadr_target_list** der neuen Primärdatenbank aufgelistet sind, werden automatisch so aktualisiert, dass sie auf die neue Primärdatenbank verweisen. Datenbanken, die nicht in **hadr_target_list** der neuen Primärdatenbank aufgelistet sind, werden nicht aktualisiert. Diese Datenbanken versuchen weiterhin, eine Verbindung zur alten Primärdatenbank herzustellen, und werden zurückgewiesen, weil die alte Primärdatenbank nun eine Bereitschaftsdatenbank ist. Die alte Primärdatenbank befindet sich aufgrund der Anforderung der gegenseitigen Einbeziehung in die Zielliste auf jeden Fall in der Zielliste der neuen Primärdatenbank.

Bei einer erzwungenen Übernahme funktioniert die automatische Aktualisierung in der neuen Primärdatenbank und in den zugehörigen Bereitschaftsdatenbanken (mit Ausnahme der alten Primärdatenbank) wie bei einer nicht erzwungenen Übernahme. Die automatische Aktualisierung findet in der alten Primärdatenbank jedoch erst statt, nachdem diese beendet und zur Reintegration als Bereitschaftsdatenbank erneut gestartet wurde.

Alle Datenbanken, die während der Übernahme nicht online sind, werden nach ihrem Start automatisch rekonfiguriert. Die automatische Rekonfiguration wird möglicherweise nicht sofort beim Start wirksam, da sie von der regelmäßigen Kontaktaufnahme der Primärdatenbank zur Bereitschaftsdatenbank abhängt. Beim Start kann es vorkommen, dass eine Bereitschaftsdatenbank eine Verbindung zur alten Primärdatenbank herstellen möchte und dem Protokolldatenstrom der alten Primärdatenbank folgt, was zu einer Abweichung vom Protokolldatenstrom der neuen Primärdatenbank führt und verhindert, dass die Bereitschaftsdatenbank mit der neuen Primärdatenbank ein Paar bilden kann. Aus diesem Grund muss die alte Primärdatenbank vor der Übernahme heruntergefahren werden, um dieses sogenannte *Split-Brain-Szenario* zu verhindern.

Bereitschaftsdatenbanken steuern ihren Synchronisationsmodus und ihr Peerfenster nicht

Im Modus für mehrere Bereitschaftsdatenbanken sind ausschließlich die Einstellungen der aktuellen Primärdatenbank für die Konfigurationsparameter **hadr_syncmode** und **hadr_peer_window** relevant. Die Bereitschaftsdatenbanken arbeiten entweder mit den Parametereinstellungen der Primärdatenbank (im Falle der Hauptbereitschaftsdatenbank) oder übernehmen die Werte, die durch ihre Rolle als Nebenbereitschaftsdatenbank vorgegeben sind.

Synchronisationsmodus

Im Modus für mehrere Bereitschaftsdatenbanken muss die Einstellung für den Konfigurationsparameter **hadr_syncmode** in der Primärdatenbank nicht unbedingt mit der entsprechenden Einstellung der Bereitschaftsdatenbanken übereinstimmen. Die Einstellung für den Konfigurationsparameter **hadr_syncmode**, die auf einer Bereitschaftsdatenbank angegeben ist, wird als ihr *konfigurierter Synchronisationsmodus* betrachtet. Diese Einstellung ist aus-

schließlich dann relevant, wenn die Bereitschaftsdatenbank die Rolle einer Primärdatenbank übernimmt. Der Bereitschaftsdatenbank wird ein *effektiver Synchronisationsmodus* zugeordnet. Für alle Nebenbereitschaftsdatenbanken lautet der effektive Synchronisationsmodus immer SUPERASYNC. Für die Hauptbereitschaftsdatenbank entspricht der effektive Synchronisationsmodus der Einstellung für den Konfigurationsparameter **hadr_syncmode** der Primärdatenbank. Bei einer Bereitschaftsdatenbank zeigen die Überwachungsschnittstellen den effektiven Synchronisationsmodus als Synchronisationsmodus an.

Peerfenster
Im Modus für mehrere Bereitschaftsdatenbanken muss die Einstellung für den Konfigurationsparameter **hadr_peer_window** in der Primärdatenbank nicht unbedingt mit der entsprechenden Einstellung der Bereitschaftsdatenbanken übereinstimmen. Tatsächlich werden alle Einstellungen für den Konfigurationsparameter **hadr_peer_window** in den Nebenbereitschaftsdatenbanken ignoriert, weil die Peerfensterfunktionalität nicht mit dem Modus SUPERASYNC kompatibel ist. Die Hauptbereitschaftsdatenbank verwendet die Peerfenstereinstellung der Primärdatenbank, die nur dann anwendbar ist, wenn der Wert des Konfigurationsparameters **hadr_syncmode** der Bereitschaftsdatenbank SYNC oder NEARSYNC lautet (wie beim Modus für eine Bereitschaftsdatenbank).

Schrittweise Upgrades im Modus für mehrere HADR-Bereitschaftsdatenbanken

Wie im Modus für eine HADR-Bereitschaftsdatenbank kann auch in diesem Modus ein schrittweises Upgrade durchgeführt werden. Der wichtigste Unterschied besteht darin, dass Sie diese Prozedur bei mehreren Bereitschaftsdatenbanken unter Aufrechterhaltung des HADR-Schutzes durchführen können, indem eine Primärdatenbank und eine Bereitschaftsdatenbank aktiv bleiben.

Es ist immer eine Primärdatenbank vorhanden, die Datenbankservices bereitstellt, und diese Primärdatenbank verfügt immer über mindestens eine Bereitschaftsdatenbank, die HA- und DR-Schutz gewährleistet.

Bei mehreren Bereitschaftsdatenbanken sollten Sie die Aktualisierung oder das Upgrade zuerst auf allen Bereitschaftsdatenbanken und dann auf der Primärdatenbank durchführen. Dies ist insbesondere bei einer Aktualisierung der Fixpackstufe wichtig, da HADR nicht zulässt, dass die Primärdatenbank eine höhere Fixpackstufe als die Bereitschaftsdatenbanken aufweist.

Diese Prozedur entspricht weitgehend der Prozedur im Modus für eine Bereitschaftsdatenbank, mit der Ausnahme, dass das Upgrade für jede Datenbank einzeln durchgeführt und zuerst mit einer Nebenbereitschaftsdatenbank begonnen werden sollte. Nehmen wir beispielsweise die folgende HADR-Konfiguration:

- host1 ist die Primärdatenbank
- host2 ist die Hauptbereitschaftsdatenbank
- host3 ist die Nebenbereitschaftsdatenbank

Führen Sie für diese Konfiguration nacheinander die folgenden Schritte aus, um ein schrittweises Upgrade bzw. eine schrittweise Aktualisierung durchzuführen:

1. Inaktivieren Sie host3, nehmen Sie die erforderlichen Änderungen vor, aktivieren Sie host3 und starten Sie HADR auf host3 (als Bereitschaftsdatenbank).

2. Wenn host3 mit der Protokollwiedergabe beschäftigt ist, inaktivieren Sie host2, nehmen Sie die erforderlichen Änderungen vor, aktivieren Sie host2 und starten Sie HADR auf host2 (als Bereitschaftsdatenbank).
3. Wenn host2 mit der Protokollwiedergabe beschäftigt ist und sich mit host1 im Peerstatus befindet, geben Sie eine Übernahme auf host2 an.
4. Inaktivieren Sie host1, nehmen Sie die erforderlichen Änderungen vor, aktivieren Sie host1 und starten Sie HADR auf host1 (als Bereitschaftsdatenbank).
5. Wenn sich host1 im Peerstatus mit host2 befindet, geben Sie eine Übernahme auf host1 an, damit dieser wieder die Rolle der Primärdatenbank und host2 wieder die Rolle der Hauptbereitschaftsdatenbank übernimmt.

HADR-Überwachung (High Availability Disaster Recovery) im Modus für mehrere Bereitschaftsdatenbanken

Der Modus für mehrere HADR-Bereitschaftsdatenbanken unterstützt dieselben Überwachungsschnittstellen wie der Modus für eine Bereitschaftsdatenbank. Sie sollten jedoch nur den Befehl **db2pd** und die Tabellenfunktion MON_GET_HADR verwenden, weil andere Überwachungsschnittstellen keine vollständige Sicht aller Bereitschaftsdatenbanken liefern.

Die von der Überwachungsschnittstelle zurückgegebenen Informationen hängen davon ab, wo diese Informationen ausgegeben werden. Die Überwachung auf einer Bereitschaftsdatenbank gibt nur Informationen zu dieser Bereitschaftsdatenbank und zur Primärdatenbank zurück. Zu anderen Bereitschaftsdatenbanken werden keine Informationen zurückgegeben. Die Überwachung auf der Primärdatenbank gibt bei Verwendung des Befehls **db2pd** oder der Tabellenfunktion MON_GET_HADR Informationen zu allen Bereitschaftsdatenbanken zurück. Sogar Bereitschaftsdatenbanken, die nicht verbunden, aber im Konfigurationsparameter **hadr_target_list** der Primärdatenbank konfiguriert sind, werden angezeigt. Andere Schnittstellen wie der Befehl **GET SNAPSHOT FOR DATABASE** geben nur Informationen zur Primärdatenbank und zur Hauptbereitschaftsdatenbank zurück.

Der Befehl **db2pd** und die Tabellenfunktion MON_GET_HADR geben im Grunde dieselben Informationen zurück, für den Befehl **db2pd** muss die Funktion für Leseoperationen in der Bereitschaftsdatenbank (für Informationen aus einer Bereitschaftsdatenbank) jedoch nicht aktiviert werden. Darüber hinaus ist der Befehl **db2pd** bei Übernahmen die beste Wahl, da ein Zeitfenster vorhanden sein könnte, in dem weder die Primärdatenbank noch die Bereitschaftsdatenbank Clientverbindungen zulässt.

Befehl db2pd

Im folgenden Beispiel setzt der Datenbankadministrator den Befehl **db2pd** auf einer Primärdatenbank mit drei Bereitschaftsdatenbanken ab. Es werden drei Gruppen von Daten zurückgegeben, wobei jede Gruppe einen Protokollübertragungskanal zwischen Primär- und Bereitschaftsdatenbank darstellt. Das Feld HADR_ROLE gibt die Rolle der Datenbank an, für die der Befehl **db2pd** abgesetzt wird, und enthält daher für alle drei Gruppen den Eintrag PRIMARY. Der Wert von HADR_STATE lautet für die beiden Nebenbereitschaftsdatenbanken (hostS2 und hostS3) REMOTE_CATCHUP, weil diese Datenbanken unabhängig von der konfigurierten Einstellung für **hadr_syncmode** automatisch im Modus SUPERASYNC ausgeführt werden (was auch der Ausgabe von **db2pd** zu entnehmen ist). Der Wert für STANDBY_ID unterscheidet die Bereitschaftsdatenbanken. Er wird vom System generiert und die Zuordnung von ID zu Bereitschaftsdatenbank kann je nach Abfrage unterschiedlich sein; der Hauptbereitschaftsdatenbank wird jedoch immer die ID '1' zugewiesen.

Anmerkung: Für den aktuellen Status nicht relevante Felder sind in der Ausgabe möglicherweise nicht enthalten. In der folgenden Ausgabe sind beispielsweise keine Informationen zum Zeitfenster für das Anwenden von Protokollen (z. B. Startzeit und Transaktionsanzahl) enthalten, da dieser Modus nicht aktiv ist.

```
db2pd -db hadr_db -hadr

  Database Member 0 -- Database hadr_db -- Active -- Up 0 days 00:23:17 -- Date 06/08/2011 13:57:23

                              HADR_ROLE = PRIMARY
                            REPLAY_TYPE = PHYSICAL
                          HADR_SYNCMODE = SYNC
                             STANDBY_ID = 1
                          LOG_STREAM_ID = 0
                             HADR_STATE = PEER
                      PRIMARY_MEMBER_HOST = hostP.ibm.com
                         PRIMARY_INSTANCE = db2inst1
                            PRIMARY_MEMBER = 0
                      STANDBY_MEMBER_HOST = hostS1.ibm.com
                         STANDBY_INSTANCE = db2inst2
                            STANDBY_MEMBER = 0
                      HADR_CONNECT_STATUS = CONNECTED
                 HADR_CONNECT_STATUS_TIME = 06/08/2011 13:38:10.199479 (1307565490)
            HEARTBEAT_INTERVAL(seconds) = 30
                  HADR_TIMEOUT(seconds) = 120
          TIME_SINCE_LAST_RECV(seconds) = 3
               PEER_WAIT_LIMIT(seconds) = 0
             LOG_HADR_WAIT_CUR(seconds) = 0.000
      LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
     LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                    LOG_HADR_WAIT_COUNT = 82
  SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 50772
  SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87616
              PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
              STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                    HADR_LOG_GAP(bytes) = 0
       STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
         STANDBY_RECV_REPLAY_GAP(bytes) = 0
                      PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                      STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
           STANDBY_RECV_BUF_SIZE(pages) = 16
               STANDBY_RECV_BUF_PERCENT = 0
             STANDBY_SPOOL_LIMIT(pages) = 0
                  PEER_WINDOW(seconds) = 0
               READS_ON_STANDBY_ENABLED = Y
      STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N
                              HADR_ROLE = PRIMARY
                            REPLAY_TYPE = PHYSICAL
                          HADR_SYNCMODE = SUPERASYNC
                             STANDBY_ID = 2
                          LOG_STREAM_ID = 0
                             HADR_STATE = REMOTE_CATCHUP
                      PRIMARY_MEMBER_HOST = hostP.ibm.com
                         PRIMARY_INSTANCE = db2inst1
                            PRIMARY_MEMBER = 0
                      STANDBY_MEMBER_HOST = hostS2.ibm.com
                         STANDBY_INSTANCE = db2ins3t
                            STANDBY_MEMBER = 0
                      HADR_CONNECT_STATUS = CONNECTED
                 HADR_CONNECT_STATUS_TIME = 06/08/2011 13:35:51.724447 (1307565351)
            HEARTBEAT_INTERVAL(seconds) = 30
                  HADR_TIMEOUT(seconds) = 120
          TIME_SINCE_LAST_RECV(seconds) = 16
               PEER_WAIT_LIMIT(seconds) = 0
             LOG_HADR_WAIT_CUR(seconds) = 0.000
      LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
     LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                    LOG_HADR_WAIT_COUNT = 82
  SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
  SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
```

```
            PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
            STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                  HADR_LOG_GAP(bytes) = 0
      STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
        STANDBY_RECV_REPLAY_GAP(bytes) = 0
                     PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                     STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
               STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
          STANDBY_RECV_BUF_SIZE(pages) = 16
              STANDBY_RECV_BUF_PERCENT = 0
            STANDBY_SPOOL_LIMIT(pages) = 0
                  PEER_WINDOW(seconds) = 0
            READS_ON_STANDBY_ENABLED = Y

                               HADR_ROLE = PRIMARY
                             REPLAY_TYPE = PHYSICAL
                          HADR_SYNCMODE = SUPERASYNC
                             STANDBY_ID = 3
                          LOG_STREAM_ID = 0
                             HADR_STATE = REMOTE_CATCHUP
                     PRIMARY_MEMBER_HOST = hostP.ibm.com
                        PRIMARY_INSTANCE = db2inst1
                          PRIMARY_MEMBER = 0
                     STANDBY_MEMBER_HOST = hostS3.ibm.com
                        STANDBY_INSTANCE = db2inst3
                          STANDBY_MEMBER = 0
                     HADR_CONNECT_STATUS = CONNECTED
                HADR_CONNECT_STATUS_TIME = 06/08/2011 13:46:51.561873 (1307566011)
             HEARTBEAT_INTERVAL(seconds) = 30
                   HADR_TIMEOUT(seconds) = 120
           TIME_SINCE_LAST_RECV(seconds) = 6
                PEER_WAIT_LIMIT(seconds) = 0
              LOG_HADR_WAIT_CUR(seconds) = 0.000
     LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
    LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                    LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
            PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
            STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                  HADR_LOG_GAP(bytes) = 0
      STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
        STANDBY_RECV_REPLAY_GAP(bytes) = 0
                     PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                     STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
               STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
          STANDBY_RECV_BUF_SIZE(pages) = 16
              STANDBY_RECV_BUF_PERCENT = 0
            STANDBY_SPOOL_LIMIT(pages) = 0
                  PEER_WINDOW(seconds) = 0
            READS_ON_STANDBY_ENABLED = N
```

Tabellenfunktion MON_GET_HADR

Im folgenden Beispiel ruft der Datenbankadministrator die Tabellenfunktion **MON_GET_HADR** auf einer Primärdatenbank mit drei Bereitschaftsdatenbanken auf. Es werden drei Zeilen zurückgegeben. Jede Zeile stellt einen Protokollübertragungskanal zwischen Primär- und Bereitschaftsdatenbank dar. Die Spalte HADR_ROLE gibt die Rolle der Datenbank an, für die die Abfrage ausgegeben wird. Deshalb lautet der Eintrag für dieses Feld in allen drei Zeilen PRIMARY. Der Wert von HADR_STATE lautet für die beiden Nebenbereitschaftsdatenbanken (hostS2 und hostS3) REMOTE_CATCHUP, weil diese Datenbanken unabhängig von der konfigurierten Einstellung für **hadr_syncmode** automatisch im Modus SUPERASYNC ausgeführt werden.

```
db2 "select HADR_ROLE, STANDBY_ID, HADR_STATE, varchar(PRIMARY_MEMBER_HOST,20)
as PRIMARY_MEMBER_HOST, varchar(STANDBY_MEMBER_HOST,20) as STANDBY_MEMBER_HOST from table (mon_get
```

```
HADR_ROLE  STANDBY_ID HADR_STATE      PRIMARY_MEMBER_HOST  STANDBY_MEMBER_HOST
---------  ---------- -------------  -------------------  --------------------
PRIMARY             1 PEER           hostP.ibm.com        hostS1.ibm.com
PRIMARY             2 REMOTE_CATCHUP hostP.ibm.com        hostS2.ibm.com
PRIMARY             3 REMOTE_CATCHUP hostP.ibm.com        hostS3.ibm.com

3 record(s) selected.
```

Übernahme im Modus für mehrere HADR-Bereitschaftsdatenbanken

Wenn eine HADR-Bereitschaftsdatenbank in einer Umgebung mit mehreren Bereitschaftsdatenbanken die Rolle der Primärdatenbank übernimmt, gibt es im Vergleich zum Modus für eine Bereitschaftsdatenbank einige wichtige Unterschiede.

Bei HADR gibt es zwei Typen von Übernahmen: *Rollenwechsel* und *Funktionsübernahme*. Der Rollenwechsel, der manchmal auch als normale oder nicht erzwungene Übernahme bezeichnet wird, kann nur dann erfolgen, wenn die Primärdatenbank verfügbar ist und diese mit einer Bereitschaftsdatenbank die Rollen tauscht. Eine Funktionsübernahme oder eine erzwungene Übernahme kann auch dann erfolgen, wenn die Primärdatenbank nicht verfügbar ist. Diese Art der Übernahme ist bei Ausfällen der Primärdatenbank üblich, um die Bereitschaftsdatenbank als neue Primärdatenbank festzulegen. Bei einer erzwungenen Übernahme verbleibt die alte Primärdatenbank in der primären Rolle. Beide Typen von Übernahmen werden im Modus für mehrere Bereitschaftsdatenbanken unterstützt und jede Bereitschaftsdatenbank kann die Funktion der Primärdatenbank übernehmen. In diesem Zusammenhang muss allerdings unbedingt beachtet werden, dass eine Bereitschaftsdatenbank, die nicht in der Zielliste der neuen Primärdatenbank enthalten ist, als verwaist gilt und keine Verbindung zur neuen Primärdatenbank herstellen kann.

Bei einer Übernahme nimmt DB2 automatisch eine Reihe von Konfigurationsänderungen vor, damit die in der Zielliste der neuen Primärdatenbank aufgeführten Bereitschaftsdatenbanken eine Verbindung zur neuen Primärdatenbank herstellen können. Die Konfigurationsparameter **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** werden in der neuen Primärdatenbank aktualisiert und die Bereitschaftsdatenbanken werden wie folgt aufgelistet:

- **In der neuen Primärdatenbank:** Sie beziehen sich auf die Hauptbereitschaftsdatenbank (die in der Zielliste der neuen Primärdatenbank als erstes aufgelistete Datenbank).
- **In den Bereitschaftsdatenbanken:** Sie beziehen sich auf die neue Primärdatenbank. Wenn eine alte Primärdatenbank reintegriert wird, um die Funktion einer Bereitschaftsdatenbank zu übernehmen, wird diese Datenbank mit dem Befehl **START HADR** AS STANDBY zuerst in eine Bereitschaftsdatenbank konvertiert. Sie kann daher auch automatisch auf die neue Primärdatenbank umgeleitet werden, wenn sie in der Zielliste der neuen Primärdatenbank enthalten ist.

  Anmerkung: Verwaiste Bereitschaftsdatenbanken werden auf diese Weise nicht automatisch aktualisiert. Wenn diese Datenbanken als Bereitschaftsdatenbanken hinzugefügt werden sollen, müssen Sie sicherstellen, dass sie in der Zielliste der neuen Primärdatenbank enthalten sind und die neue Primärdatenbank in den Ziellisten dieser Datenbanken enthalten ist.

Rollenwechsel

Wie im Modus für eine Bereitschaftsdatenbank wird auch im Modus für mehrere Bereitschaftsdatenbanken sichergestellt, dass bei einem Rollenwechsel keine Daten zwischen der alten und der neuen Primärdatenbank verloren gehen. Andere im Konfigurationsparameter **hadr_target_list** der

neuen Primärdatenbank konfigurierte Bereitschaftsdatenbanken werden automatisch auf die neue Datenbank umgeleitet und empfangen weiterhin Protokolle.

Funktionsübernahme

Wenn eine Funktionsübernahme im Modus für mehrere Bereitschaftsdatenbanken Datenverluste zur Folge hat (die neue Primärdatenbank verfügt nicht über alle Daten der alten Primärdatenbank), weichen die Protokolldatenströme der alten und der neuen Primärdatenbank wie im Modus für eine Bereitschaftsdatenbank voneinander ab und die alte Primärdatenbank muss reinitialisiert werden. Wenn eine der anderen Bereitschaftsdatenbanken über den Abweichungspunkt hinaus Protokolle von der alten Primärdatenbank empfangen hat, muss diese reinitialisiert werden. Ansonsten kann sie eine Verbindung zur neuen Primärdatenbank herstellen und die Protokollübertragung und -wiedergabe fortsetzen. Deshalb ist es äußerst wichtig, die Protokollpositionen aller Bereitschaftsdatenbanken zu prüfen und die Bereitschaftsdatenbank mit den meisten Daten als Funktionsübernahmeziel festzulegen. Diese Informationen können mit dem Befehl **db2pd** oder der Tabellenfunktion MON_GET_HADR abgefragt werden.

Anmerkung: Die erfolgreiche automatische Rekonfiguration der Konfigurationsparameter **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** einer Bereitschaftsdatenbank dahingehend, dass sie auf die neue Primärdatenbank verweisen, bedeutet nicht, dass die Bereitschaftsdatenbank als Partner der neuen Primärdatenbank akzeptiert wird. Es bedeutet lediglich, dass die Bereitschaftsdatenbank eine TCP-Verbindung zur Primärdatenbank herstellen kann. Wenn DB2 beim Verbindungsaufbau feststellt, dass die beiden Datenbanken über abweichende Protokolldatenströme verfügen, wird die Paarungsanforderung abgelehnt und die Verbindung geschlossen.

Szenario: Implementieren einer Konfiguration mit mehreren HADR-Bereitschaftsdatenbanken

In diesem Szenario wird die Planung, Einrichtung und Implementierung einer HADR-Konfiguration für eine Bank namens ExampleBANK beschrieben. Die Konfiguration verfügt über drei Bereitschaftsdatenbanken: eine Hauptbereitschaftsdatenbank und zwei Nebenbereitschaftsdatenbanken.

Hintergrund

Da das Bankwesen eine Branche ist, in der rund um die Uhr gearbeitet wird, spielt die Hochverfügbarkeit bei der Technologiestrategie eine ausschlaggebende Rolle. Zudem kam es bei ExampleBANK in Stadt A, in der sich die Zentrale der Bank befindet, wegen eines Hurricanes zu einem Beinaheunfall, sodass die Bank außerdem eine Disaster-Recovery-Strategie benötigt. HADR (High Availability Disaster Recovery) bietet eine Lösung, die der Bank dabei helfen kann, diese beiden Ziele mit einer einzigen Technologie zu realisieren, nämlich dem Einsatz mehrerer HADR-Bereitschaftsdatenbanken.

Die HADR-Lösung muss für ExampleBANK unbedingt die folgenden Anforderungen erfüllen:

Aggressive Zielsetzung hinsichtlich der Recovery-Zeit

Als Bank mit einem 24-Stunden-Onlineservice möchte ExampleBANK die Zeiten minimieren, in denen die Anwendungen keine Verbindung zur Datenbank herstellen können.

Aggressive Zielsetzung hinsichtlich des Recovery-Punkts
ExampleBANK kann keine Datenverluste tolerieren, sodass die Zielsetzung hinsichtlich des Recovery-Punkts möglichst nahe bei 0 sein sollte.

Geplante Nichtverfügbarkeit nahe Null
Die Datenbank von ExampleBANK soll nahezu kontinuierlich verfügbar sein, auch während geplanter Aktivitäten wie Upgrades oder Wartungsmaßnahmen.

Datenschutz durch geografische Streuung
Im Rahmen der gültigen Compliance-Standards soll die Funktion zur Wiederherstellung von Operationen an einem fernen Standort implementiert werden.

Einfache Implementierung und Verwaltung
Die überlastete IT-Abteilung von ExampleBANK sucht nach einer Lösung, die relativ leicht zu konfigurieren ist und über Automatisierungsfunktionen verfügt.

Die folgenden Szenarios machen deutlich, dass die HADR-Funktion im Modus für mehrere Bereitschaftsdatenbanken sämtliche Anforderungen von ExampleBANK erfüllen kann.

Planung für eine Konfiguration mit mehreren Bereitschaftsdatenbanken

Laut ExampleBANK soll die HADR-Konfiguration sowohl Hochverfügbarkeit als auch Disaster-Recovery-Schutz beinhalten. Deshalb entscheidet sich das Unternehmen für die maximale Anzahl an Bereitschaftsdatenbanken (3). Zur Realisierung der Zielsetzung hinsichtlich der Recovery-Zeit muss die Bank über eine Bereitschaftsdatenbank verfügen, die in enger Synchronisation mit der Primärdatenbank steht (Bereitschaftsdatenbank im Modus SYNC oder NEARSYNC) und mit der Primärdatenbank kollokiert ist. Es ist sinnvoll, diese Bereitschaftsdatenbank als Hauptbereitschaftsdatenbank festzulegen, da nur diese Datenbank alle Synchronisationsmodi unterstützt. Die Primärdatenbank und die Hauptbereitschaftsdatenbank befinden sich beide in der Zentrale von ExampleBANK in Stadt A und sind über ein LAN miteinander verbunden.

Darüber hinaus konfiguriert der Datenbankadministrator von ExampleBANK zwei Bereitschaftsdatenbanken in der Niederlassung der Bank in Stadt B, um die Bank vor Datenverlusten aufgrund von Katastrophenfällen zu schützen. Die Niederlassung ist über ein WAN mit der Zentrale in Stadt A verbunden. Die Entfernung zwischen den beiden Städten hat keine Auswirkungen auf die Primärdatenbank, da es sich bei den Bereitschaftsdatenbanken um Nebenbereitschaftsdatenbanken handelt, die automatisch im Modus SUPERASYNC ausgeführt werden. Der Datenbankadministrator kann die Kosten für diese zusätzlichen Datenbanken zusätzlich rechtfertigen, indem er eine dieser Datenbanken zur Verwendung der Funktion für Leseoperationen in der Bereitschaftsdatenbank und die andere zur Verwendung der Funktion für zeitverzögerte Wiedergabe konfiguriert. Darüber hinaus können diese Bereitschaftsdatenbanken durch ein Szenario für schrittweise Aktualisierung oder Wartungsmaßnahmen zur Aufrechterhaltung der Datenverfügbarkeit beitragen und so den Verlust des Schutzes durch HADR verhindern.

Einrichten einer Konfiguration mit mehreren Bereitschaftsdatenbanken

Der Datenbankadministrator von ExampleBANK erstellt ein Backup der geplanten Primärdatenbank HADR_DB:

```
DB2 BACKUP DB hadr_db TO backup_dir
```

Dann stellt der Datenbankadministrator das Backup auf jedem geplanten Bereitschaftshost wieder her, indem er den folgenden Befehl absetzt:

```
DB2 RESTORE DB hadr_db FROM backup_dir
```

Tipp: Weitere Informationen zu den Optionen zur Erstellung einer Bereitschaftsdatenbank finden Sie in „Initialisieren einer Bereitschaftsdatenbank" auf Seite 60.

Für die Erstkonfiguration ist nach Ansicht des Datenbankadministrators die Mehrheit der Standardkonfigurationseinstellungen ausreichend. Die folgenden Datenbankkonfigurationsparameter müssen jedoch wie bei einer normalen HADR-Konfiguration explizit gesetzt werden:

- **hadr_local_host**
- **hadr_local_svc**
- **hadr_remote_host**
- **hadr_remote_inst**
- **hadr_remote_svc**

Zur Ermittlung der korrekten Werte für diese Konfigurationsparameter legt der Datenbankadministrator den Hostnamen, die Portnummer und den Instanznamen der vier Datenbanken in der HADR-Konfiguration fest:

Tabelle 9. Hostname, Portnummer und Instanzname für die Datenbanken

Geplante Rolle	Hostname	Portnummer	Instanzname
Primär	host1	10	dbinst1
Hauptbereitschafts-datenbank	host2	40	dbinst2
Nebenbereitschafts-datenbank	host3	41	dbinst3
Nebenbereitschafts-datenbank	host4.ibm.com	42	dbinst4

Auf der Primärdatenbank entsprechen die Einstellungen für die Konfigurationsparameter **hadr_remote_host**, **hadr_remote_inst** und **hadr_remote_svc** dem Hostnamen, dem Instanznamen und der Portnummer der Hauptbereitschaftsdatenbank. Auf den Bereitschaftsdatenbanken entsprechen die Werte dieser Konfigurationsparameter dem Hostnamen, der Portnummer und dem Instanznamen der Primärdatenbank. Darüber hinaus verwendet der Datenbankadministrator den Hostnamen und die Portwerte zum Definieren des Konfigurationsparameters **hadr_target_list** für alle Datenbanken. Außerdem fügt der Datenbankadministrator (obwohl dies nicht erforderlich ist) die Informationen zu allen in der Konfiguration enthaltenen Bereitschaftsdatenbanken zur Zielliste der anderen Bereitschaftsdatenbanken hinzu. Weitere Informationen zu diesem Thema finden Sie in „Datenbankkonfiguration für HADR (High Availability Disaster Recovery)" auf Seite 43.

Wie bereits erwähnt möchte die Bank eine möglichst enge Synchronisierung zwischen der Primär- und der Hauptbereitschaftsdatenbank erreichen. Aus diesem Grund setzt der Datenbankadministrator den Parameter **hadr_syncmode** auf der Primärdatenbank auf SYNC. Obwohl der effektive Synchronisationsmodus der Hauptbereitschaftsdatenbank automatisch auf SYNC gesetzt wird, nachdem die Verbin-

dung zur Primärdatenbank hergestellt wurde, setzt der Datenbankadministrator den Parameter **hadr_syncmode** auf der Hauptbereitschaftsdatenbank dennoch auf SYNC. Wenn die Hauptbereitschaftsdatenbank dann mit der Primärdatenbank die Rollen tauscht, ist der Synchronisationsmodus für das neue Paar aus Primärdatenbank und Hauptbereitschaftsdatenbank ebenfalls auf SYNC gesetzt.

Der Datenbankadministrator legt host2, der sich in einer anderen Stadt als die Nebenbereitschaftsdatenbanken befindet, als Hauptbereitschaftsdatenbank für die Nebenbereitschaftsdatenbanken fest. Wenn eine der Nebenbereitschaftsdatenbanken die Rolle der Primärdatenbank übernimmt, ist als Synchronisationsmodus zwischen der Primärdatenbank und host2 am fernen Standort SUPERASYNC eine gute Wahl. Daher setzt der Datenbankadministrator den Parameter **hadr_syncmode** auf den Nebenbereitschaftsdatenbanken auf SUPERASYNC, obwohl der effektive Synchronisationsmodus dieser Datenbanken nach Herstellung der Verbindung zur Primärdatenbank automatisch auf SUPERASYNC gesetzt wird. Weitere Informationen zu diesem Thema finden Sie in „HADR-Synchronisationsmodus (High Availability Disaster Recovery)" auf Seite 67.

Schließlich informiert sich der Datenbankadministrator über die neue HADR-Funktion der verzögerten Wiedergabe, bei der eine Bereitschaftsdatenbank durch eine Verzögerung der Protokollwiedergabe absichtlich auf einem früheren Stand als die Primärdatenbank gehalten werden kann. Der Datenbankadministrator kommt zu dem Schluss, dass die Aktivierung dieser Funktion den Schutz der Daten von ExampleBANK vor fehlerhaften Transaktionen auf der Primärdatenbank erhöhen würde. Der Datenbankadministrator wählt für diese Funktion host4 (Nebenbereitschaftsdatenbank) aus und notiert, dass die Funktion inaktiviert werden muss, bevor host4 die Rolle der Primärdatenbank übernehmen kann. Weitere Informationen zu diesem Thema finden Sie in „Verzögerte Wiedergabe bei HADR" auf Seite 213.

Der Datenbankadministrator setzt die folgenden Befehle ab, um die Konfigurationsparameter auf den einzelnen Datenbanken zu aktualisieren:

- Auf host1 (Primärdatenbank):

```
DB2 "UPDATE DB CFG FOR hadr_db USING
     HADR_TARGET_LIST  host2:40|host3:41|host4:42
     HADR_REMOTE_HOST  host2
     HADR_REMOTE_SVC   40
     HADR_LOCAL_HOST   host1
     HADR_LOCAL_SVC    10
     HADR_SYNCMODE     sync
     HADR_REMOTE_INST  db2inst2"
```

- Auf host2 (Hauptbereitschaftsdatenbank):

```
DB2 "UPDATE DB CFG FOR hadr_db USING
     HADR_TARGET_LIST  host1:10|host3:41|host4:42
     HADR_REMOTE_HOST  host1
     HADR_REMOTE_SVC   10
     HADR_LOCAL_HOST   host2
     HADR_LOCAL_SVC    40
     HADR_SYNCMODE     sync
     HADR_REMOTE_INST  db2inst1"
```

- Auf host3 (Nebenbereitschaftsdatenbank):

```
DB2 "UPDATE DB CFG FOR hadr_db USING
     HADR_TARGET_LIST  host2:40|host1:10|host4:42
     HADR_REMOTE_HOST  host1
     HADR_REMOTE_SVC   10
     HADR_LOCAL_HOST   host3
     HADR_LOCAL_SVC    41
     HADR_SYNCMODE     superasync
     HADR_REMOTE_INST  db2inst1"
```

- Auf host4 (Nebenbereitschaftsdatenbank):

```
DB2 "UPDATE DB CFG FOR hadr_db USING
     HADR_TARGET_LIST  host2.:40|host1:10|host3:41
     HADR_REMOTE_HOST  host2
     HADR_REMOTE_SVC   10
     HADR_LOCAL_HOST   host4
     HADR_LOCAL_SVC    42
     HADR_SYNCMODE     superasync
     HADR_REMOTE_INST  db2inst1
     HADR_REPLAY_DELAY 86400"
```

Schließlich möchte der Datenbankadministrator von ExampleBANK die HADR-Funktion für Leseoperationen in der Bereitschaftsdatenbank aus den folgenden Gründen aktivieren:

- Zur Durchführung von Onlineänderungen an einigen HADR-Konfigurationsparametern auf den Bereitschaftsdatenbanken.
- Zum Aufrufen der Tabellenfunktion MON_GET_HADR auf den Bereitschaftsdatenbanken.
- Zur Umleitung einiger Leseprozesse aus der Primärdatenbank.

Der Datenbankadministrator aktualisiert die Registrierdatenbankvariablen auf den Bereitschaftsdatenbanken, indem er die folgenden Befehle jeweils für host2, host3 und host4 absetzt:

```
DB2SET DB2_HADR_ROS=ON
DB2SET DB2_STANDBY_ISO=UR
```

Starten der HADR-Datenbanken

Der Datenbankadministrator startet die Bereitschaftsdatenbanken zuerst, indem er den folgenden Befehl jeweils für host2, host3 und host4 absetzt:

```
DB2 START HADR ON DB hadr_db AS STANDBY
```

Dann startet der Datenbankadministrator HADR auf der Primärdatenbank (host1):

```
DB2 START HADR ON DB hadr_db AS PRIMARY
```

Der Datenbankadministrator möchte prüfen, ob HADR betriebsbereit ist, und fragt dazu den Status der Datenbanken auf der Primärdatenbank (host1) ab, indem er den Befehl **db2pd** absetzt, der Informationen zu allen Bereitschaftsdatenbanken liefert:

```
db2pd -db hadr_db -hadr

  Database Member 0 -- Database hadr_db -- Active -- Up 0 days 00:23:17 -- Date 06/08/2011 13:57:23

                            HADR_ROLE = PRIMARY
                          REPLAY_TYPE = PHYSICAL
                        HADR_SYNCMODE = SYNC
                           STANDBY_ID = 1
                        LOG_STREAM_ID = 0
                           HADR_STATE = PEER
                  PRIMARY_MEMBER_HOST = host1
                     PRIMARY_INSTANCE = db2inst1
                       PRIMARY_MEMBER = 0
                  STANDBY_MEMBER_HOST = host2
                     STANDBY_INSTANCE = db2inst2
                       STANDBY_MEMBER = 0
                  HADR_CONNECT_STATUS = CONNECTED
             HADR_CONNECT_STATUS_TIME = 06/08/2011 13:38:10.199479 (1307565490)
           HEARTBEAT_INTERVAL(seconds) = 30
                  HADR_TIMEOUT(seconds) = 120
          TIME_SINCE_LAST_RECV(seconds) = 3
               PEER_WAIT_LIMIT(seconds) = 0
```

```
           LOG_HADR_WAIT_CUR(seconds) = 0.000
    LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
   LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                 LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 50772
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87616
            PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
            STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                  HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                   PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                   STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
             STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
         STANDBY_RECV_BUF_SIZE(pages) = 16
             STANDBY_RECV_BUF_PERCENT = 0
           STANDBY_SPOOL_LIMIT(pages) = 0
                PEER_WINDOW(seconds) = 0
            READS_ON_STANDBY_ENABLED = Y
    STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N
                          HADR_ROLE = PRIMARY
                      REPLAY_TYPE = PHYSICAL
                    HADR_SYNCMODE = SUPERASYNC
                       STANDBY_ID = 2
                    LOG_STREAM_ID = 0
                       HADR_STATE = REMOTE_CATCHUP
                PRIMARY_MEMBER_HOST = host1
                   PRIMARY_INSTANCE = db2inst1
                     PRIMARY_MEMBER = 0
                STANDBY_MEMBER_HOST = host3
                   STANDBY_INSTANCE = db2inst3
                     STANDBY_MEMBER = 0
                HADR_CONNECT_STATUS = CONNECTED
           HADR_CONNECT_STATUS_TIME = 06/08/2011 13:35:51.724447 (1307565351)
         HEARTBEAT_INTERVAL(seconds) = 30
               HADR_TIMEOUT(seconds) = 120
       TIME_SINCE_LAST_RECV(seconds) = 16
            PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
    LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
   LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                 LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
            PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
            STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                  HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                   PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                   STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
             STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
         STANDBY_RECV_BUF_SIZE(pages) = 16
             STANDBY_RECV_BUF_PERCENT = 0
           STANDBY_SPOOL_LIMIT(pages) = 0
                PEER_WINDOW(seconds) = 0
            READS_ON_STANDBY_ENABLED = Y
    STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N
                          HADR_ROLE = PRIMARY
                      REPLAY_TYPE = PHYSICAL
                    HADR_SYNCMODE = SUPERASYNC
                       STANDBY_ID = 3
                    LOG_STREAM_ID = 0
                       HADR_STATE = REMOTE_CATCHUP
                PRIMARY_MEMBER_HOST = host1
                   PRIMARY_INSTANCE = db2inst1
                     PRIMARY_MEMBER = 0
                STANDBY_MEMBER_HOST = host4
                   STANDBY_INSTANCE = db2inst4
                     STANDBY_MEMBER = 0
```

```
                 HADR_CONNECT_STATUS = CONNECTED
            HADR_CONNECT_STATUS_TIME = 06/08/2011 13:46:51.561873 (1307566011)
          HEARTBEAT_INTERVAL(seconds) = 30
                HADR_TIMEOUT(seconds) = 120
       TIME_SINCE_LAST_RECV(seconds) = 6
            PEER_WAIT_LIMIT(seconds) = 0
         LOG_HADR_WAIT_CUR(seconds) = 0.000
   LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
  LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                 LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
           PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
           STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                 HADR_LOG_GAP(bytes) = 0
    STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
      STANDBY_RECV_REPLAY_GAP(bytes) = 0
                   PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                   STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
             STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
       STANDBY_RECV_BUF_SIZE(pages) = 16
           STANDBY_RECV_BUF_PERCENT = 0
         STANDBY_SPOOL_LIMIT(pages) = 0
              PEER_WINDOW(seconds) = 0
           READS_ON_STANDBY_ENABLED = Y
   STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N
```

Beispiele für Übernahmen im Modus für mehrere HADR-Bereitschaftsdatenbanken

Die in diesem Abschnitt beschriebenen Übernahmebeispiele (erzwungen und nicht erzwungen) im Modus für mehrere HADR-Bereitschaftsdatenbanken basieren auf einer Konfiguration mit drei Bereitschaftsdatenbanken. Die Beispiele sollen veranschaulichen, wie die automatische Rekonfiguration mehrerer Bereitschaftsdatenbanken in einer Übernahmesituation funktioniert.

- „Ordnungsgemäße Übernahme durch eine Hauptbereitschaftsdatenbank (Rollenwechsel)" auf Seite 242
- „Erzwungene Übernahme (Funktionsübernahme) durch eine Nebenbereitschaftsdatenbank" auf Seite 243
- „Erzwungene Übernahme (Funktionsübernahme) durch eine Nebenbereitschaftsdatenbank in einer SA MP-Umgebung" auf Seite 245

Die Erstkonfiguration für jedes dieser Beispiele sieht wie folgt aus:

- 1 Primärdatenbank (host1)
- 1 Hauptbereitschaftsdatenbank (host2)
- 2 Nebenbereitschaftsdatenbanken (host3 und host4)

Alle Datenbanken haben den Namen 'hadr_db'. Der Synchronisationsmodus der Primärdatenbank und der Hauptbereitschaftsdatenbank ist auf SYNC und der Synchronisationsmodus der Bereitschaftsdatenbanken ist auf SUPERASYNC gesetzt.

Die Konfigurationswerte für die einzelnen Datenbanken können Sie Tabelle 10 entnehmen.

Tabelle 10. Konfigurationswerte für die einzelnen HADR-Datenbanken

Konfigurationsparameter	Host1	Host2	Host3	Host4
hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	host2:40 | host1:10 | host4:42	host2:40 | host1:10 | host3:41

Tabelle 10. Konfigurationswerte für die einzelnen HADR-Datenbanken (Forts.)

Konfigurations-parameter	Host1	Host2	Host3	Host4
hadr_remote_host	host2	host1	host1	host1
hadr_remote_svc	40	10	10	10
hadr_remote_inst	dbinst2	dbinst1	dbinst1	dbinst1
hadr_local_host	host1	host2	host3	host4
hadr_local_svc	10	40	41	42
Konfigurierter **hadr_syncmode** (Bezieht sich auf den explizit definierten Synchronisationsmodus, der verwendet wird, wenn die Datenbank die Rolle einer Primärdatenbank übernimmt.)	SYNC	SYNC	SUPERASYNC	SUPERASYNC
Effektiver **hadr_syncmode** (Bezieht sich auf den Synchronisationsmodus, der verwendet wird, wenn die Datenbank momentan eine Bereitschaftsdatenbank ist.)	nicht zutreffend	SYNC	SUPERASYNC	SUPERASYNC

Ordnungsgemäße Übernahme durch eine Hauptbereitschaftsdatenbank (Rollenwechsel)

Der Datenbankadministrator führt in der Hauptbereitschaftsdatenbank eine Übernahme durch, indem er auf host2 den folgenden Befehl absetzt:

```
 DB2 TAKEOVER HADR
ON DB hadr_db
```

Nach der erfolgreichen Übernahme wird host2 zur neuen Primärdatenbank und host1, der in **hadr_target_list** von host2 als erstes aufgelistet ist (siehe Tabelle 10 auf Seite 241), wird zur Hauptbereitschaftsdatenbank. Der Synchronisationsmodus dieser Datenbanken lautet SYNC, da der Parameter **hadr_syncmode** von host2 auf SYNC gesetzt ist. Die Parameter **hadr_remote_host** und **hadr_remote_svc** der Nebenbereitschaftsdatenbankziele host3 und host4 verweisen auf die alte Primärdatenbank host1, die Datenbanken werden jedoch automatisch an die neue Primärdatenbank host2 umgeleitet. Bei dieser Umleitung führen host3 und host4 eine (dauerhafte) Aktualisierung der Konfigurationsparameter **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** durch. Sie stellen als Nebenbereitschaftsdatenbanken eine erneute Verbindung zu host2 her und werden von host2 angewiesen, den effektiven Synchronisationsmodus SUPERASYNC zu verwenden (unabhängig von der lokalen Konfiguration für **hadr_syncmode**). Die Einstellungen für **hadr_syncmode** werden nicht dauerhaft aktualisiert. Die Konfigurationswerte für die einzelnen Datenbanken können Sie Tabelle 11 auf Seite 243 entnehmen.

Tabelle 11. Konfigurationswerte für die einzelnen HADR-Datenbanken nach einem Rollenwechsel. Die Zeilen 3 bis 5 in den Spalten 4 und 5 erscheinen in Fettdruck, um hervorzuheben, dass eine automatische Rekonfiguration erfolgt ist.

Konfigurations-parameter	Host1	Host2	Host3	Host4
hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	host2:40 | host1:10 | host4:42	host2:40 | host1:10 | host3:41
hadr_remote_host	host2	host1	**host2**	**host2**
hadr_remote_svc	40	10	**40**	**40**
hadr_remote_inst	dbinst2	dbinst1	**dbinst2**	**dbinst2**
hadr_local_host	host1	host2	host3	host4
hadr_local_svc	10	40	41	42
Konfigurierter **hadr_syncmode**	SYNC	SYNC	SUPERASYNC	SUPERASYNC
Effektiver **hadr_syncmode**	SYNC	nicht zutreffend	SUPERASYNC	SUPERASYNC

Anmerkung: Eine Reihe von Werten werden aus den folgenden Gründen nicht aktualisiert:

- Da die Konfigurationsparameter **hadr_remote_host** und **hadr_remote_svc** von host2 bereits auf die zugehörige Hauptbereitschaftsdatenbank host1 verweisen, werden diese Werte auf host2 nicht aktualisiert.
- Da die Konfigurationsparameter **hadr_remote_host** und **hadr_remote_svc** von host1 bereits auf die neue Primärdatenbank verweisen, werden diese Werte auf host1 nicht aktualisiert.
- Da der aktive Synchronisationsmodus von host1 SYNC und der aktive Synchronisationsmodus von host3 und host4 SUPERASYNC lautet, kommt es zu keiner Änderung des effektiven Synchronisationsmodus.

Erzwungene Übernahme (Funktionsübernahme) durch eine Nebenbereitschaftsdatenbank

Ein umfassender Stromausfall in Stadt A führt dazu, dass die Primärdatenbank (host1) nicht mehr verfügbar ist. Normalerweise wäre die Hauptbereitschaftsdatenbank (host2), die sich im Modus SYNC befindet, die beste Wahl für eine Übernahme als neue Primärdatenbank. host2 ist aufgrund des Stromausfalls momentan jedoch ebenfalls nicht verfügbar. Der Datenbankadministrator fragt die beiden Nebenbereitschaftsdatenbanken ab, um festzustellen, welche dieser Datenbanken über die meisten Protokolldaten verfügt:

```
db2pd -hadr -db hadr_db | grep 'PRIMARY_LOG_FILE,PAGE,POS|STANDBY_LOG_FILE,PAGE,POS'
```

Der Datenbankadministrator stellt fest, dass host3 auf dem aktuellsten Stand ist (obwohl er bei der Protokollwiedergabe etwas hinten liegt), und wählt diesen Host als neue Primärdatenbank aus.

```
DB2 TAKEOVER HADR ON DB hadr_db BY FORCE
```

Nach einer erfolgreichen Übernahme wird host3 zur neuen Primärdatenbank. In der Zwischenzeit wird host2 wieder verfügbar. host3 informiert host2 und host4 darüber, dass er die Rolle der Primärdatenbank übernommen hat. Auf host3 werden die Werte für **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** so rekonfiguriert, dass sie auf host2 verweisen. Dieser Host fungiert als Hauptbereit-

schaftsdatenbank, weil er in **hadr_target_list** auf host3 als erster Eintrag aufgeführt ist. Auf host2 wird der Synchronisationsmodus als SUPERASYNC konfiguriert, da dieser Modus der Einstellung für **hadr_syncmode** auf host3 entspricht. Darüber hinaus werden **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** (dauerhaft) aktualisiert. host4 wird automatisch auf die neue Primärdatenbank (host3) umgeleitet. Bei dieser Umleitung führt host4 eine (dauerhafte) Aktualisierung der Konfigurationsparameter **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** durch. Auf host1 findet erst dann eine automatische Rekonfiguration statt, wenn dieser wieder verfügbar ist. Die Konfigurationswerte für die einzelnen Datenbanken können Sie Tabelle 12 entnehmen.

Tabelle 12. Konfigurationswerte für die einzelnen HADR-Datenbanken nach einer Funktionsübernahme. Die Zeilen 3 bis 5 in den Spalten 3 bis 5 erscheinen in Fettdruck, um hervorzuheben, dass eine automatische Rekonfiguration erfolgt ist.

Konfigurationsparameter	Host1 (nicht verfügbar)	Host2	Host3	Host4
hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	host2:40 | host1:10 | host4:42	host2:40 | host1:10 | host3:41
hadr_remote_host	host2	**host3**	**host2**	**host3**
hadr_remote_svc	40	**41**	**40**	**41**
hadr_remote_inst	dbinst2	**dbinst3**	**dbinst2**	**dbinst3**
hadr_local_host	host1	host2	host3	host4
hadr_local_svc	10	40	41	42
Konfigurierter **hadr_syncmode**	SYNC	SYNC	SUPERASYNC	SUPERASYNC
Effektiver **hadr_syncmode**	nicht zutreffend	**SUPERASYNC**	nicht zutreffend	SUPERASYNC

Nach kurzer Zeit wird host1 wieder verfügbar. Der Datenbankadministrator versucht, host1 als Bereitschaftsdatenbank zu starten. Da host1 jedoch über mehr Protokolle verfügt, als an host3 weitergegeben wurden, wird host1 beim ersten Handshake mit der neuen Primärdatenbank zurückgewiesen. Der Datenbankadministrator führt ein Backup der neuen Primärdatenbank durch, stellt dieses auf host1 wieder her und startet HADR auf diesem Host:

```
DB2 BACKUP DB hadr_db

DB2 RESTORE DB hadr_db

DB2 START HADR ON DB hadr_db AS STANDBY
```

host1 wurde rekonfiguriert (siehe Tabelle 13).

Tabelle 13. Konfigurationswerte für eine reintegrierte Bereitschaftsdatenbank. Mehrere Zeilen in Spalte 2 erscheinen in Fettdruck, um hervorzuheben, dass eine automatische Rekonfiguration erfolgt ist.

Konfigurationsparameter	Host1	Host2	Host3	Host4
hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	host2:40 | host1:10 | host4:42	host2:40 | host1:10 | host3:41
hadr_remote_host	**host3**	host3	host2	host3
hadr_remote_svc	**41**	41	40	41
hadr_remote_inst	**dbinst3**	dbinst3	dbinst2	dbinst3
hadr_local_host	host1	host2	host3	host4

Tabelle 13. Konfigurationswerte für eine reintegrierte Bereitschaftsdatenbank (Forts.). Mehrere Zeilen in Spalte 2 erscheinen in Fettdruck, um hervorzuheben, dass eine automatische Rekonfiguration erfolgt ist.

Konfigurations-parameter	**Host1**	**Host2**	**Host3**	**Host4**
hadr_local_svc	10	40	41	42
Konfigurierter **hadr_syncmode**	SYNC	SYNC	SUPERASYNC	SUPERASYNC
Effektiver **hadr_syncmode**	**SUPERASYNC**	SUPERASYNC	nicht zutreffend	SUPERASYNC

Wenn der Datenbankadministrator host1 wieder als Primärdatenbank einsetzen möchte, muss lediglich eine Zurücksetzungsoperation ausgeführt werden, um die ursprüngliche Konfiguration wiederherzustellen (siehe Tabelle 10 auf Seite 241).

Erzwungene Übernahme (Funktionsübernahme) durch eine Nebenbereitschaftsdatenbank in einer SA MP-Umgebung

Dieses Beispiel ähnelt dem obigen Beispiel, allerdings wurde HADR zur Automatisierung der Funktionsübernahme mit IBM Tivoli System Automation for Multiplatforms (SA MP) implementiert.

Ein Stromausfall in Stadt A führt dazu, dass die Hauptbereitschaftsdatenbank (host2) nicht mehr verfügbar ist. Daraufhin erfolgt ein Ausfall der Primärdatenbank (host1). Normalerweise veranlasst SA MP (Cluster-Manager) eine automatische Funktionsübernahme durch die Hauptbereitschaftsdatenbank (host2). Aufgrund des Stromausfalls muss jedoch eine der Nebenbereitschaftsdatenbanken das Übernahmeziel sein. Die Funktionsübernahme durch Nebenbereitschaftsdatenbanken kann nicht automatisiert werden, deshalb muss der Datenbankadministrator diese manuell ausführen. Zuvor muss der Datenbankadministrator jedoch sicherstellen, dass TSA inaktiviert ist, um die Möglichkeit einer *Split-Brain-Situation* auszuschließen, bei der mehrere Datenbanken unabhängig voneinander als Primärdatenbanken arbeiten, wenn host1 oder host2 verfügbar wird. Dazu setzt der Datenbankadministrator auf host1 und host2 (sobald diese verfügbar werden) den folgenden Befehl ab:

```
db2haicu -disable
```

Darüber hinaus muss der Datenbankadministrator dafür sorgen, dass host1 offline bleibt, um die Möglichkeit auszuschließen, dass die alte Primärdatenbank erneut gestartet wird, wenn ein Client eine Verbindung zu dieser Datenbank herstellt.

Der Datenbankadministrator fragt die beiden Nebenbereitschaftsdatenbank ab, um festzustellen, welche dieser Datenbanken über die meisten Protokolldaten verfügt:

```
db2pd -hadr -db hadr_db | grep 'STANDBY_LOG_FILE,PAGE,POS'
```

Der Datenbankadministrator stellt fest, dass host3 auf dem aktuellsten Stand ist und wählt diesen Host als neue Primärdatenbank aus.

Dann gibt der Datenbankadministrator eine erzwungene Übernahme auf host3 an:

```
DB2 TAKEOVER HADR ON DB hadr_db BY FORCE
```

Nach einer erfolgreichen Übernahme wird host3 zur neuen Primärdatenbank. In der Zwischenzeit wird host2 wieder verfügbar. host3 informiert host2 und host4 darüber, dass er die Rolle der Primärdatenbank übernommen hat. Auf host3 wer-

den die Werte für **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** so rekonfiguriert, dass sie auf host2 verweisen. Dieser Host fungiert als Hauptbereitschaftsdatenbank, weil er in **hadr_target_list** auf host3 als erster Eintrag aufgeführt ist. Auf host2 wird der Synchronisationsmodus als SUPERASYNC konfiguriert, da dieser Modus der Einstellung für **hadr_syncmode** auf host3 entspricht. Darüber hinaus werden **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** (dauerhaft) aktualisiert. host4 wird automatisch auf die neue Primärdatenbank (host3) umgeleitet. Bei dieser Umleitung führt host4 eine (dauerhafte) Aktualisierung der Konfigurationsparameter **hadr_remote_host**, **hadr_remote_svc** und **hadr_remote_inst** durch. Auf host1 findet keine automatische Rekonfiguration statt. Die Konfigurationswerte für die einzelnen Datenbanken können Sie Tabelle 14 entnehmen.

Tabelle 14. Konfigurationswerte für die einzelnen HADR-Datenbanken nach einer Funktionsübernahme. Die Zeilen 3 bis 5 in den Spalten 3 bis 5 erscheinen in Fettdruck, um hervorzuheben, dass eine automatische Rekonfiguration erfolgt ist.

Konfigurations-parameter	Host1 (nicht verfügbar)	Host2	Host3	Host4
hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	host2:40 | host1:10 | host4:42	host2:40 | host1:10 | host3:41
hadr_remote_host	host2	**host3**	**host2**	**host3**
hadr_remote_svc	40	**41**	**40**	**41**
hadr_remote_inst	dbinst2	**dbinst3**	**dbinst2**	**dbinst3**
hadr_local_host	host1	host2	host3	host4
hadr_local_svc	10	40	41	42
Konfigurierter **hadr_syncmode**	SYNC	SYNC	SUPERASYNC	SUPERASYNC
Effektiver **hadr_syncmode**	nicht zutreffend	**SUPERASYNC**	nicht zutreffend	SUPERASYNC

HADR-Funktion für Leseoperationen in der Bereitschaftsdatenbank

Sie können die Funktionalität für Leseoperationen in der Bereitschaftsdatenbank verwenden, um in Ihrer HADR-Lösung schreibgeschützte Operationen in der Bereitschaftsdatenbank auszuführen (HADR = High Availability and Disaster Recovery). Leseoperationen, die in der Bereitschaftsdatenbank ausgeführt werden, beeinträchtigen nicht die Hauptfunktion der Bereitschaftsdatenbank, die darin besteht, die von der Primärdatenbank übertragenen Protokolle anzuwenden.

Die Funktion für Leseoperationen in der Bereitschaftsdatenbank trägt zur Senkung der Gesamtbetriebskosten Ihrer HADR-Installation bei. Die erweiterte Funktionalität der Bereitschaftsdatenbank gibt Ihnen die Möglichkeit, diese Datenbank auf neue Art und Weise zu verwenden, z. B. durch die Übernahme eines Teils der Verarbeitungsprozesse, die ansonsten auf der Primärdatenbank ausgeführt würden. Die dadurch entlastete Primärdatenbank kann somit andere Verarbeitungsprozesse übernehmen.

Die Lese- und Schreibclients werden nach wie vor mit der Primärdatenbank verbunden; zusätzlich können die Leseclients nun aber auch eine Verbindung zu der zu Leseprozessen befähigten oder *aktiven Bereitschaftsdatenbank* herstellen, sofern sich diese nicht im Status 'Lokales Catch-up' oder im Zeitfenster für das Anwenden von Protokollen befindet. Die Hauptfunktion einer aktiven Bereitschaftsdatenbank ist aber nach wie vor das Anwenden der von der Primärdatenbank übertragenen

Protokolle. Folglich sollten die Daten in der Bereitschaftsdatenbank im Prinzip mit den Daten in der Primärdatenbank identisch sein. Im Falle einer Funktionsübernahme werden alle Benutzerverbindungen zur Bereitschaftsdatenbank beendet und die Bereitschaftsdatenbank fungiert anschließend als neue Primärdatenbank.
Alle Arten von Leseabfragen, einschließlich verschiebbarer und nicht verschiebbarer Cursor, werden für die Bereitschaftsdatenbank unterstützt. Die Lesefunktionalität wird in allen vier HADR-Synchronisationsmodi (SYNC, NEARSYNC, ASYNC und SUPERASYNC) und in allen HADR-Statuszuständen (außer lokales Catch-up) unterstützt.

Aktivieren von Leseoperationen in der Bereitschaftsdatenbank

Sie können die Funktion für Leseoperationen in der Bereitschaftsdatenbank unter Verwendung der Registrierungsvariablen **DB2_HADR_ROS** für Ihre HADR-Bereitschaftsdatenbank aktivieren (HADR = High Availability and Disaster Recovery).

Vorbereitende Schritte

Es wird empfohlen, den Datenbankkonfigurationsparameter **logindexbuild** auf ON zu setzen. Dies verhindert Leistungsbeeinträchtigungen durch Abfragezugriffspläne, da ungültige Indizes vermieden werden.

Es wird außerdem empfohlen, eine virtuelle IP-Adresse zu verwenden, wenn Sie Leseoperationen in der Bereitschaftsdatenbank aktiviert haben. Die Clientweiterleitung unterscheidet nicht zwischen Datenbanken, in die geschrieben werden kann (Primär- und Standarddatenbanken), und Datenbanken, in denen nur gelesen werden kann (Bereitschaftsdatenbanken). Das Konfigurieren der Clientweiterleitung zwischen der Primär- und Bereitschaftsdatenbank kann Anwendungen aber auch an die Datenbank weiterleiten, auf der sie nicht ausgeführt werden sollen.

Vorgehensweise

1. Setzen Sie die Registrierdatenbankvariable **DB2_HADR_ROS** auf ON.
2. Richten Sie die Primär- und Bereitschaftsdatenbank für HADR ein und initialisieren Sie die Datenbanken. Weitere Informationen finden Sie in „Initialisieren von High Availability Disaster Recovery (HADR)" auf Seite 38.

Ergebnisse

Ihre Bereitschaftsdatenbank wird nun als eine *aktive Bereitschaftsdatenbank* betrachtet, d. h. sie akzeptiert schreibgeschützte Workloads.

Nächste Schritte

Anschließend können Sie die Bereitschaftsdatenbank nach Bedarf nutzen, beispielsweise, indem Sie einen Teil der schreibgeschützten Workloads auf ihr ausführen.

Damit Ihre Anwendungen langfristig auf Ihre Bereitschaftsdatenbank zugreifen können, führen Sie die im White Paper „Continuous access to Read on Standby databases using Virtual IP addresses" aufgelisteten Schritte durch.

Zeitnahe Aktualisierung der aktiven Bereitschaftsdatenbank

Änderungen in der HADR-Primärdatenbank werden nicht in jedem Fall sofort in der aktiven HADR-Bereitschaftsdatenbank nachvollzogen. Nicht festgeschriebene Änderungen in der Primärdatenbank werden möglicherweise erst in die Bereitschaftsdatenbank repliziert, wenn die Primärdatenbank ihre Protokolle auf Platte geschrieben oder übertragen hat.

Protokolle werden nur dann definitiv auf Platte geschrieben und somit an die Bereitschaftsdatenbank übertragen, wenn sie festgeschrieben wurden. Das Schreiben von Protokollen kann auch durch nicht deterministische Bedingungen ausgelöst werden, beispielsweise dann, wenn der Protokollpuffer voll ist. Folglich ist es möglich, dass nicht festgeschriebene Änderungen in der Primärdatenbank für eine lange Zeit im Protokollpuffer der Primärdatenbank verbleiben. Da die Protokollfunktion es vermeidet, Seiten nur teilweise zu schreiben, betrifft diese Situation in erster Linie weniger umfangreiche, nicht festgeschriebene Änderungen in der Primärdatenbank.

Wenn es Ihre Verarbeitungsprozesse erfordern, dass die Daten in der Bereitschaftsdatenbank im Prinzip jederzeit mit denen in der Primärdatenbank identisch sind, sollten Sie Ihre Transaktionen möglicherweise häufiger festschreiben.

Isolationsstufe in der aktiven Bereitschaftsdatenbank

In einer aktiven Bereitschaftsdatenbank (d. h. in einer HADR-Bereitschaftsdatenbank, in der Leseprozesse ausgeführt werden können) wird nur die Isolationsstufe UR (Uncommitted Read) unterstützt. Wenn eine Anwendung, Anweisung oder Unteranweisung eine höhere Isolationsstufe als UR anfordert, wird eine Fehlernachricht zurückgegeben (SQL1773N Ursachencode 1).

Wenn Sie eine andere Isolationsstufe als UR benötigen, sollte für die betreffende Anwendung die Primär- statt der Bereitschaftsdatenbank verwendet werden. Wenn Sie nur vermeiden wollen, dass diese Fehlernachricht zurückgegeben wird, setzen Sie die Registrierdatenbankvariable **DB2_STANDBY_ISO** auf UR. Wenn **DB2_STANDBY_ISO** auf UR gesetzt ist, wird automatisch die Isolationsstufe UR erzwungen. Diese Einstellung hat Vorrang vor allen anderen Einstellungen der Isolationsstufe, wie z. B. Anweisungs- und Paketisolation.

Zeitfenster für das Anwenden von Protokollen in der aktiven Bereitschaftsdatenbank

Wenn eine aktive HADR-Bereitschaftsdatenbank DDL-Protokollsätze oder Wartungsoperationen anwendet, beginnt das *Zeitfenster für das Anwenden von Protokollen* (Replay-only Window). Während sich die Bereitschaftsdatenbank in diesem Zeitfenster befindet, werden die Verbindungen, die bereits zur Bereitschaftsdatenbank bestehen, beendet und neue Verbindungen zur Bereitschaftsdatenbank werden blockiert (SQL1776N Ursachencode 4).

Wenn das Anwenden aller aktiven DDL-Protokollsätze und Wartungsoperationen beendet wurde, können wieder Verbindungen zur Bereitschaftsdatenbank hergestellt werden.

Die einzigen Benutzerverbindungen in einer Bereitschaftsdatenbank, die während des Zeitfensters für das Anwenden von Protokollen aktiv bleiben können, sind diejenigen, die die Befehle **DEACTIVATE DATABASE** oder **TAKEOVER** ausführen. Wenn Anwendungen zu Beginn des Zeitfensters für das Anwenden von Protokollen zwangsweise unterbrochen werden, wird eine Fehlernachricht angezeigt (SQL1224N). Abhängig von der Anzahl der Leseanwendungen, die mit der aktiven Bereitschaftsdatenbank verbunden sind, ergibt sich gegebenenfalls eine leichte Verzögerung, bevor die DDL-Protokollsätze oder Wartungsoperationen in der Bereitschaftsdatenbank angewendet werden.

Es gibt eine Reihe von DDL-Anweisungen und Wartungsoperationen, die bei Ausführung in der HADR-Primärdatenbank ein Zeitfenster für das Anwenden von Protokollen in der Bereitschaftsdatenbank auslösen. Die folgenden Listen sind nicht vollständig.

DDL-Anweisungen

- CREATE, ALTER oder DROP TABLE (außer DROP TABLE für DGTT)
- CREATE GLOBAL TEMP TABLE
- TRUNCATE TABLE
- RENAME TABLE
- RENAME TABLESPACE
- CREATE, DROP oder ALTER INDEX
- CREATE oder DROP VIEW
- CREATE, ALTER oder DROP TABLESPACE
- CREATE, ALTER oder DROP BUFFER POOL
- CREATE, ALTER oder DROP FUNCTION
- CREATE, ALTER oder DROP PROCEDURE
- CREATE oder DROP TRIGGER
- CREATE, ALTER oder DROP TYPE
- CREATE, ALTER oder DROP ALIAS
- CREATE oder DROP SCHEMA
- CREATE, ALTER oder DROP METHOD
- CREATE, ALTER oder DROP MODULE
- CREATE, ALTER oder DROP NICKNAME
- CREATE, ALTER oder DROP SEQUENCE
- CREATE, ALTER oder DROP WRAPPER
- CREATE, ALTER oder DROP FUNCTION MAPPING
- CREATE oder DROP INDEX EXTENSION
- CREATE oder DROP INDEX FOR TEXT
- CREATE oder DROP EVENT MONITOR
- CREATE, ALTER oder DROP SECURITY LABEL
- CREATE, ALTER oder DROP SECURITY LABEL COMPONENT
- CREATE, ALTER oder DROP SECURITY POLICY
- CREATE oder DROP TRANSFORM
- CREATE, ALTER oder DROP TYPE MAPPING
- CREATE, ALTER oder DROP USER MAPPING
- CREATE oder DROP VARIABLE
- CREATE, ALTER oder DROP WORKLOAD
- GRANT USAGE ON WORKLOAD
- REVOKE USAGE ON WORKLOAD
- CREATE, ALTER oder DROP SERVICE CLASS
- CREATE, ALTER oder DROP WORK CLASS SET
- CREATE, ALTER oder DROP WORK ACTION SET
- CREATE, ALTER oder DROP THRESHOLD
- CREATE, ALTER oder DROP HISTOGRAM TEMPLATE
- AUDIT
- CREATE, ALTER oder DROP AUDIT POLICY
- CREATE oder DROP ROLE
- CREATE, ALTER oder DROP TRUSTED CONTEXT
- REFRESH TABLE

- SET INTEGRITY

Wartungsoperationen
- REORG (klassisch oder offline)
- REORG (INPLACE oder online)
- INDEX REORG (INDEXES ALL, einzelner Index)
- Reorganisation für Freigabe bei MDC- und ITC-Tabellen
- LOAD
- BIND oder REBIND
- db2rbind
- RUNSTATS
- Versetzen von Tabellen
- Automatische Statistikerstellung
- Automatische Reorganisation
- Echtzeitstatistiken

Weitere Operationen oder Aktionen
- Automatische Wörterverzeichniserstellung für Tabellen mit dem Attribut COMPRESS YES
- Asynchrone Indexbereinigung für Tabellenpartition mit aufgehobener Zuordnung
- Impliziter Rebind
- Impliziter Indexrebuild
- Manuelle Aktualisierung von Statistikdaten
- Verzögerter MDC-Rollout
- Asynchrone Indexbereinigung nach MDC-Rollout
- Wiederverwendung eines gelöschten MDC- oder ITC-Blocks beim Einfügen in MDC- oder ITC-Tabelle
- Asynchrone Hintergrundprozesse, die die Katalogtabellen SYSJOBS und SYSTASKS für Einfüge-, Aktualisierungs- und Löschtasks aktualisieren

Überwachung des Zeitfensters für das Anwenden von Protokollen

Um das Zeitfenster für das Anwenden von Protokollen für eine Bereitschaftsdatenbank zu überwachen, verwenden Sie den Befehl **db2pd** mit der Option **-hadr**. Im Beispiel unten sind die folgenden drei Elemente relevant:
- **ReplayOnlyWindowStatus** gibt an, ob zu diesem Zeitpunkt DDL- oder Wartungsoperationen in der Bereitschaftsdatenbank angewendet werden. Normalerweise wird der Wert "Inactive" verwendet, doch wenn das Zeitfenster für das Anwenden von Protokollen aktiv ist, wird der Wert "Active" verwendet.
- **ReplayWindowStartTime** gibt den Zeitpunkt an, zu dem das aktuelle Zeitfenster für das Anwenden von Protokollen (falls vorhanden) aktiviert wurde.
- **MaintenanceTxCount** oder **DDLTxCount** gibt die Gesamtzahl der vorhandenen nicht festgeschriebenen DDL- oder Wartungstransaktionen an, die bis zu diesem Zeitpunkt im aktuellen Zeitfenster für das Anwenden von Protokollen (falls vorhanden) ausgeführt wurden.

```
db2pd -db hadrdb -hadr
Database Partition 0 -- Database HADRDB -- Active -- Up 0 days 00:00:06

HADR-Information:
```

```
Role     State    SyncMode    HeartBeatsMissed    LogGapRunAvg (bytes)
Standby Peer  Nearsync 0                  0

ConnectStatus   ConnectTime                          Timeout
Connected     Sat Jun 15 03:09:35 2008          120

ReplayOnlyWindowStatus       ReplayOnlyWindowStartTime           MaintenanceTxCount
Active                       Sun Jun 16 08:09:35 2008            5

LocalHost  LocalService
skua       52601

RemoteHost  RemoteService  RemoteInstance
gull        52600          vinci

PrimaryFile    PrimaryPg    PrimaryLSN
S0000000.LOG 1          0x000000000137126F

StandByFile  StandByPg  StandByLSN
S0000000.LOG 0          0x000000000137092E
```

Empfehlungen für das Minimieren des Einflusses des Zeitfensters für das Anwenden von Protokollen

Da die Operationen zum Anwenden von Protokollen in einer HADR-Bereitschaftsdatenbank höhere Priorität als Leseprozesse haben, können häufige Lesezeitfenster für Leseprozesse, die mit der Bereitschaftsdatenbank verbunden sind oder eine Verbindung zu ihr herstellen wollen, zu Unterbrechungen führen. Die folgenden Empfehlungen können zur Minimierung dieser Einflüsse beitragen:

- Führen Sie DDL- und Wartungsoperationen während eines geplanten Wartungszeitfensters - vorzugsweise zu Zeiten geringer Systemauslastung - durch.
- Führen Sie DDL-Operationen gebündelt und nicht in einzelnen Gruppen aus.
- Führen Sie **REORG** oder **RUNSTATS** nur für die erforderlichen Tabellen aus und nicht für alle Tabellen.
- Verwenden Sie zum Beenden der Anwendungen in der aktiven Bereitschaftsdatenbank den Befehl **FORCE APPLICATION** mit der Option **ALL**, bevor Sie die DDL- oder Wartungsoperationen in der Primärdatenbank ausführen. Überwachen Sie das Zeitfenster für das Anwenden von Protokollen, um zu ermitteln, wann es inaktiv ist, und implementieren Sie die Anwendungen erneut in der Bereitschaftsdatenbank.

Vorübergehendes Beenden von Leseanwendungen bei einer aktiven Bereitschaftsdatenbank

Auch wenn eine aktive HADR-Bereitschaftsdatenbank zur Ausführung von schreibgeschützten Workloads verwendet werden kann, bleibt es ihre Hauptfunktion, Protokollsätze anzuwenden, um mit der HADR-Primärdatenbank synchron zu bleiben für den Fall, dass sie die Funktion der Primärdatenbank übernehmen muss.

In Fällen, in denen die schreibgeschützten Workloads dazu führen, dass das Anwenden der Protokollsätze verzögert wird, müssen Sie möglicherweise alle Verbindungen zur Bereitschaftsdatenbank vorübergehend beenden, damit der entstandene Rückstand aufgeholt werden kann.

Informationen zu diesem Vorgang

Gehen Sie wie folgt vor, um den Zugriff von Leseanwendungen auf eine aktive Bereitschaftsdatenbank vorübergehend auszusetzen.

Vorgehensweise

1. Setzen Sie den Befehl **FORCE APPLICATION** ab. Dies beendet die bestehenden Verbindungen zur Bereitschaftsdatenbank.
2. Ändern Sie die virtuelle IP-Konfiguration. Dies verhindert, dass neue Verbindungen zur Bereitschaftsdatenbank aufgebaut werden.

Nächste Schritte

Wenn die Bereitschaftsdatenbank den Rückstand auf die Primärdatenbank durch das Anwenden von Protokollsätzen aufgeholt hat, müssen Sie die virtuelle IP-Konfiguration wieder auf die ursprüngliche Einstellung zurücksetzen, sodass die Verbindungen zur aktiven Bereitschaftsdatenbank wiederaufgenommen werden können.

Einschränkungen bei Leseoperationen in der Bereitschaftsdatenbank

Die HADR-Funktion für Leseoperationen in der Bereitschaftsdatenbank gibt Ihnen die Möglichkeit, schreibgeschützte Workloads in einer aktiven HADR-Bereitschaftsdatenbank auszuführen (HADR = High Availability and Disaster Recovery). Neben der Tatsache, dass nur Leseoperationen möglich sind, gelten weitere Einschränkungen, die beachtet werden sollten.

- Schreiboperationen sind in der Bereitschaftsdatenbank nicht zulässig. Unter einer Schreiboperation wird in diesem Kontext eine Operation verstanden, die Änderungen an permanenten Datenbankobjekten wie Katalogen, Tabellen und Indizes vornimmt. Schreiboperationen in der Bereitschaftsdatenbank geben einen Fehler zurück (SQL1773N Ursachencode 5). In keinem Fall können Operationen ausgeführt werden, die zur Erstellung von Protokollsätzen in der Bereitschaftsdatenbank führen würden.
- Benutzerverbindungen zur Bereitschaftsdatenbank sind während des Anwendens von DDL-Protokollsätzen oder bei Wartungsoperationen (im *Zeitfenster für das Anwenden von Protokollen*) nicht möglich. Weitere Informationen hierzu finden Sie im Abschnitt „Zeitfenster für das Anwenden von Protokollen in der aktiven Bereitschaftsdatenbank" auf Seite 248.
- Benutzerverbindungen zur Bereitschaftsdatenbank können nicht hergestellt werden, wenn sich die Datenbank im Status 'Lokales Catch-up' befindet. Clients, die versuchen, bei diesem Status eine Verbindung herzustellen, erhalten eine Fehlernachricht (SQL1776N Ursachencode 1).
- In der Bereitschaftsdatenbank wird nur die Isolationsstufe UR (Uncommitted Read) unterstützt. Anwendungen, Anweisungen oder Unteranweisungen, die eine höhere Isolationsstufe anfordern, erhalten eine Fehlernachricht (SQL1773N Ursachencode 1). Weitere Informationen hierzu finden Sie in „Isolationsstufe in der aktiven Bereitschaftsdatenbank" auf Seite 248.
- Die Prüfkonfiguration auf Instanzebene wird nicht in die Bereitschaftsdatenbank repliziert. Sie müssen mit dem Tool db2audit sicherstellen, dass die Prüfeinstellungen auf Instanzebene in der Primär- und Bereitschaftsdatenbank identisch sind.
- Deklarierte temporäre Tabellen (DGTTs) werden in der Bereitschaftsdatenbank nicht unterstützt. Bei dem Versuch, solche Tabellen in der Bereitschaftsdatenbank zu erstellen oder dort auf sie zuzugreifen, wird eine Fehlernachricht angezeigt (SQL1773N Ursachencode 4).
- Erstellte temporäre Tabellen (CGTTs) können nur in der Primärdatenbank erstellt werden und ihre Definitionen werden in die Bereitschaftsdatenbank repliziert.

Da der Zugriff auf CGTTs in der Bereitschaftsdatenbank jedoch nicht unterstützt wird, wird bei dem Versuch, dort auf sie zuzugreifen, ein Fehler zurückgegeben (SQL1773N Ursachencode 4).

- Die Erstellung von CGTTs in der Primärdatenbank führt dazu, dass in der Bereitschaftsdatenbank das Zeitfenster für das Anwenden von Protokollen ausgelöst wird.
- Auf Tabellen, die mit der Option NOT LOGGED INITIALLY erstellt wurden (NLI-Tabellen), kann in der Bereitschaftsdatenbank nicht zugegriffen werden. Anwendungen, die versuchen, eine NLI-Tabelle in der Bereitschaftsdatenbank zu lesen, erhalten eine Fehlernachricht (SQL1477N).
- Abfragen in der Bereitschaftsdatenbank können nur SMS-Tabellenbereiche für temporäre Systemtabellen verwenden. Eine für die Bereitschaftsdatenbank ausgeführte Abfrage, die DMS-Tabellenbereiche für temporäre Systemtabellen verwendet, kann einen Fehler verursachen (SQL1773N Ursachencode 5).
- XML- und LOB-Daten (großes Objekt) müssen für eine erfolgreiche Abfrage integriert sein, da ansonsten ein Fehler (SQL1773N Ursachencode 3) zurückgegeben wird.
- Die folgenden Daten können nicht abgefragt werden: LF (Langfeld), ein einzigartiger Typ auf der Basis einer dieser Datentypen und Spalten des strukturierten Typs. Bei dem Versuch, einen dieser Datentypen abzufragen, wird eine Fehlernachricht zurückgegeben (SQL1773N Ursachencode 3).
- EXPLAIN-Tools (**db2exfmt**, **db2expln**) und **db2batch**-Tools werden in der Bereitschaftsdatenbank nicht unterstützt (SQL1773N Ursachencode 5). Wenn Sie die Leistung der schreibgeschützten Workloads analysieren wollen, müssen Sie diese Tools zunächst in der Primärdatenbank ausführen, die erforderlichen Optimierungen an den Verarbeitungsprozessen (Workload) in der Primärdatenbank vornehmen und dann die optimierten Verarbeitungsprozesse in die Bereitschaftsdatenbank verlagern.
- Explizites Binden und erneutes Binden sowie ein impliziter Rebind von Paketen wird in der Bereitschaftsdatenbank nicht unterstützt. Versuche, statische Pakete auszuführen, die sich auf inaktivierte Objekte beziehen, sowie implizite Rebinds dieser Pakete führen zu einem Fehler (SQL1773N Ursachencodes 5 bzw. 6). Stattdessen müssen Sie Pakete an die Primärdatenbank binden und das Paket in der Bereitschaftsdatenbank ausführen, nachdem die Änderung in die Bereitschaftsdatenbank repliziert wurde.
- Der Speichermanager für automatische Leistungsoptimierung (STMM) wird in der Bereitschaftsdatenbank nicht unterstützt. Wenn Sie die Bereitschaftsdatenbank optimieren möchten (entweder damit die schreibgeschützten Workloads ausgeführt werden können oder um gute Leistungswerte nach der Übernahme zu erzielen), müssen Sie dies manuell tun.
- DDL-Anweisungen des Workload-Managers (WLM) in der Primärdatenbank werden in der Bereitschaftsdatenbank angewendet, aber sie werden in der Bereitschaftsdatenbank nicht wirksam; dagegen werden Definitionen in dem Datenbankbackup, das zur Einrichtung der Bereitschaftsdatenbank verwendet wurde, in der zu Leseprozessen befähigten Bereitschaftsdatenbank aktiviert.
- Das Erstellen und Ändern von Sequenzen wird in der Bereitschaftsdatenbank nicht unterstützt. Außerdem können Sie den Ausdruck NEXT VALUE nicht verwenden, um den nächsten Wert in einer Sequenz zu generieren.
- Die erneute Aktivierung inaktivierter Objekte während der Laufzeit wird in der Bereitschaftsdatenbank nicht unterstützt.
- Die Bereitschaftsdatenbank kann nicht als 'Federation Server' konfiguriert werden.

- Backup- und Archivierungsoperationen werden in der Bereitschaftsdatenbank nicht unterstützt.
- Quiesce-Operationen werden in der Bereitschaftsdatenbank nicht unterstützt.

Feststellen und Handhaben von Systemausfällen in einer Hochverfügbarkeitslösung

Die Implementierung einer Lösung mit hoher Verfügbarkeit verhindert nicht, dass Hardware- oder Softwarefehler auftreten. Das Vorhandensein redundanter Systeme und ein Funktionsübernahmemechanismus stellen jedoch sicher, dass Ihre Lösung Fehler feststellt und handhaben kann, und dass Verarbeitungsprozesse weitergeleitet werden, sodass Benutzeranwendungen immer noch funktionieren können.

Vorgehensweise

Wenn ein Fehler auftritt, muss Ihre Datenbank wie folgt vorgehen:

1. Feststellen des Fehlers.

   Die Software für die Funktionsübernahme kann die Überwachungssignalfunktion verwenden, um die Verfügbarkeit von Systemkomponenten zu bestätigen. Ein Überwachungssignalmonitor überwacht die reguläre Kommunikation aller Komponenten im System. Wenn der Überwachungssignalmonitor von einer Komponente keine Signale mehr empfängt, signalisiert der Überwachungssignalmonitor dem System, dass die Komponente fehlgeschlagen ist.
2. Handhabung des Fehlers: Funktionsübernahme.
   a. Eine sekundäre Komponente muss identifiziert, online geschaltet und initialisiert werden, damit sie die Operationen der fehlgeschlagenen Komponente übernehmen kann.
   b. Verarbeitungsprozesse an die sekundäre Komponente weiterleiten.
   c. Entfernen der fehlgeschlagenen Komponente aus dem System.
3. Durchführen einer Recovery nach dem Fehler.

   Schlägt ein primärer Datenbankserver fehl, ist die oberste Priorität, die Clients an einen alternativen Server weiterzuleiten oder die Funktionsübernahme durch eine Bereitschaftsdatenbank zu beginnen, sodass die Clientanwendungen in ihrem Betrieb so wenig wie möglich unterbrochen werden. War die Funktionsübernahme erfolgreich, müssen Sie den Fehler auf dem fehlgeschlagenen Datenbankserver reparieren, sodass dieser wieder in die Lösung integriert werden kann. "Reparatur" kann ggf. einfach ein Neustart des Servers sein.
4. Rückkehr zum normalen Betrieb.

   Sobald das fehlgeschlagene Datenbanksystem repariert ist, müssen Sie es wieder in die Datenbanklösung integrieren. Sie können eine fehlgeschlagene Primärdatenbank als Bereitschaftsdatenbank für jene Datenbank reintegrieren, die die Funktion als Primärdatenbank übernommen hatte, als der Fehler auftrat. Sie könnten außerdem erzwingen, dass der reparierte Datenbankserver erneut seine vorherige Funktion als primärer Datenbankserver übernimmt.

Nächste Schritte

Die DB2-Datenbank kann einige dieser Schritte für Sie ausführen. Beispiel:

- Das Monitorelement für Überwachungssignale **hadr_heartbeat** von DB2 High Availability Disaster Recovery (HADR) kann feststellen, dass eine Primärdatenbank ausgefallen ist.

- Die DB2-Clientweiterleitung kann Workload von einem fehlgeschlagenen Datenbankserver an einen sekundären Datenbankserver weiterleiten.
- Der DB2-Fehlermonitor kann eine Datenbankinstanz erneut starten, die unerwartet beendet wurde.

Protokoll mit Benachrichtigungen für die Systemverwaltung

Beim Protokoll mit Benachrichtigungen für die Systemverwaltung (*instanzname*.nfy) handelt es sich um das Repository, aus dem Informationen zu zahlreichen Datenbankverwaltungs- und Datenbankwartungsaktivitäten abgerufen werden können. Ein Datenbankadministrator kann diese Informationen zum Diagnostizieren von Problemen, zur Optimierung der Datenbank oder einfach zur Überwachung der Datenbank verwenden.

Der DB2-Datenbankmanager schreibt die folgenden Arten von Informationen in das Protokoll mit Benachrichtigungen für die Systemverwaltung unter UNIX- und Linux-Betriebssystemplattformen (unter Windows-Betriebssystemplattformen werden Benachrichtigungen für die Systemverwaltung im Ereignisprotokoll aufgezeichnet):

- Status von DB2-Dienstprogrammen wie beispielsweise **REORG** und **BACKUP**
- Clientanwendungsfehler
- Serviceklassenänderungen
- Lizenzierungsaktivitäten
- Dateipfade
- Speicherprobleme
- Überwachungsaktivitäten
- Indexierungsaktivitäten
- Tabellenbereichsprobleme

Nachrichten des Protokolls mit Benachrichtigungen für die Systemverwaltung werden unter Verwendung des standardisierten Nachrichtenformats ebenfalls im Protokoll **db2diag** aufgezeichnet.

Hinweisnachrichten bieten zusätzliche Informationen als Ergänzung zu angegebenen SQLCODE-Werten.

Das Protokoll mit Benachrichtigungen für die Systemverwaltung kann in zwei unterschiedlichen Formaten vorliegen:

Einzelne Protokolldatei mit Benachrichtigungen für die Systemverwaltung
Eine aktive Protokolldatei mit Benachrichtigungen für die Systemverwaltung namens *instanzname*.nfy, deren Größe unbegrenzt wachsen kann. Dies ist das Standardformat, das immer dann vorliegt, wenn für den Konfigurationsparameter **diagsize** des Datenbankmanagers der Wert 0 (der Standardwert für diesen Parameter) angegeben ist.

Rotierende Protokolldateien mit Benachrichtigungen für die Systemverwaltung
Eine einzelne aktive Protokolldatei (namens *instanzname.N*.nfy, wobei *N* der bei 0 beginnende und kontinuierlich zunehmende Dateinamensindex ist), wenngleich mehrere Protokolldateien mit Benachrichtigungen für die Systemverwaltung an der vom Konfigurationsparameter **diagpath** definierten Position vorhanden sein können. Jede dieser Dateien kann bis zu einer Obergrenze anwachsen. Ist diese Obergrenze erreicht, wird die betreffende Datei geschlossen und eine neue Datei mit einem höheren Dateinamensindex (*instanzname.N+1*.nfy) erstellt und für die Protokollierung geöffnet. Die-

ses Format liegt vor, wenn der Konfigurationsparameter **diagsize** des Datenbankmanagers einen Wert ungleich null aufweist.

Anmerkung: Auf einer Windows-Betriebssystemplattform stehen weder einzelne noch rotierende Protokolldateien mit Benachrichtigungen für die Systemverwaltung zur Verfügung.

Sie können selbst entscheiden, welche dieser beiden Formate auf Ihrem System verwendet werden soll, indem Sie den Konfigurationsparameter **diagsize** des Datenbankmanagers entsprechend einstellen.

Konfiguration

Die Größe, die Position, die Ereignistypen und der Detaillierungsgrad der Aufzeichnungen lassen sich anhand der folgenden Konfigurationsparameter des Datenbankmanagers für die Protokolldateien mit Benachrichtigungen für die Systemverwaltung einstellen:

diagsize
Anhand des Werts für **diagsize** wird festgelegt, welches Format für die Protokolldatei mit Benachrichtigungen für die Systemverwaltung verwendet wird. Beim Wert 0 wird eine einzelne Protokolldatei mit Benachrichtigungen für die Systemverwaltung verwendet. Bei einem Wert ungleich null werden rotierende Protokolldateien mit Benachrichtigungen für die Systemverwaltung verwendet. Gleichzeitig gibt dieser Wert ungleich null auch die Gesamtgröße aller rotierenden Diagnoseprotokolldateien und aller rotierenden Protokolldateien mit Benachrichtigungen für die Systemverwaltung an. Wird der Parameter **diagsize** geändert, muss die Instanz erneut gestartet werden, damit der neue Wert wirksam wird. Ausführliche Informationen hierzu finden Sie im Thema „diagsize - Konfigurationsparameter für Diagnoseprotokolldateigröße".

diagpath
Für Diagnoseinformationen kann angegeben werden, ob sie in Protokolldateien mit Benachrichtigungen für die Systemverwaltung an der vom Konfigurationsparameter **diagpath** angegebenen Position geschrieben werden sollen. Ausführliche Informationen hierzu finden Sie im Thema „diagpath - Konfigurationsparameter für Diagnoseinformationsverzeichnispfad".

notifylevel
Die Ereignistypen und der Detaillierungsgrad der Aufzeichnungen für die Protokolldateien mit Benachrichtigungen für die Systemverwaltung können mithilfe des Konfigurationsparameters **notifylevel** angegeben werden. Ausführliche Informationen hierzu finden Sie im Thema „notifylevel - Konfigurationsparameter für Benachrichtigungsstufe".

Anmerkung: Wenn der Konfigurationsparameter **diagsize** auf einen Wert ungleich null gesetzt wird, dann gibt der Wert die Gesamtgröße aller rotierenden Protokolldateien mit Benachrichtigungen für die Systemverwaltung zusammen mit allen rotierenden Protokolldateien der Diagnoseprogramme innerhalb des Diagnosedatenverzeichnisses an. Beispiel: Wenn in einem System mit vier Datenbankpartitionen der Parameter **diagsize** auf 1 GB gesetzt ist, dann kann die Gesamtgröße der Benachrichtigungs- und der Diagnoseprogrammprotokolle zusammen maximal 4 GB (4 x 1 GB) betragen.

Feststellen einer ungeplanten Betriebsunterbrechung

Bevor Sie den Fehler oder Ausfall einer Komponente beheben können, müssen Sie zunächst feststellen, dass die betreffende Komponente fehlgeschlagen ist. DB2 Data Server verfügt über mehrere Tools zum Überwachen des ordnungsgemäßen Zustands einer Datenbank oder ggf. zum Feststellen, dass eine Datenbank fehlgeschlagen ist.

Sie können diese Tools so konfigurieren, dass sie Sie benachrichtigen oder dass sie vordefinierte Maßnahmen ergreifen sollen, wenn sie einen Fehler feststellen.

Vorgehensweise

Mit den beiden folgenden Tools können Sie feststellen, ob ein Fehler in Ihrer DB2-Datenbanklösung aufgetreten ist:

DB2-Fehlermonitorfunktion

Die DB2-Fehlermonitorfunktion sorgt dafür, dass die DB2-Datenbankinstanzen betriebsbereit sind. Wird eine DB2-Datenbankinstanz, der ein DB2-Fehlermonitor zugeordnet ist, unerwartet beendet, startet der DB2-Fehlermonitor die Instanz erneut. Wenn Ihre Datenbanklösung in einem Cluster implementiert ist, sollten Sie die Cluster-Management-Software so konfigurieren, dass fehlgeschlagene Datenbankinstanzen von ihr erneut gestartet werden, statt von dem DB2-Fehlermonitor.

Überwachung durch die Überwachungssignalfunktion in Clusterumgebungen

Die Cluster-Management-Software verwendet Überwachungssignalnachrichten zwischen den Knoten eines Clusters, um den Zustand der Knoten zu überwachen. Der Cluster-Manager stellt fest, dass ein Knoten fehlgeschlagen ist, wenn der Knoten nicht mehr antwortet oder keine Nachrichten mehr sendet.

Überwachung von DB2 High Availability Disaster Recovery-Datenbanken (HADR-Datenbanken)

Die HADR-Funktion hat ihren eigenen Überwachungssignalmonitor. Die Primärdatenbank und die Bereitschaftsdatenbank erwarten voneinander in regelmäßigen Intervallen Überwachungssignalnachrichten.

HADR-Überwachung

Überwachung ist ein integraler Bestandteil beim Einrichten und Verwalten der HADR-Konfiguration. Die DB2-Überwachungsschnittstellen geben Ihnen ein detailliertes Bild der Konfiguration und des Zustands Ihrer Umgebung.

Es stehen verschiedene Verfahren zur Verfügung, die Sie zur Überwachung des Status Ihrer HADR-Datenbanken verwenden können. Folgendes sind die beiden bevorzugten Möglichkeiten der HADR-Überwachung:

- Befehl db2pd
- Tabellenfunktion MON_GET_HADR

Sie können auch die folgenden Methoden verwenden, aber ab Version 10.1 werden sie nicht weiter unterstützt und möglicherweise werden sie in einem zukünftigen Release entfernt:

- Befehl **GET SNAPSHOT FOR DATABASE**
- API 'db2GetSnapshot'
- Verwaltungssicht SNAPHADR
- Tabellenfunktion SNAP_GET_HADR

- Andere verwaltungsbezogene Momentaufnahmesichten und Tabellenfunktionen

Befehl db2pd

Dieser Befehl ruft Informationen aus den DB2-Speichergruppen ab. Sie können diesen Befehl entweder von einer Primär- oder einer Bereitschaftsdatenbank aus absetzen. Wenn Sie den Modus für mehrere Bereitschaftsdatenbanken verwenden und diesen Befehl über eine Bereitschaftsdatenbank absetzen, dann werden keine Informationen zu den anderen Bereitschaftsdatenbanken zurückgegeben. Wenn Sie diesen Befehl auf der Primärdatenbank absetzen, werden Informationenzu zu allen Bereitschaftsdatenbanken zurückgegeben.

Um HADR-Informationen für die Datenbank HADRDB anzuzeigen, könnten Sie den folgenden Befehl absetzen:

```
db2pd -db HADRDB -hadr
```

Angenommen, Sie haben diesen Befehl auf der Primärdatenbank abgesetzt, so könnte dieser Befehl beispielsweise die folgende Ausgabe zurückgeben:

```
Database Member 0 -- Database HADRDB -- Active -- Up 0 days 00:23:17 -- Date 06/08/2011 13:57:23

                           HADR_ROLE = PRIMARY
                       REPLAY_TYPE = PHYSICAL
                     HADR_SYNCMODE = SYNC
                        STANDBY_ID = 1
                     LOG_STREAM_ID = 0
                        HADR_STATE = PEER
                PRIMARY_MEMBER_HOST = hostP.ibm.com
                   PRIMARY_INSTANCE = db2inst
                     PRIMARY_MEMBER = 0
                STANDBY_MEMBER_HOST = hostS1.ibm.com
                   STANDBY_INSTANCE = db2inst
                     STANDBY_MEMBER = 0
                HADR_CONNECT_STATUS = CONNECTED
           HADR_CONNECT_STATUS_TIME = 06/08/2011 13:38:10.199479 (1307565490)
         HEARTBEAT_INTERVAL(seconds) = 25
               HADR_TIMEOUT(seconds) = 100
       TIME_SINCE_LAST_RECV(seconds) = 3
            PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
    LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
   LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                 LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 50772
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87616
           PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
           STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                 HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                   PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                   STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
             STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
        STANDBY_RECV_BUF_SIZE(pages) = 16
            STANDBY_RECV_BUF_PERCENT = 0
          STANDBY_SPOOL_LIMIT(pages) = 0
                PEER_WINDOW(seconds) = 0
           READS_ON_STANDBY_ENABLED = Y
    STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N
```

Tabellenfunktion MON_GET_HADR

Wenn Sie diese Abfrage auf der Primärdatenbank absetzen, werden Informationen zu allen Bereitschaftsdatenbanken zurückgegeben. Wenn Sie die Funktion MON_GET_HADR für eine Bereitschaftsdatenbank absetzen wollen, dann müssen Sie die folgenden Aspekte berücksichtigen:

- In der Bereitschaftsdatenbank müssen Leseoperation aktiviert werden.

- Auch wenn sich Ihre HADR-Konfiguration im Modus für mehrere Bereitschaftsdatenbanken befindet, gibt die Tabellenfunktion keine Informationen zu anderen Bereitschaftseinheiten zurück.

Sie könnten in der Primärdatenbank beispielsweise die folgende Abfrage absetzen:

```
db2 "select HADR_ROLE, STANDBY_ID, HADR_STATE,
            varchar(PRIMARY_MEMBER_HOST,20) as PRIMARY_MEMBER_HOST,
            varchar(STANDBY_MEMBER_HOST,20) as STANDBY_MEMBER_HOST
     from table (mon_get_hadr(NULL))"
```

Das folgende Beispiel zeigt die mögliche Ausgabe:

```
HADR_ROLE  STANDBY_ID  HADR_STATE  PRIMARY_MEMBER_HOST  STANDBY_MEMBER_HOST
---------  ----------  ----------  -------------------  -------------------
PRIMARY             1  PEER        hostP.ibm.com        hostS1.ibm.com

1 Satz/Sätze ausgewählt.
```

Befehl GET SNAPSHOT FOR DATABASE

Dieser Befehl sammelt Statusinformationen und formatiert die Ausgabe. Die zurückgelieferten Informationen stellen eine Momentaufnahme des Betriebsstatus des Datenbankmanagers zum Zeitpunkt der Befehlsausführung dar. HADR-Informationen werden in der Ausgabe unter der Überschrift `HADR-Status` angezeigt.

API 'db2GetSnapshot'

Diese Anwendungsprogrammierschnittstelle (API) sammelt Überwachungsdaten zum Datenbankmanager und schreibt sie in einen dem Benutzer zugeordneten Datenpuffer. Die zurückgelieferten Informationen stellen eine Momentaufnahme des Betriebsstatus des Datenbankmanagers zum Zeitpunkt des API-Aufrufs dar.

Verwaltungssicht SNAPHADR und Tabellenfunktion SNAP_GET_HADR
Diese Verwaltungssicht und diese Tabellenfunktion geben Informationen zu HADR von einer Datenbankmomentaufnahme (insbesondere der logischen HADR-Datengruppe) zurück.

Andere verwaltungsbezogene Momentaufnahmesichten und Tabellenfunktionen
Zum Abfragen eines Unterabschnitts der HADR-Informationen können Sie die folgenden verwaltungsbezogenen Momentaufnahmesichten und Tabellenfunktionen verwenden, die nicht HADR-spezifisch sind und eine größere Menge an Informationen zurückgeben:

- ADMIN_GET_STORAGE_PATHS
- MON_GET_TRANSACTION_LOG
- SNAPDB
- SNAPDB_MEMORY_POOL
- SNAPDETAILLOG
- SNAP_GET_DB
- SNAP_GET_DB_MEMORY_POOL

Handhabung einer ungeplanten Betriebsunterbrechung

Wenn Ihre Datenbankmanagement-Software oder die Software für das Cluster-Management feststellt, dass ein Datenbankserver ausgefallen ist, muss Ihre Datenbanklösung auf diesen Ausfall so schnell und reibungslos wie möglich reagieren.

Ihre Datenbanklösung muss versuchen, die Benutzeranwendungen vor dem Ausfall abzuschirmen, indem Verarbeitungsprozesse umgeleitet werden, falls möglich, und indem die Funktionsübernahme durch eine sekundäre bzw. eine Bereitschaftsdatenbank stattfindet, falls diese verfügbar ist.

Vorgehensweise

Wenn Ihre Datenbank- oder Cluster-Management-Software feststellt, dass ein Datenbankserver ausgefallen ist, müssen Sie oder Ihre Cluster-Management-Software wie folgt vorgehen:

1. Ein sekundärer Datenbankserver muss identifiziert, online geschaltet und initialisiert werden, damit er die Operationen des fehlgeschlagenen Datenbankservers übernehmen kann.

   Wenn Sie DB2 High Availability Disaster Recover (HADR) verwenden, um den primären und den Bereitschaftsdatenbankserver zu verwalten, sorgt HADR dafür, dass die Bereitschaftsdatenbank immer mit der Primärdatenbank synchronisiert bleibt. Ebenso verwaltet HADR die Funktionsübernahme von der Primärdatenbank durch die Bereitschaftsdatenbank.
2. Verarbeitungsprozesse der Benutzeranwendungen müssen an den sekundären Datenbankserver weitergeleitet werden.

   Die DB2-Clientweiterleitung kann Clientanwendungen automatisch von einem fehlgeschlagenen Datenbankserver an einen sekundären Datenbankserver weiterleiten, welcher zuvor zu diesem Zweck identifiziert und konfiguriert worden ist.
3. Die fehlgeschlagenen Datenbankserver müssen aus dem System entfernt und repariert werden.

   Wenn die Benutzeranwendungen an einen sekundären oder einen Bereitschaftsdatenbankserver weitergeleitet wurden, kann der fehlgeschlagene Datenbankserver etwaige Anforderungen von Clientanwendungen erst dann wieder handhaben, wenn er neu gestartet oder anderweitig repariert worden ist. Wenn z. B. die Ursache des Fehlers in der Primärdatenbank das unerwartete Beenden einer Datenbankinstanz war, startet die DB2-Fehlermonitorfunktion die Instanz automatisch neu.

Automatische Clientweiterleitung - Beispiele

Die DB2 Data Server-Clientweiterleitung kann Clientanwendungen automatisch von einem fehlgeschlagenen Datenbankserver an einen sekundären Datenbankserver weiterleiten, welcher zuvor zu diesem Zweck identifiziert und konfiguriert worden ist. Sie können ohne großen Aufwand Clientanwendungen erstellen, die diese DB2 Data Server-Funktionalität testen und veranschaulichen.

Das folgende Beispiel zeigt eine automatische Clientweiterleitung für eine Clientanwendung (nur in Pseudocode dargestellt):

```
int checkpoint = 0;

        check_sqlca(unsigned char *str, struct sqlca *sqlca)
{
   if (sqlca->sqlcode == -30081)
   {
                    // Bei Kommunikationsverlust Anwendung sofort beenden
                    exit(1);
   }
            else

                    // Fehler ausgeben
      printf(...);
```

```
 if (sqlca->sqlcode == -30108)
 {
     // Bei erneutem Verbindungsaufbau fehlgeschlagene Transaktion wiederholen
                          if (checkpoint == 0)
          {
                                  goto checkpt0;
          }
             else if (checkpoint == 1)
          {
                                goto checkpt1;
          }
                          else if (checkpoint == 2)
          {
                                  goto checkpt2;
          }
          ....
                          exit;
 }
 }
}

        main()
{
              connect to mydb;
              check_sqlca("connect failed", &sqlca);

        checkpt0:
              EXEC SQL set current schema XXX;
              check_sqlca("set current schema XXX failed", &sqlca);

   EXEC SQL create table t1...;
              check_sqlca("create table t1 failed", &sqlca);

              EXEC SQL commit;
              check_sqlca("commit failed", &sqlca);

              if (sqlca.sqlcode == 0)
   {
                    checkpoint = 1;
   }

        checkpt1:
              EXEC SQL set current schema YYY;
              check_sqlca("set current schema YYY failed", &sqlca);

   EXEC SQL create table t2...;
              check_sqlca("create table t2 failed", &sqlca);

              EXEC SQL commit;
              check_sqlca("commit failed", &sqlca);

              if (sqlca.sqlcode == 0)
   {
                    checkpoint = 2;
   }
...
}
```

Auf dem Clientsystem wird die Datenbank mit dem Namen „mydb" katalogisiert, die auf einen Knoten „hornet" verweist, wobei „hornet" ebenfalls im Knotenverzeichnis (Hostname „hornet" mit Portnummer 456) katalogisiert wird.

Beispiel mit einer Nicht-HADR-Datenbank

Auf dem Server „hornet" (Hostname ist gleich „hornet" mit einer Portnummer) wird eine Datenbank mit dem Namen „mydb" erstellt. Darüber hinaus wird die Datenbank „mydb" auch auf dem alternativen Server (Hostname „montero" mit Portnummer 456) erstellt. Sie müssen außerdem den alternativen Server für die Datenbank „mydb" auf dem Server „hornet" wie folgt aktualisieren:

```
db2 update alternate server for database mydb using hostname montero port 456
```

In der obigen Beispielanwendung und bei nicht eingerichteter Funktion zur automatischen Clientweiterleitung wird die Anwendung im Fall eines Kommunikationsfehlers in der Anweisung `create table t1` beendet. Bei eingerichteter Funktion zur automatischen Clientweiterleitung versucht der DB2-Datenbankmanager die Verbindung zum Host „hornet" (mit Port 456) erneut herzustellen. Wenn dieser noch nicht wieder funktioniert, versucht der DB2-Datenbankmanager den alternativen Serverstandort (Hostname „montero" mit Port 456). Unter der Voraussetzung, dass es keinen Kommunikationsfehler in der Verbindung zum alternativen Serverstandort gibt, kann die Anwendung anschließend mit der Ausführung nachfolgender Anweisungen fortfahren (und die fehlgeschlagene Transaktion erneut ausführen).

Beispiel mit einer HADR-Datenbank

Auf dem Server „hornet" (Hostname ist gleich „hornet" mit einer Portnummer) wird die Primärdatenbank „mydb" erstellt. Eine Bereitschaftsdatenbank wird außerdem auf Host „montero" mit Port 456 erstellt. Informationen zur Konfiguration von HADR für die Primär- und die Bereitschaftsdatenbank finden Sie in *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz*. Sie müssen außerdem den alternativen Server für die Datenbank „mydb" wie folgt aktualisieren:

```
db2 update alternate server for database mydb using hostname montero port 456
```

In der obigen Beispielanwendung und bei nicht eingerichteter Funktion zur automatischen Clientweiterleitung wird die Anwendung im Fall eines Kommunikationsfehlers in der Anweisung `create table t1` beendet. Bei eingerichteter Funktion zur automatischen Clientweiterleitung versucht das DB2-Datenbanksystem die Verbindung zum Host „hornet" (mit Port 456) erneut herzustellen. Wenn dieser noch nicht wieder funktioniert, versucht das DB2-Datenbanksystem den alternativen Serverstandort (Hostname „montero" mit Port 456). Unter der Voraussetzung, dass es keinen Kommunikationsfehler in der Verbindung zum alternativen Serverstandort gibt, kann die Anwendung anschließend mit der Ausführung nachfolgender Anweisungen fortfahren (und die fehlgeschlagene Transaktion erneut ausführen).

Beispiel mit SSL

Sie können die Clientweiterleitung auch während der gleichzeitigen Verwendung von SSL für Ihre Verbindungen verwenden. Die Konfiguration ähnelt der im vorherigen Beispiel für eine HADR-Datenbank.

Auf dem Clientsystem wird für die Datenbank namens „mydb" der Datenbankaliasname „mydb_ssl" katalogisiert, der auf den Knoten „hornet_ssl" verweist. Der Knoten „hornet_ssl" wird im Knotenverzeichnis (Hostname „hornet", SSL-Portnummer 45678 und Einstellung SSL für den Sicherheitsparameter) katalogisiert.

Auf dem alternativen Server (Hostname „montero", SSL-Portnummer 45678 und Einstellung SSL für den Sicherheitsparameter) wird ebenfalls ein Datenbankaliasna-

me katalogisiert. Außerdem müssen Sie den alternativen Server für den Aliasnamen „mydb_ssl" auf dem Server „hornet" wie folgt aktualisieren:

```
db2 update alternate server for database mydb_ssl using hostname montero port 45678
```

Ändern Sie in der Beispielanwendung oben die Verbindungsanweisung in `connect to mydb_ssl`. Ist die Funktion zur automatischen Clientweiterleitung nicht eingerichtet, wird die Anwendung im Fall eines Kommunikationsfehlers in der Anweisung `create table t1` beendet. Bei eingerichteter Funktion zur automatischen Clientweiterleitung versucht der DB2-Datenbankmanager die Verbindung zum Host „hornet" (mit Port 45678) mithilfe von SSL erneut herzustellen. Wenn dieser noch nicht wieder funktioniert, versucht der DB2-Datenbankmanager, über SSL eine Verbindung zum alternativen Serverstandort (Hostname „montero" mit Port 45678) herzustellen. Unter der Voraussetzung, dass es keinen Kommunikationsfehler in der Verbindung zum alternativen Serverstandort gibt, kann die Anwendung anschließend mit der Ausführung nachfolgender Anweisungen fortfahren (und die fehlgeschlagene Transaktion erneut ausführen).

Ausführen einer HADR-Funktionsübernahmeoperation

Wenn Sie die aktuelle Bereitschaftsdatenbank als neue Primärdatenbank einrichten wollen, da die aktuelle Primärdatenbank nicht verfügbar ist, können Sie eine Funktionsübernahme ausführen.

Informationen zu diesem Vorgang

Warnung:

Diese Prozedur kann zu Datenverlusten führen. Lesen Sie die folgenden Informationen, bevor Sie diese Notfallprozedur ausführen:

- Stellen Sie sicher, dass die Primärdatenbank keine Datenbanktransaktionen mehr verarbeitet. Wenn die Primärdatenbank immer noch aktiv ist, jedoch mit der Bereitschaftsdatenbank nicht mehr kommunizieren kann, könnte eine erzwungene Übernahmeoperation (Befehl **TAKEOVER HADR** mit der Option **BY FORCE**) zu zwei Primärdatenbanken führen. Wenn zwei Primärdatenbanken vorhanden sind, enthalten diese Datenbanken jeweils unterschiedliche Daten, und eine automatische Synchronisierung der Datenbanken ist nicht mehr möglich.
  - Deaktivieren Sie die Primärdatenbank, oder stoppen Sie ihre Instanz, sofern möglich. (Wenn das primäre System blockiert, ausgefallen oder aus anderen Gründen nicht verfügbar ist, ist dies eventuell nicht möglich.) Nach einer Übernahmeoperation übernimmt die ausgefallene Datenbank, wenn sie später erneut gestartet wird, nicht automatisch die Rolle der Primärdatenbank.
- Die Wahrscheinlichkeit und der Umfang eines Transaktionsverlusts hängt von Ihrer spezifischen Konfiguration und den Umständen ab:
  - Wenn die Primärdatenbank ausfällt, während sie sich im Status 'Peer' oder 'Unterbrochener Peer' befindet und der Synchronisationsmodus SYNC (synchron) verwendet wird, gehen in der Bereitschaftsdatenbank keine Transaktionen verloren, die einer Anwendung vor dem Ausfall der Primärdatenbank als festgeschrieben gemeldet wurden.
  - Wenn die Primärdatenbank ausfällt, während sie sich im Status 'Peer' oder 'Unterbrochener Peer' befindet und der Synchronisationsmodus NEARSYNC (fast synchron) verwendet wird, gehen in der Bereitschaftsdatenbank nur dann von der Primärdatenbank festgeschriebene Transaktionen verloren, wenn die Primärdatenbank und die Bereitschaftsdatenbank gleichzeitig ausfallen.

– Wenn die Primärdatenbank ausfällt, während sie sich im Status 'Peer' oder 'Unterbrochener Peer' befindet und der Synchronisationsmodus ASYNC (asynchron) verwendet wird, können in der Bereitschaftsdatenbank von der Primärdatenbank festgeschriebene Transaktionen verloren gehen, wenn die Bereitschaftsdatenbank vor dem Ausführen der Übernahmeoperation nicht alle Protokollsätze empfangen hat. In der Bereitschaftsdatenbank können von der Primärdatenbank festgeschriebene Transaktionen außerdem verloren gehen, wenn die Bereitschaftsdatenbank ausfällt, bevor alle empfangenen Protokolle auf Platte geschrieben werden konnten.

  Anmerkung: Ein Peerfenster ist im Modus ASYNC nicht zulässig; aus diesem Grund weist die Primärdatenbank in diesem Modus niemals den Status 'Unterbrochener Peer' auf.
– Wenn die Primärdatenbank ausfällt, während sie sich im Status 'Fernes Catch-up' befindet und der Synchronisationsmodus SUPERASYNC (super asynchron) verwendet wird, können in der Bereitschaftsdatenbank von der Primärdatenbank festgeschriebene Transaktionen verloren gehen, wenn die Bereitschaftsdatenbank vor dem Ausführen der Übernahmeoperation nicht alle Protokollsätze empfangen hat. In der Bereitschaftsdatenbank können von der Primärdatenbank festgeschriebene Transaktionen außerdem verloren gehen, wenn die Bereitschaftsdatenbank ausfällt, bevor alle empfangenen Protokolle auf Platte geschrieben werden konnten.

  Anmerkung: Datenbanken können im Modus SUPERASYNC niemals den Status 'Peer' oder 'Ünterbrochener Peer' aufweisen. Eine Funktionsübernahme (erzwungene Übernahme) ist im Status 'Fernes Catch-up' nur zulässig, wenn der Synchronisationsmodus SUPERASYNC lautet.
– Wenn die Primärdatenbank ausfällt, während sie sich im Status *Fernes Catch-up anstehend* befindet, gehen Transaktionen, die von der Bereitschaftsdatenbank nicht empfangen und verarbeitet wurden, verloren.

  Anmerkung: Die in der Momentaufnahme der Datenbank angezeigte Protokollabstimmungsdiskrepanz stellt die Diskrepanz zu dem Zeitpunkt dar, an dem die Primärdatenbank und die Bereitschaftsdatenbank zuletzt miteinander kommunizierten. Die Primärdatenbank kann nach diesem Zeitpunkt eine große Anzahl Transaktionen verarbeitet haben.

• Stellen Sie sicher, dass jede Anwendung, die eine Verbindung zur neuen Primärdatenbank herstellt (bzw. durch die Clientweiterleitung an die neue Primärdatenbank weitergeleitet wird), darauf vorbereitet ist, die folgenden Fälle aufzufangen:
  – Bei der Funktionsübernahme kommt es zu Datenverlust. Die neue Primärdatenbank enthält nicht alle Transaktionen, die in der alten Primärdatenbank festgeschrieben wurden. Dies ist möglich, selbst wenn der Konfigurationsparameter **hadr_syncmode** auf SYNC gesetzt ist. Da eine HADR-Bereitschaftsdatenbank Protokolle sequenziell anwendet, können Sie von der Annahme ausgehen, dass, wenn eine Transaktion in einer SQL-Sitzung in der neuen Primärdatenbank festgeschrieben (COMMIT) wird, alle vorherigen Transaktionen in derselben Sitzung ebenfalls in der neuen Primärdatenbank festgeschrieben wurden. Die COMMIT-Folge der Transaktionen über mehrere Sitzungen hinweg lässt sich nur durch eine detaillierte Analyse des Protokolldatenstroms bestimmen.
  – Es ist möglich, dass eine Transaktion an die ursprüngliche Primärdatenbank abgesetzt, in der ursprünglichen Primärdatenbank festgeschrieben und in die neue Primärdatenbank (d. h. die ursprüngliche Bereitschaftsdatenbank) repliziert wird, jedoch nicht als festgeschrieben gemeldet wird, weil die ur-

sprüngliche Primärdatenbank abstürzt, bevor sie an den Client melden kann, dass die Transaktion festgeschrieben wurde. Jede Anwendung, die Sie schreiben, sollte solche Fälle verarbeiten können, in denen Transaktionen, die an die ursprüngliche Primärdatenbank abgesetzt wurden, jedoch nicht als in der ursprünglichen Primärdatenbank festgeschrieben gemeldet werden, in der neuen Primärdatenbank (der ursprünglichen Bereitschaftsdatenbank) festgeschrieben werden.

  - Manche Operationen werden nicht repliziert. Dazu gehören Änderungen an der Datenbankkonfiguration und an externen UDF-Objekten.

- Der Befehl **TAKEOVER HADR** kann nur auf der Bereitschaftsdatenbank abgesetzt werden.
- HADR tauscht keine Daten mit dem DB2-Fehlermonitor (db2fm) aus, der dazu verwendet werden kann, eine ausgefallene Datenbank automatisch erneut zu starten. Wenn der Fehlermonitor aktiviert ist, sollten Sie sich darüber im Klaren sein, dass der Fehlermonitor für eine vermutlich ausgefallene Primärdatenbank wahrscheinlich Aktionen ausführen wird.
- Eine Übernahmeoperation kann nur dann ausgeführt werden, wenn sich die Primärdatenbank und die Bereitschaftsdatenbank im Peerstatus bzw. die Bereitschaftsdatenbank im Status *Fernes Catch-up anstehend* befindet. Befindet sich die Bereitschaftsdatenbank in einem anderen Status, wird ein Fehler zurückgegeben.

  Anmerkung: Sie können eine Bereitschaftsdatenbank, die sich im Status *Lokales Catch-up anstehend* befindet, zur normalen Verwendung zur Verfügung stellen, indem Sie sie in eine Standarddatenbank konvertieren. Fahren Sie die Datenbank dazu mit dem Befehl **DEACTIVATE DATABASE** herunter und setzen Sie anschließend den Befehl **STOP HADR** ab. Nachdem HADR gestoppt wurde, müssen Sie eine aktualisierende Recovery für die frühere Bereitschaftsdatenbank durchführen, bevor sie wieder verwendet werden kann. Eine Datenbank kann nicht wieder in ein HADR-Paar aufgenommen werden, nachdem sie von einer Bereitschaftsdatenbank in eine Standarddatenbank konvertiert wurde. Um HADR auf den beiden Servern erneut zu starten, führen Sie die Prozedur zum Initialisieren von HADR aus.
  Wenn Sie ein Peerfenster konfiguriert haben, beenden Sie die Primärdatenbank, bevor das Fenster abgelaufen ist, um im Falle einer damit zusammenhängenden Funktionsübernahme einen möglichen Transaktionsverlust zu vermeiden.

In einem Funktionsübernahmeszenario kann eine Übernahmeoperation über den Befehlszeilenprozessor (CLP) oder über die Anwendungsprogrammierschnittstelle (API) db2HADRTakeover ausgeführt werden.

Vorgehensweise

Die folgende Prozedur zeigt, wie Sie über den CLP eine Funktionsübernahme in der Primärdatenbank oder der Bereitschaftsdatenbank einleiten:

1. Inaktivieren Sie die fehlgeschlagene Primärdatenbank vollständig. Wenn in einer Datenbank interne Fehler auftreten, kann sie mit normalen Befehlen zum Herunterfahren möglicherweise nicht vollständig inaktiviert werden. Sie müssen gegebenenfalls Betriebssystembefehle verwenden, um Ressourcen wie Prozesse, gemeinsam genutzten Speicher oder Netzverbindungen zu entfernen.
2. Setzen Sie den Befehl **TAKEOVER HADR** mit der Option **BY FORCE** für die Bereitschaftsdatenbank ab. Im folgenden Beispiel wird die Funktionsübernahme für die Datenbank LEAFS ausgeführt:

   ```
   TAKEOVER HADR ON DB LEAFS BY FORCE
   ```

Die Option **BY FORCE** ist erforderlich, da davon auszugehen ist, dass die Primärdatenbank offline ist.

Wenn die Primärdatenbank nicht vollständig inaktiviert ist, ist die Bereitschaftsdatenbank weiterhin mit ihr verbunden und sendet eine Anforderung zum Herunterfahren an die Primärdatenbank. Die Bereitschaftsdatenbank übernimmt unabhängig davon, ob sie eine Bestätigung erhält, dass die Primärdatenbank heruntergefahren wurde, die Rolle der Primärdatenbank.

Wechseln von Datenbankrollen bei High Availability Disaster Recovery (HADR)

Wenn Sie HADR verwenden, können Sie mit dem Befehl **TAKEOVER HADR** die Rollen der Primärdatenbank und der Bereitschaftsdatenbank wechseln.

Informationen zu diesem Vorgang

- Der Befehl **TAKEOVER HADR** kann nur für die Bereitschaftsdatenbank abgesetzt werden. Wenn beim Absetzen des Befehls keine Verbindung zwischen der Primärdatenbank und der Bereitschaftsdatenbank besteht, schlägt die Übernahmeoperation fehl.
- Der Befehl **TAKEOVER HADR** kann nur für einen Rollenwechsel der Primärdatenbank und der Bereitschaftsdatenbank verwendet werden, wenn sich die beiden Datenbanken im Peerstatus befinden. Wenn die Bereitschaftsdatenbank einen anderen Status hat, wird eine Fehlernachricht zurückgegeben.

Vorgehensweise

Gehen Sie wie folgt vor, um die Rollen der HADR-Datenbanken zu wechseln:

- Verwenden Sie den Befehlszeilenprozessor, um eine Übernahmeoperation in der Bereitschaftsdatenbank einzuleiten. Setzen Sie dazu den Befehl **TAKEOVER HADR** ohne die Option **BY FORCE** in der Bereitschaftsdatenbank ab.

  Im folgenden Beispiel wird die Übernahmeoperation für die Bereitschaftsdatenbank LEAFS ausgeführt:

  ```
  TAKEOVER HADR ON DB LEAFS
  ```

  Direkt im Anschluss an eine Übernahmeoperation ist das Auftreten eines Fehlers wegen eines gefüllten Protokolls geringfügig wahrscheinlicher. Um die Wahrscheinlichkeit eines solchen Fehlers zu begrenzen, wird am Ende jeder TAKEOVER-Operation automatisch ein asynchrones Löschen (Flush) des Pufferpools gestartet. Die Wahrscheinlichkeit, dass ein Fehler wegen eines vollen Protokolls auftritt, sinkt mit dem Fortschreiten der asynchronen Flushoperation für den Pufferpool. Wenn darüber hinaus Ihre Konfiguration einen ausreichend großen Speicherbereich für die aktiven Protokolldateien bereitstellt, ist ein Fehler wegen eines vollen Protokolls noch unwahrscheinlicher. Wenn dennoch ein Fehler durch ein volles Protokoll verursacht wird, wird die aktuelle Transaktion abgebrochen und rückgängig gemacht.

  Anmerkung: Durch die Ausführung des Befehls **TAKEOVER HADR** ohne die Option **BY FORCE** wird die Verbindung aller Anwendungen, die zurzeit mit der HADR-Primärdatenbank verbunden sind, zwangsweise getrennt. Diese Aktion ist dafür vorgesehen, in Koordination mit der automatischen Clientweiterleitung die Weiterleitung von Clients an die neue HADR-Primärdatenbank nach einem Rollwechsel zu unterstützen. Wenn jedoch die erzwungene Trennung von Anwendungen von der Primärdatenbank in Ihrer Umgebung zu Unterbrechungen führen kann, ist es für Sie vielleicht sinnvoller, eine eigene Prozedur zu imple-

mentieren, mit der Sie solche Anwendungen vor dem Rollenwechsel beenden und nach erfolgtem Rollenwechsel mit der neuen HADR-Primärdatenbank als Ziel erneut zu starten.

- Rufen Sie die Anwendungsprogrammierschnittstelle (API) db2HADRTakeover über eine Anwendung auf.
- Öffnen Sie den Taskassistenten in IBM Data Studio für den Befehl **TAKEOVER HADR**.

Reintegrieren einer Datenbank nach einer Übernahmeoperation

Wenn in einer HADR-Umgebung die Primärdatenbank fehlschlägt und Sie daher eine Übernahmeoperation ausgeführt haben, können Sie die fehlgeschlagene Datenbank wieder in den Onlinestatus versetzen und entweder als Bereitschaftsdatenbank verwenden oder sie in ihren früheren Status als Primärdatenbank zurückversetzen.

Vorgehensweise

Gehen Sie wie folgt vor, um die fehlgeschlagene Primärdatenbank als neue Bereitschaftsdatenbank in das HADR-Paar zu integrieren:

1. Reparieren Sie das System, auf dem sich die ursprüngliche Primärdatenbank befand. Dies kann eine Reparatur von ausgefallener Hardware oder einen Neustart des fehlgeschlagenen Betriebssystems umfassen.
2. Starten Sie die fehlgeschlagene Primärdatenbank als Bereitschaftsdatenbank. Im folgenden Beispiel wird die Datenbank LEAFS als Bereitschaftsdatenbank gestartet:

   ```
   START HADR ON DB LEAFS AS STANDBY
   ```

   Anmerkung: Die Reintegration schlägt fehl, wenn die Protokolldatenströme der beiden Datenbankkopien nicht kompatibel sind. Insbesondere setzt HADR voraus, dass die ursprüngliche Primärdatenbank keine protokollierten Operationen anwendet hat, die in der ursprünglichen Bereitschaftsdatenbank noch nicht nachvollzogen waren, bevor diese die Rolle als neue Primärdatenbank übernahm. War dies der Fall, können Sie die ursprüngliche Primärdatenbank als Bereitschaftsdatenbank neu starten, indem Sie ein Backup-Image der neuen Primärdatenbank wiederherstellen oder eine geteilte Spiegeldatenbank initialisieren.

   Eine Erfolgsmeldung dieses Befehls bedeutet nicht, dass die Integration erfolgreich war. Sie bedeutet lediglich, dass die Datenbank gestartet wurde. Die Reintegration befindet sich noch in der Ausführung. Wenn die Reintegration nachfolgend fehlschlägt, fährt sich die Datenbank selbst herunter. Sie sollten die Statuszustände der Bereitschaftsdatenbank mithilfe des Befehls **GET SNAPSHOT FOR DATABASE** oder des Tools **db2pd** überwachen, um sicherzustellen, dass die Bereitschaftsdatenbank online verbleibt und mit den normalen Statusübergängen fortfährt. Falls erforderlich, können Sie den Status der Datenbank anhand der Protokolldatei mit Benachrichtigungen für die Systemverwaltung und der Protokolldatei **db2diag** feststellen.

Nächste Schritte

Nachdem die ursprüngliche Primärdatenbank dem HADR-Paar als Bereitschaftsdatenbank erneut hinzugefügt wurde, können Sie eine Zurücksetzungsoperation ausführen, um die Rollen der Datenbanken wieder zu tauschen, sodass die ursprüngli-

che Primärdatenbank erneut als aktuelle Primärdatenbank verwendet werden kann. Setzen Sie den folgenden Befehl für die Bereitschaftsdatenbank ab, um diese Zurücksetzungsoperation auszuführen:

```
TAKEOVER HADR ON DB LEAFS
```

Anmerkung:

1. Wenn sich die HADR-Datenbanken nicht im Peerstatus befinden oder keine Verbindung zwischen ihnen besteht, schlägt dieser Befehl fehl.
2. Geöffnete Primärdatenbanksitzungen werden zwangsweise geschlossen, und unvollständige Transaktionen werden rückgängig gemacht.
3. Wenn Sie die Rollen der Primär- und der Bereitschaftsdatenbank tauschen, kann die Option **BY FORCE** des Befehls **TAKEOVER HADR** nicht angegeben werden.

Kapitel 6. Fehlerverwaltung mit DB2-Cluster-Services

Dieser Abschnitt enthält eine Übersicht über die DB2-Cluster-Services und ausführliche Informationen dazu, wie sie bei bestimmten Ausfällen von Hosts, Cluster-Caching-Funktionen und Membern verfahren.

Bei den DB2-Cluster-Services handelt es sich um eine Software, die eine automatische Fehlererkennung durch Überwachungssignale (Heartbeats) bereitstellt und automatisch die erforderlichen Recoveryoperationen einleitet, wenn ein Fehler erkannt wird. Sie stellen außerdem das Clusterdateisystem zur Verfügung, das jedem Host in einer DB2 pureScale-Instanz den Zugriff auf ein gemeinsames Dateisystem ermöglicht. Die DB2-Cluster-Services enthalten Technologie aus der Software von IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP), der Software von IBM Reliable Scalable Clustering Technology (RSCT) und der Software von IBM General Parallel File System (GPFS). Diese Technologie ist als integraler Bestandteil im Umfang von DB2 pureScale Feature enthalten.

Automatisierte Funktionsübernahme für Cluster-Caching-Funktion

Wenn die primäre Cluster-Caching-Funktion (CF) ausfällt, versuchen die DB2-Cluster-Services automatisch, diese erneut zu starten, und übertragen die primäre Rolle auf die sekundäre Cluster-Caching-Funktion (bei der empfohlenen Konfiguration mit zwei Cluster-Caching-Funktionen).

Aufgrund der zentralen Rolle, die die Cluster-Caching-Funktion für eine DB2 pureScale-Instanz spielt, wird die fehlgeschlagene Cluster-Caching-Funktion möglichst schnell erneut gestartet und wieder eingegliedert. Die Auswirkungen des Ausfalls hängen von folgenden Faktoren ab:

- Wie viele Cluster-Caching-Funktionen sind Bestandteil der DB2 pureScale-Instanz?

  Wenn nur eine Cluster-Caching-Funktion in der DB2 pureScale-Instanz enthalten ist, wird die Instanz heruntergefahren. Wenn die Cluster-Caching-Funktion aufgrund eines Softwarefehlers ausgefallen ist, wird automatisch ein Gruppenneustart eingeleitet. Wenn die Cluster-Caching-Funktion aufgrund eines Hardwarefehlers ausgefallen ist, muss der Benutzer das Problem beheben. Im Anschluss daran wird automatisch ein Gruppenneustart eingeleitet.

  Wenn zwei Cluster-Caching-Funktionen in der DB2 pureScale-Instanz enthalten sind (dies ist die empfohlene Konfiguration), versuchen die DB2-Cluster-Services, die primäre Rolle an die sekundäre Cluster-Caching-Funktion zu übergeben. Wenn die primäre Cluster-Caching-Funktion aufgrund eines Softwarefehlers ausgefallen ist, wird sie automatisch erneut gestartet und als sekundäre Cluster-Caching-Funktion wieder eingegliedert. Wenn die primäre Cluster-Caching-Funktion aufgrund eines Hardwarefehlers ausgefallen ist, muss der Benutzer das Problem beheben. Anschließend wird die Funktion automatisch erneut gestartet und als sekundäre Cluster-Caching-Funktion wieder eingegliedert.
- Status der sekundären Cluster-Caching-Funktion beim Ausfall der primären Cluster-Caching-Funktion

  Wenn sich die sekundäre Cluster-Caching-Funktion im PEER-Status befindet, übertragen die DB2-Cluster-Services die primäre Rolle auf sie.

  Befindet sich die sekundäre Cluster-Caching-Funktion nicht im PEER-Status, wird die Instanz inaktiv. In diesem Fall leiten die DB2-Cluster-Services einen

Gruppenneustart ein, wobei die vorherige sekundäre Cluster-Caching-Funktion nun in der primären Rolle gestartet wird.

Automatisierter Neustart

In einer DB2 pureScale-Umgebung erkennen die DB2-Cluster-Services automatisch Software- und Hardwarefehler und initialisieren einen Memberneustart oder einen Gruppenneustart - abhängig von der Art des Fehlers, der aufgetreten ist.

Der automatisierte Neustart trägt dazu bei sicherzustellen, dass sich Fehler bei Host-Member oder Cluster-Caching-Funktion nur minimal auf die Datenbank auswirken. Ein Memberfehler (und der nachfolgende Neustart) ist für Anwendungen transparent und wirkt sich nur zeitweilig auf nicht festgeschriebene Transaktionen aus, die auf dem ausgefallenen Member ausgeführt werden. Anwendungen können weiterhin auf Datenbanken zugreifen, auch wenn mehrere Host- oder Memberfehler aufgetreten sind. In Situationen, in denen die DB2-Cluster-Services das ausgefallene Member nicht auf seinem ursprünglichen Host erneut starten können, wird das ausgefallene Member auf einem anderen Host erneut gestartet. Dieser Prozess wird als *Light-Neustart* bezeichnet. Extreme Störungen, wie wie z. B. der Ausfall aller Cluster-Caching-Funktionen in der DB2 pureScale-Instanz, führen zu einem Datenbankausfall. In solchen Fällen leiten die DB2-Cluster-Services einen Gruppenneustart der Cluster-Caching-Funktionen und der Member ein.

Neustart und Recovery nach Absturz eines Members

Ein *Memberneustart* ist ein Prozess, bei dem die Datenbankserverprozesse auf einem ausgefallenen Member erneut gestartet werden und eine Recovery nach Absturz eines Members für jede Datenbank durchgeführt wird, für die dies erforderlich ist.

Wenn ein Software- oder Hardwarefehler auf einem Host zum Ausfall eines Members führt, erkennen die DB2-Cluster-Services den Ausfall und starten das Member automatisch neu. Der Memberneustart kann entweder ein *lokaler Neustart*, das heißt, das Member wird auf dem ursprünglichen Host (*Benutzerhost*) erneut gestartet, oder ein *Light-Neustart* sein, wenn das Member auf einem anderen Host erneut gestartet wird.

Lokaler Neustart
Wenn ein Member aufgrund eines Softwarefehlers ausfällt, der Benutzerhost des Members jedoch aktiv bleibt, versuchen die DB2-Cluster-Services, einen lokalen Neustart auszuführen. Der lokale Memberneustart verwendet ein reduziertes Speichermodell, um sicherzustellen, dass die in Verarbeitung befindlichen Daten schnell in einem konsistenten Zustand wiederhergestellt werden. Wenn die in Verarbeitung befindlichen Daten wiederhergestellt wurden, wird das normale Speichermodell der Datenbank initialisiert, um die vollständige Transaktionsverarbeitung wieder aufzunehmen.

Light-Neustart
Wenn der Benutzerhost eines Members nicht aktiv ist oder ein lokaler Neustartversuch fehlschlägt, wird das Member automatisch als *Gastmember* auf dem Benutzerhost eines anderen Members unter Verwendung minimaler Ressourcen erneut gestartet. Ein Member, das im Light-Neustartmodus aktiv ist, verarbeitet keine neuen Transaktionen, da sein einziger Zweck darin besteht, eine Recovery nach Absturz eines Members auszuführen.

Mehrere Memberausfälle sind im Allgemeinen durch unabhängig voneinander und gleichzeitig ausgeführte Neustartrecoverys für Member behebbar, sodass ein Gruppenneustart gewöhnlich nicht erforderlich wird. Die Datenbank bleibt über andere

verbleibende Member für den Zugriff geöffnet. Lediglich Daten, die auf den ausgefallenen Membern in Bearbeitung waren, sind für die Dauer der Memberneustarts nicht verfügbar.

Recovery nach Absturz eines Members

Die Recovery nach Absturz eines Members ist im Rahmen eines Memberneustarts dafür zuständig, einen Rollback der unvollständigen Transaktionen durchzuführen und festgeschriebene Transaktionen abzuschließen. Dadurch wird sichergestellt, dass die von dem betreffenden Member geänderten Datenbankdaten in einen konsistenten Zustand gebracht werden. Wenn am Ende einer Recovery nach Absturz eines Members unbestätigte Transaktionen vorhanden sind, ist das Member verfügbar, sodass sie aufgelöst werden können.

Eine Recovery nach Absturz eines Members wird ausgeführt, wenn weiterhin eine funktionsfähige Cluster-Caching-Funktion verfügbar ist, die Datenbank auf einem Member aktiviert ist und festgestellt wird, dass der Protokolldatenstrom auf der Platte aufgrund einer abnormalen Beendigung inkonsistent ist. In den meisten Fällen wird die Recovery nach Absturz eines Members automatisch durch einen Memberneustart und den Agenten für automatische Recovery aufgerufen. Der Agent für automatische Recovery, der beim Initialisieren der Instanz gestartet wird, wird dann aktiv, wenn er feststellt, dass eine Datenbank des Members inkonsistent ist.

Wenn die Recovery nach Absturz eines Members für eine Datenbank abgeschlossen ist, ist das Member in der Lage, eingehende Verbindungsanforderungen von anderen Anwendungen zu akzeptieren, sofern dieses Member auf seinem ursprünglichen Host gestartet wurde.

Neustart und Recovery nach Absturz einer Gruppe

Ein *Gruppenneustart* ist der Prozess, bei dem die gesamte DB2 pureScale-Instanz erneut gestartet wird, indem die Datenbankserverprozesse für alle Member und Cluster-Caching-Funktionen erneut gestartet werden und eine Recovery nach dem Absturz einer Gruppe durchgeführt wird, um die Datenbank wieder online verfügbar zu machen.

Ein Gruppenneustart findet statt, wenn keine funktionsfähige primäre Cluster-Caching-Funktion in der DB2 pureScale-Instanz vorhanden ist. Dieses Ereignis wird automatisch erkannt und von den DB2-Cluster-Services behandelt. Ein Gruppenneustart wird automatisch eingeleitet, sobald eine primäre Cluster-Caching-Funktion und ein Member verfügbar werden. Beim Gruppenneustart besteht bei allen Membern kein Zugriff auf die Datenbank.

Es gibt nur wenige Situationen, in denen ein Gruppenneustart erforderlich wird:

- Wenn die Instanz mit nur einer Cluster-Caching-Funktion ausgeführt wird und diese Cluster-Caching-Funktion ausfällt.
- Wenn die primäre Cluster-Caching-Funktion ausfällt, bevor die sekundäre Cluster-Caching-Funktion den PEER-Status erreicht hat.
- Wenn beide Cluster-Caching-Funktionen ausfallen.

Recovery nach Absturz einer Gruppe

Eine Recovery nach dem Absturz einer Gruppe hat die Aufgabe, die Datenbank konsistent zu machen, indem sie Operationen, die noch nicht auf Platte geschrieben wurden, erneut ausführt und alle Transaktionen, die zum Zeitpunkt des Fehlers nicht festgeschrieben waren, durch einen Rollback rückgängig macht. Eine Re-

covery nach dem Absturz einer Gruppe entspricht etwa der Recovery nach einem DB2-Absturz außerhalb einer DB2 pureScale-Umgebung, aber sie verwendet die kombinierten Datenströme aller Member, die bei der Datenbank aktiv sind. Weil die Recovery nach dem Absturz einer Gruppe automatisch im Rahmen eines Gruppenneustarts erfolgt, müssen die Benutzer im Allgemeinen keine Maßnahmen ergreifen, wenn eine Recovery nach dem Absturz einer Gruppe bei Vorhandensein eines funktionierenden Cluster-Managers erforderlich ist.

Light-Neustart

Wenn ein ausgefallenes Member auf seinem ursprünglichen Host oder *Benutzerhost* nicht erneut gestartet werden kann, starten die DB2-Cluster-Services das Member auf einem der anderen verfügbaren Hosts in der DB2 pureScale-Instanz. Dieser Prozess, der als *Light-Neustart* bezeichnet wird, ermöglicht das Ausführen einer Recovery nach dem Absturz eines Members, ohne dass dadurch der übrige Teil der Instanz signifikant beinträchtigt wird.

Ein Member, das im Light-Neustartmodus auf einem anderen Host gestartet wurde, wird als *Gastmember* bezeichnet, während ein Member, das auf seinem Benutzerhost ausgeführt wird, als *residentes Member* bezeichnet wird. Ein Gastmember nutzt weniger Ressourcen als ein residentes Member und es kann keine Instanzverbindungen oder Datenbankverbindungen von externen Anwendungen akzeptieren (SQL1032N).

Der einzige Zweck, zu dem ein Member im Light-Neustartmodus gestartet wird, besteht in der Durchführung einer Recovery nach einem Absturz eines Members. Das Gastmember wird mithilfe eines vorab zugeordneten reduzierten Speichermodells gestartet, um den Einfluss auf das residente Member zu minimieren, dessen Host für den Light-Neustart verwendet wird. Wenn das Gastmember, das im Light-Neustartmodus ausgeführt wird, die Recovery nach dem Absturz eines Members für alle erforderlichen Datenbanken abgeschlossen hat, wartet es darauf, dass es auf seinen Benutzerhost zurückübertragen wird. Es kann erst wieder neue Transaktionen verarbeiten, wenn diese Rückübertragung (Failback) auf den Benutzerhost erfolgt ist. Beachten Sie, dass ein Member im Light-Neustartmodus anscheinend betriebsbereit ist (sich aber im Status WAITING_FOR_FAILBACK befindet), wenn sein Status abgefragt wird. Wenn der ausgefallene Benutzerhost wieder online verfügbar ist, wird die Rückübertragung des Members im Light-Neustartmodus auf seinen Benutzerhost automatisch durch die DB2-Cluster-Services ausgeführt. Dadurch wird es zu einem vollständig aktiven Member, das wieder neue Transaktionen verarbeiten kann.

Der Light-Neustart ist ein automatisierter Prozess, der wie folgt abläuft: Bei Softwarefehlern, bei denen der Benutzerhost des Members weiterhin aktiv bleibt, versuchen die DB2-Cluster-Services, ein ausgefallenes Member auf seinem Benutzerhost erneut zu starten. Wenn der erste Versuch, das ausgefallene Member erneut zu starten, auf seinem Benutzerhost nicht erfolgreich ist, wird dieses Member als Gastmember auf einem anderen Host im Light-Modus erneut gestartet. Der Light-Neustart wird sofort für ein Member initialisiert, wenn eine der folgenden Situationen eintritt:

- Es tritt ein Netzausfall ein, bei dem der Benutzerhost die Verbindung zum Rest der DB2 pureScale-Instanz verliert.
- Der Benutzerhost wird durch einen Stromausfall oder durch das Zurücksetzen einer Hardwarekomponente, einer logischen Partition (LPAR), eines VM oder des Betriebssystems unerwartet inaktiv.
- Das Member wird abnormal beendet, wenn sein Benutzerhost für Wartungsoperationen gestoppt wird.

Ein Light-Neustart wird sehr schnell ausgeführt, da bestimmte DB2-Leerlaufprozesse auf jedem Host Ressourcen für den Neustart von Gastmembern vorab zuordnen. DB2 aktiviert diese Prozesse, anstatt neue Prozesse zu erstellen, um Member im Light-Modus erneut zu starten. Da während des Light-Neustarts keine neuen Prozesse erstellt werden und weil das Gastmember nicht mit dem residenten Member um Ressourcen konkurriert, wird die Ausführung der Member-Recovery beschleunigt.

Inaktivieren der automatischen Rückübertragung von Membern

In einigen Situationen möchten Sie möglicherweise die automatische Rückübertragung von Membern verzögern. Dazu steht der Befehl **db2cluster** zur Verfügung.

Informationen zu diesem Vorgang

Standardmäßig führt DB2-Cluster-Services eine Rückübertragung aller Member durch, die im Light-Neustartmodus ausgeführt werden, sobald der Benutzerhost des entsprechenden Members verfügbar wird und alle Alerts behoben wurden. Wenn Sie die Ursache für den Hostausfall untersuchen möchten, kann es von Vorteil sein, die automatische Rückübertragung von Membern bis auf Weiteres zu inaktivieren.

Vorgehensweise

Gehen Sie folgendermaßen vor, um die automatische Rückübertragung zu inaktivieren:

1. Setzen Sie den Befehl **db2cluster** ab:
   ```
   db2cluster -cm -set -option autofailback -value off
   ```
2. Führen Sie einen Neustart der DB2 pureScale-Instanz durch.

Ergebnisse

Alle Member, die sich im Light-Neustartmodus befinden, sowie alle zukünftigen Member in diesem Modus verbleiben selbst dann auf Ihrem Gasthost, wenn alle Member-Alerts behoben wurden und der residente Host des Members aktiv ist.

Beispiel

Nach einem wiederholten Hostausfall, der dazu führte, dass Member im Light-Modus neu gestartet wurden, entscheidet sich der Datenbankadministrator, alle fehlgeschlagenen Hosts so lange frei von ihren residenten Membern zu halten, bis der Grund für den Ausfall ermittelt werden konnte. Der Datenbankadministrator setzt den folgenden Befehl ab:

```
db2cluster -cm -set -option autofailback -value off
```

Es wird folgende Nachricht zurückgegeben:

```
Der Befehl 'db2cluster' war erfolgreich. Die Option AUTOFAILBACK wurde auf "OFF" gesetzt.
Beim nächstfolgenden Start der DB2-Datenbankmanagerinstanz wird die automatische
Rückübertragung inaktiviert.
```

Anmerkung: Wenn die automatische Rückübertragung aus einem bestimmten Grund bereits inaktiviert wurde, würde der Befehl erfolgreich abgeschlossen, jedoch würde in der Nachricht nicht der Neustart der Instanz erwähnt.

Der Datenbankadministrator startet die Instanz neu. Nachdem er die Ursache für den Hostausfall überprüft hat, setzt der Datenbankadministrator den folgenden befehl ab, um die automatische Rückübertragung von Membern zu aktivieren:

```
db2cluster -cm -set -option autofailback on
```

Es wird folgende Nachricht zurückgegeben:

```
Der Befehl 'db2cluster' war erfolgreich. Die Option AUTOFAILBACK wurde auf "ON" gesetzt.
Beim nächstfolgenden Start der DB2-Datenbankmanagerinstanz wird die automatische
Rückübertragung aktiviert.
```

Nach dem Neustart der Instanz stellt der Datenbankadministrator mit dem folgenden Befehl sicher, dass die automatische Rückübertragung von Membern aktiviert ist:

```
db2cluster -cm -list -option -autofailback
```

Es wird folgende Nachricht zurückgegeben:

```
Die Option AUTOFAILBACK wurde auf "ON" gesetzt.
```

Hinweise zum Hauptspeicher für den Light-Neustart

Eine begrenzte Hauptspeichermenge wird beim Start von DB2 für Recoveryzwecke reserviert, um die Recovery von Membern im Light-Neustartmodus zu erleichtern. Dieser reservierte Speicher für den Light-Neustart wird im Voraus definiert. Dies verbessert die Recoveryleistung, da der Speicher reserviert ist und bei Bedarf direkt für die Recovery zur Verfügung steht.

Die Speicherkapazität, die zu Zwecken einer Recovery mit Light-Neustart auf einem Host reserviert werden kann, wird durch den Konfigurationsparameter des Datenbankmanagers *rstrt_light_mem* begrenzt. Der Standardwert von *rstrt_light_mem* ist AUTOMATIC. Dies bedeutet, dass DB2 automatisch eine feste obere Grenze für die Speichermenge berechnet, die vorab zugeordnet und für Zwecke der Recovery mit Light-Neustart reserviert wird. Der Wert wird beim Start von DB2 eingestellt. DB2 berechnet den Wert auf der Basis der Einstellungen für die Konfigurationsparameter *instance_memory* und *numdb* und der Anzahl von Membern auf dem Host. Der automatisch berechnete Wert beträgt zwischen 1 und 10 Prozent der Instanzspeicherbegrenzung und ist in der Gesamtgröße des Instanzspeichers enthalten. Da sich die Größe des für den Light-Neustart reservierten Speichers jedoch negativ auf die Leistung eines residenten Members auswirken kann, können Benutzer die Konfiguration für den Light-Neustartspeicher entsprechend der jeweiligen Auslastung anpassen.

Anzeigen des für den Light-Neustart reservierten Speichers

Informationen zur Gesamtspeichermenge, die auf einem DB2-Host zugeordnet ist, können Sie mit dem Befehl **db2pd** unter Angabe der Option `-totalmem` abrufen. Diese Informationen beinhalten die für den Light-Neustart reservierte Speichermenge, die im Voraus auf dem aktuellen im Zugriff befindlichen DB2-Host zugeordnet wird Zum Abrufen von Informationen zu allen Hosts im Cluster führen Sie den Befehl 'db2pd' auf separaten Hosts parallel aus. Im folgenden Beispiel wird der Befehl 'db2pd' auf Host B mit Member 20 ausgeführt.

```
db2pd -totalmem

                        Controller     Memory        Current        HWM           Cached
                        Automatic      Limit         Usage          Usage         Memory
Member 20               Yes            25750080 KB   9031201 KB     9391744 KB    480064 KB
 Restart Light Memory   Yes            2575008 KB    64182 KB       69265 KB      5250 KB

Total current usage: 9095383 KB
Total cached memory:  485314 KB
```

Recovery verdeckter Pufferpools

Zur Recovery nach dem Absturz eines Members wird ein reduziertes Speichermodell für die Pufferpools verwendet. Da die Pufferpools normalerweise den größten Teil des Speichers im gemeinsam genutzten Datenbankspeicher verbraucht, ist die Zuordnung umfangreicher Pufferpools sehr zeitaufwendig. Das reduzierte Speichermodell verbessert die Recoveryleistung, weil kleine verdeckte Recoverypufferpools anstatt großer benutzerdefinierter Pufferpools zugeordnet werden, die sehr aufwendig sind. Ebenso wie bei den vorhandenen verdeckten Pufferpools werden vier verdeckte Recoverypufferpools, d. h. einer für jede Größe von 4K, 8K, 16K und 32K, zugeordnet. Die verdeckten Pufferpools haben aber immer eine Größe von 16 Seiten; die verdeckten Recoverypufferpools haben eine Mindestgröße von 250 Seiten und können auch größer sein - abhängig von der Größe des Light-Neustartspeichers und der Pufferpoolgrößenberechnungen.

Im folgenden Beispiel wurden zwei Pufferpools, BP1 mit 100 Seiten und BP2 mit 200 Seiten, für die Datenbank TESTDB erstellt. Member 0 befindet sich im Light-Neustartmodus. Member 1 befindet sich nicht im Light-Neustartmodus. Das Beispiel enthält einen Teil der Ausgabe aus dem nachfolgenden Befehl 'db2pd'. Member 1 zeigt die benutzererstellten Pufferpools und die verdeckten Pufferpools, auch wenn Member 0 nur die vier verdeckten Recoverypufferpools zeigt.

```
db2pd -allmembers -db testdb -bufferpools

Database Member 1--Database TESTDB--Active--Up 0 days 00:00:14--Date 08/12/2010 18:55:19

Bufferpools:
First Active Pool ID      1
Max Bufferpool ID         3
Max Bufferpool ID on Disk 3
Num Bufferpools           7

Address            Id   Name               PageSz     PA-NumPgs  BA-NumPgs  BlkSize
0x00002AAADE443140 1    IBMDEFAULTBP       4096       1000       0          0
0x00002AAADE45B080 2    BP1                4096       100        0          0
0x00002AAADE45F060 3    BP2                4096       200        0          0
0x00002AAADDB83CC0 4096 IBMSYSTEMBP4K      4096       16         0          0
0x00002AAADDB13CC0 4097 IBMSYSTEMBP8K      8192       16         0          0
0x00002AAADDB03CC0 4098 IBMSYSTEMBP16K     16384      16         0          0
0x00002AAADE453140 4099 IBMSYSTEMBP32K     32768      16         0          0

...NumTbsp    PgsToRemov CurrentSz  PostAlter  SuspndTSCt Automatic
   3          0          1000       1000       0          False
   0          0          100        100        0          False
   0          0          200        200        0          False
   0          0          16         16         0          False
   0          0          16         16         0          False
   0          0          16         16         0          False
   0          0          16         16         0          False


Database Member 0 -- Database TESTDB -- Active -- Up 0 days 00:00:13

Bufferpools:
First Active Pool ID      4096
Max Bufferpool ID         0
Max Bufferpool ID on Disk 3
Num Bufferpools           4

Address            Id   Name               PageSz     PA-NumPgs  BA-NumPgs  BlkSize
0x00002AAAD9F946E0 4096 IBMSYSTEMBP4K      4096       9954       0          0
0x00002AAADA743140 4097 IBMSYSTEMBP8K      8192       250        0          0
0x00002AAADA733140 4098 IBMSYSTEMBP16K     16384      250        0          0
0x00002AAADA74B080 4099 IBMSYSTEMBP32K     32768      250        0          0

...NumTbsp    PgsToRemov CurrentSz  PostAlter  SuspndTSCt Automatic
   3          0          9954       9954       0          False
   0          0          250        250        0          False
   0          0          250        250        0          False
   0          0          250        250        0          False
```

Speicherbelegung während eines Light-Neustarts

Idealerweise ermöglicht ein Neustart die sofortige Recovery eines ausgefallenen Members auf einem anderen Host als dem Benutzerhost dieses Members - und dies ohne Auswirkungen auf das residente Member dieses Hosts. Zu diesem Zweck wird der für den Light-Neustart reservierte Recoveryspeicher zuerst dazu verwendet, Operationen der Datenbankrecovery auszuführen. Wenn die Datenbankrecovery jedoch Speicherressourcen benötigt, die den für den Light-Neustart zugeordneten Speicher übersteigen, fordert der Light-Neustart zusätzlich freien Instanzspeicher an. Diese kritischen Speicheranforderungen versuchen, die aktuelle Speicherbelegung durch das residente Member zu verringern. Wenn noch immer nicht ausreichend Speicher zur Verfügung steht, um den Light-Neustart abzuschließen, fordert DB2 zusätzlichen Speicher vom Betriebssystem an. Wenn dieser Fall eintritt, schlagen alle anderen nicht kritischen Speicheranforderungen fehl, bis genügend Speicher für die Recoveryoperation freigegeben wurde. Auch wenn Anwendungen, die auf dem residenten Member ausgeführt werden, wegen nicht ausreichendem Speicher fehlschlagen, bleibt das residente Member aktiv. Wenn die Recovery abgeschlossen ist und die Datenbank(en) konsistent ist/sind, wird der gesamte zusätzliche Speicher freigegeben, der vom Gastmember über den ursprünglich reservierten Recoveryspeicher hinaus verwendet wird.

Solche Anforderungen von zusätzlichen Speicherressourcen für einen Light-Neustart sind temporär, sie können sich aber negativ auf die Arbeitslast des residenten Members auswirken. Wenn Sie feststellen, dass der reservierte Recoveryspeicher nicht ausreicht, ziehen Sie eine Erhöhung des Werts des Konfigurationsparameters des Datenbankmanagers *rstrt_light_mem* in Betracht. Der Parameter ist konfigurierbar, aber nicht online. D. h., Änderungen erfordern einen globalen Befehl **db2stop** und **db2start** oder - wenn Sie *rstrt_light_mem* auf der Basis des einzelnen Members aktualisieren möchten (d. h. wenn Sie nicht alle Member gleichzeitig stoppen möchten) - Sie müssen jedes Member und die Instanz auf dem Host jedes Members wie folgt stoppen und wieder starten:

```
db2 update dbm cfg using RSTRT_LIGHT_MEM 5

db2stop member 10
db2stop instance on hostA.torolab.ibm.com
db2start instance on hostA.torolab.ibm.com
db2start member 10

db2stop member 20
db2stop instance on hostB.torolab.ibm.com
db2start instance on hostB.torolab.ibm.com
db2start member 20
```

Anzeigen der Speicherbelegung für den Light-Neustart

Es gibt zwei Möglichkeiten, Informationen zu der Gesamtspeichermenge anzuzeigen, die auf einem Host von den residenten Membern und von den Gastmembern genutzt wird:

1. Sie können den Befehl **db2pd** mit der Option `'-totalmem'` auf jedem Host ausführen. Betrachten Sie das folgende Beispiel:
   - Member 0 auf Host A
   - Member 1 auf Host B
   - Member 2 auf Host C

   Member 0 wird vom Host B im Light-Neustartmodus übernommen. Der Benutzer führt den Befehl **db2pd** auf Host B mit dem residenten Member 1 aus, da das Gastmember 0 im Light-Neustartmodus ausgeführt wird. Anschließend

führt der Benutzer den Befehl **db2pd** auf Host C aus, um den Speicher für Member 2 anzuzeigen. Das Member 0 wird in der Ausgabe durch "(guest)" gekennzeichnet und die Speicherbelegung wird in Kilobyte angegeben. Für den Befehl 'db2pd' ist keine Datenbankverbindung erforderlich.

Host B:

```
$ db2pd -totalmem
Total Memory Statistics in KB

                     Controller    Memory     Current       HWM       Cached
                     Automatic     Limit       Usage       Usage      Memory
=============================================================================+=====
Member 0 (guest)         Yes       20677572      242496      244032       15168
Member 1                 Yes       20677572      186624      697088       46080
Restart Light Memory     Yes         839720      459392      459392       19200

Total current usage:  888512
Total cached memory:  80448
```

Host C:

```
$ db2pd -totalmem
Total Memory Statistics in KB

                     Controller    Memory     Current       HWM       Cached
                     Automatic     Limit       Usage       Usage      Memory
=============================================================================+=====
Member 2                 Yes       20153284     4728832     4728832     1055104
Restart Light Memory     Yes         839720      689088      689088       28800

Total current usage:  5417920
Total cached memory:  1083904
```

2. Sie können die SQL-Schnittstelle verwenden, um die Speicherbelegung für alle Member, einschließlich Member im Light-Neustartmodus, anzuzeigen. Sie SQL-Anweisung muss nur über einen Host ausgeführt werden. Für diese Methode ist jedoch eine Datenbankverbindung erforderlich. Daher kann sie nicht von einem Member im Light-Neustartmodus aus ausgeführt werden, da Member im Light-Neustartmodus keine Verbindungen akzeptieren. Diese Anzeige kennzeichnet Member nicht als Gast ("guest") und die Speicherbelegung wird in Byte angegeben.

```
$ db2 'SELECT * FROM TABLE (SYSPROC.ADMIN_GET_MEM_USAGE()) AS T'

DBPARTITIONNUM MAX_PARTITION_MEM    CURRENT_PARTITION_MEM PEAK_PARTITION_MEM
-------------- -------------------- --------------------- --------------------
             1          21173833728            4605345792           4605345792
             0          21173833728             248840192            249888768
             2          20636962816            4651352064           4651352064

  3 Satz/Sätze ausgewählt.
```

Überwachen von Membern beim Light-Neustart

Sie können den Fortschritt einer Recovery eines Members im Light-Neustartmodus überwachen, indem Sie den Befehl **LIST UTILITIES** auf einem beliebigen aktiven Member ausführen. Der Befehl **LIST UTILITIES** kann den globalen Recoverystatus aller Member, einschließlich der Member im Light-Neustartmodus, zurückgeben.

```
LIST UTILITIES SHOW DETAIL

ID                                   = 1
Typ                                  = MEMBER CRASH RECOVERY
Datenbankname                        = SAMPLE
Partitionsnummer                     = 0
Beschreibung                         = Member Crash Recovery (Light Mode)
Startzeit                            = 11/22/2007 15:20:05.646020
Status                               = Wird ausgeführt
```

```
 Aufruftyp                       = Automatisch
 Fortschrittsüberwachung:
    Geschätzte Fertigstellung (%) = 0
    Phasennummer [aktuell]        = 1
       Beschreibung               = Vorwärts
       Gesamte Arbeit             = 4193976 Byte
       Abgeschlossene Arbeit      = 0 Byte
       Startzeit                  = 11/22/2007 15:20:05.646121
    Phasennummer                  = 2
       Beschreibung               = Rückwärts
       Gesamte Arbeit             = 4193976 Byte
       Abgeschlossene Arbeit      = 0 Byte
       Startzeit                  = Nicht gestartet
```

Es gibt verschiedene Methoden, mit denen Sie eine Sicht mit Zeitangabe Ihrer DB2 pureScale-Umgebung abrufen können, wie z. B. die Tabellenfunktion DB2_GET_INSTANCE_INFO, die Verwaltungssicht DB2_MEMBER sowie die Befehle **LIST INSTANCE** und **db2instance**. Mit jeder dieser Methoden können Sie feststellen, ob sich Member im Light-Neustartmodus befinden, auf welchen Host sie durch eine Recovery versetzt werden und in welchem Status der Recovery mit Light-Neustart sie sich gerade befinden.

In diesem Beispiel zeigt die Verwaltungssicht DB2_MEMBER, dass Member 2 im Light-Modus auf Host so3 erneut gestartet wird.

```
SELECT ID,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CURRENT_HOST,
       varchar(STATE,21) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_MEMBER

ID     HOME_HOST CURRENT_HOST STATE      ALERT
------ --------- ------------ ---------- -----
     1 so1       so1          STARTED    NO
     2 so2       so3          RESTARTING NO
     3 so3       so3          STARTED    NO

     3 Satz/Sätze ausgewählt.
```

Szenario: Light-Neustart

Dieses Szenario beschreibt die Schritte, die während des Member-Neustarts im Light-Modus ausgeführt werden. Es behandelt den häufigsten Fall, in dem ein einzelner Hostfehler auftritt, der zur Folge hat, dass das residente Member des Hosts automatisch als Gastmember auf einem anderen, noch aktiven Host erneut gestartet wird. Das Szenario veranschaulicht außerdem, wie das Gastmember auf seinen Benutzerhost zurückübertragen (Failback) wird.

Erstkonfiguration

In der DB2 pureScale-Instanz befinden sich sechs Hosts (HostA, HostB, HostC, HostD, HostE, HostF):

- Member 10 wird auf HostA (als Benutzerhost) ausgeführt.
- Member 20 wird auf HostB (als Benutzerhost) ausgeführt.
- Member 30 wird auf HostC (als Benutzerhost) ausgeführt.
- Member 40 wird auf HostD (als Benutzerhost) ausgeführt.
- Cluster-Caching-Funktion 128 (CF 128) wird auf HostE ausgeführt.
- Cluster-Caching-Funktion 129 (CF 129) wird auf HostF ausgeführt.

Für die Instanz auf jedem Host ist eine Gruppe von DB2-Leerlaufprozessen mit vorab zugeordnetem Speicher vorhanden, der für die Ausführung einer Recovery mit Light-Neustart reserviert ist. Die DB2-Cluster-Services überwachen alle Ressourcen im Cluster.

Die Statusinformationen für die Hosts, Member und CFs können mithilfe des Befehls **LIST INSTANCE** angezeigt werden . An diesem Punkt gibt der Befehl **LIST INSTANCE** Folgendes zurück:

```
LIST INSTANCE

MEMBER_ID TYPE    STATE   HOME_HOST CURRENT_HOST ALERT
--------- ------- ------- --------- ------------ -----
       10 MEMBER  STARTED hostA     hostA        NO
       20 MEMBER  STARTED hostB     hostB        NO
       30 MEMBER  STARTED hostC     hostC        NO
       40 MEMBER  STARTED hostD     hostD        NO
      128 CF      PRIMARY hostE     -            NO
      129 CF      PEER    hostF     -            NO


HOSTNAME STATE  INSTANCE_STOPPED ALERT
-------- ------ ---------------- -----
hostA    ACTIVE NO               NO
hostB    ACTIVE NO               NO
hostC    ACTIVE NO               NO
hostD    ACTIVE NO               NO
hostE    ACTIVE NO               NO
hostF    ACTIVE NO               NO
```

Hostfehler

Bei Server HostA fällt das Netz aus. Die DB2-Cluster-Services können das Member 10 auf HostA nicht erneut starten; deshalb starten sie das Member im Light-Modus auf dem nächsten verfügbaren Host: HostB.

An diesem Punkt zeigt der Befehl **LIST INSTANCE**, dass sich Member 10 nun im Status RESTARTING befindet und dass HostB sein aktueller Host ist. Der Status von HostA ist INACTIVE (das Feld INSTANCE_STOPPED ist nicht gesetzt, weil die Instanz manuell auf HostA gestoppt wurde) und es liegt ein Alert vor:

```
LIST INSTANCE

MEMBER_ID TYPE    STATE      HOME_HOST CURRENT_HOST ALERT
--------- ------- ---------- --------- ------------ -----
       10 MEMBER  RESTARTING hostA     hostB        NO
       20 MEMBER  STARTED    hostB     hostB        NO
       30 MEMBER  STARTED    hostC     hostC        NO
       40 MEMBER  STARTED    hostD     hostD        NO
      128 CF      PRIMARY    hostE     -            NO
      129 CF      PEER       hostF     -            NO


HOSTNAME STATE    INSTANCE_STOPPED ALERT
-------- -------- ---------------- -----
hostA    INACTIVE NO               YES
hostB    ACTIVE   NO               NO
hostC    ACTIVE   NO               NO
hostD    ACTIVE   NO               NO
hostE    ACTIVE   NO               NO
hostF    ACTIVE   NO               NO
```

Warten auf Rückübertragung (Failback)

Nach dem Start des Prozessmodells wird die Recovery nach dem Absturz des Members für jede Datenbank ausgeführt, für die dies erforderlich ist. Zur Überprüfung des Fortschritts der Recovery nach dem Absturz des Members verwenden Sie den Befehl **LIST UTILITIES** mit der Option **SHOW DETAIL**, wie dies in Überwachen von Membern beim Light-Neustart beschrieben wird. Nach dem Abschluss der Recovery nach dem Absturz des Members wartet das Member 10 darauf, dass es auf 'HostA' zurückübertragen (Failback) wird. Bis dahin kann es keine neuen Transaktionen verarbeiten. Sind unbestätigte Transaktionen vorhanden, müssen diese aufgelöst werden, während Member 10 auf die Rückübertragung wartet.

An diesem Punkt zeigt der Befehl **LIST INSTANCE**, dass sich Member 10 nun im Status WAITING_FOR_FAILBACK befindet:

```
LIST INSTANCE

MEMBER_ID TYPE    STATE                HOME_HOST CURRENT_HOST ALERT
--------- ------- -------------------- --------- ------------ -----
       10 MEMBER  WAITING_FOR_FAILBACK hostA     hostB        NO
       20 MEMBER  STARTED              hostB     hostB        NO
       30 MEMBER  STARTED              hostC     hostC        NO
       40 MEMBER  STARTED              hostD     hostD        NO
      128 CF      PRIMARY              hostE     -            NO
      129 CF      PEER                 hostF     -            NO


HOSTNAME STATE    INSTANCE_STOPPED ALERT
-------- -------- ---------------- -----
hostA    INACTIVE NO               YES
hostB    ACTIVE   NO               NO
hostC    ACTIVE   NO               NO
hostD    ACTIVE   NO               NO
hostE    ACTIVE   NO               NO
hostF    ACTIVE   NO               NO
```

Problem mit dem Host gelöst

Die Stromversorgung für HostA ist wiederhergestellt. Folglich wird HostA in der DB2 pureScale-Instanz wieder aktiv.

An diesem Punkt zeigt der Befehl **LIST INSTANCE**, dass HostA nun aktiv ist und dass der Alert gelöscht wurde:

```
LIST INSTANCE

MEMBER_ID TYPE    STATE                HOME_HOST CURRENT_HOST ALERT
--------- ------- -------------------- --------- ------------ -----
       10 MEMBER  WAITING_FOR_FAILBACK hostA     hostB        NO
       20 MEMBER  STARTED              hostB     hostB        NO
       30 MEMBER  STARTED              hostC     hostC        NO
       40 MEMBER  STARTED              hostD     hostD        NO
      128 CF      PRIMARY              hostE     -            NO
      129 CF      PEER                 hostF     -            NO


HOSTNAME STATE  INSTANCE_STOPPED ALERT
-------- ------ ---------------- -----
hostA    ACTIVE NO               NO
hostB    ACTIVE NO               NO
hostC    ACTIVE NO               NO
hostD    ACTIVE NO               NO
hostE    ACTIVE NO               NO
hostF    ACTIVE NO               NO
```

Rückübertragung des Members auf den Benutzerhost

Die DB2-Cluster-Services erkennen, dass 'HostA' aktiv ist und führen automatisch eine Rückübertragung von Member 10 auf diesen Host durch.

An diesem Punkt zeigt der Befehl **LIST INSTANCE**, dass sich Member 10 nun im Status RESTARTING befindet und dass sein aktueller Host wieder HostA ist:

```
LIST INSTANCE SHOW DETAIL

MEMBER_ID TYPE    STATE      HOME_HOST CURRENT_HOST ALERT
--------- ------- ---------- --------- ------------ -----
       10 MEMBER  RESTARTING hostA     hostA        NO
       20 MEMBER  STARTED    hostB     hostB        NO
       30 MEMBER  STARTED    hostC     hostC        NO
       40 MEMBER  STARTED    hostD     hostD        NO
      128 CF      PRIMARY    hostE     -            NO
      129 CF      PEER       hostF     -            NO


HOSTNAME STATE  INSTANCE_STOPPED ALERT
-------- ------ ---------------- -----
hostA    ACTIVE NO               NO
hostB    ACTIVE NO               NO
hostC    ACTIVE NO               NO
hostD    ACTIVE NO               NO
hostE    ACTIVE NO               NO
```

Neustart auf dem Benutzerhost

Wenn Member 10 den Memberneustart auf 'HostA' erfolgreich abschließt, ändert sich sein Status in STARTED. Das Member kann nun neue Transaktionen verarbeiten und Benutzerverbindungen akzeptieren. An diesem Punkt gibt der Befehl **LIST INSTANCE** dieselben Informationen zur DB2 pureScale-Instanz zurück wie in Erstkonfiguration.

Manuelles Eingreifen in Fehlersituationen

In den meisten Fällen müssen die Benutzer keine Maßnahme ergreifen, weil die DB2-Cluster-Services automatisch den erforderlichen Neustart oder die erforderliche Recovery nach einem Absturz initialisieren. Es gibt aber auch Fälle, in denen Sie den Neustart oder die Recovery nach einem Absturz manuell initialisieren oder Maßnahmen zur Recovery nach dem Absturz einer Gruppe ergreifen müssen.

Einleiten einer Recovery nach Absturz einer Gruppe

Eine Recovery nach dem Absturz einer Gruppe wird in fast allen Fällen automatisch ausgeführt, ohne dass eine Benutzeraktion erforderlich ist. Sie initiieren eine solche Recovery nur in Situation manuell, in denen es der Cluster-Manager zum wiederholten Mal nicht geschafft hat, eine Recovery nach dem Absturz einer Gruppe durchzuführen oder der Cluster-Manager (da möglicherweise das Cluster-Management inaktiviert wurde) nicht verfügbar ist. Ein weiterer Grund wäre, wenn der Konfigurationsparameter **autorestart** auf OFF gesetzt ist.

Informationen zu diesem Vorgang

Nur ein Member kann zu einem gegebenen Zeitpunkt eine Recovery nach dem Absturz einer Gruppe für eine Datenbank ausführen. Falls zwei Member eine Recovery nach dem Absturz einer Gruppe für dieselbe Datenbank zur selben Zeit einlei-

ten, wartet der Befehl des zweiten Members ab, bis die Recovery nach dem Absturz einer Gruppe auf dem ersten Member abgeschlossen ist.

Falls auf dem Member, für das nach dem Absturz einer Gruppe eine Recovery durchgeführt wurde, nach Abschluss der Recovery-Operation unbestätigte Transaktionen vorhanden sind, bleibt das Member, das die Recovery nach Absturz einer Gruppe durchgeführt hat, aktiv und es ist in der Lage, neue Arbeit zu übernehmen. Auf jegliche Daten in Verbindung mit den unbestätigten Transaktionen kann jedoch nicht zugegriffen werden, da die Zeilen, auf die die unbestätigte Transaktion zugreift, bis zur endgültigen Auflösung geschützt werden müssen (d. h., sie sind gesperrt). Die Tatsache, dass auf dem Member mit der Recovery nach dem Absturz einer Gruppe unbestätigte Transaktionen vorhanden sind, bedeutet, dass bei der nächstfolgenden Aktivierung der Datenbank auf dem Member eine weitere Recovery nach dem Absturz eines Members vorgenommen werden muss, falls das Member fehlschlägt, bevor die unbestätigten Transaktionen aufgelöst sind.

Falls eine Recovery nach dem Absturz einer Gruppe für Member A vorgenommen und dabei festgestellt wird, dass diesem Member während der Recovery nach dem Absturz einer Gruppe mindestens eine unbestätigte Transaktion von Member B zugeordnet wurde, bleibt Member B nach der Recovery nach dem Absturz einer Gruppe inkonsistent. Dies beinhaltet, dass für das Member B eine Recovery nach dem Absturz eines Members durchgeführt werden muss, bevor das Member B nach dem Abschluss der Recovery nach dem Absturz einer Gruppe gestartet werden kann. Nach dem Abschluss der Recovery nach dem Absturz eines Members steht das Member B wieder für die Annahme neuer Verbindungen zur Verfügung. Es sollte beachtet werden, dass diese nachfolgende Recovery nach dem Absturz eines Members sehr schnell abgeschlossen wird, da keine Arbeit zum Wiederherstellen oder Rückgängigmachen erforderlich ist (die Recovery nach einem Systemabsturz muss nur die unbestätigten Transaktionen in die Transaktionstabelle des Members laden) und alle Sperren, die zum Schutz der durch die unbestätigten Transaktionen betroffenen Daten erforderlich sind, in der Cluster-Caching-Funktion bereits für das Member reserviert sind. Darüber hinaus ist diese nachfolgende Operation RESTART in den meisten Fällen für den Benutzer verdeckt, da die Recovery nach dem Absturz eines Members bei Aktivierung von AUTORESTART automatisch ausgeführt wird, wenn auf dem entsprechenden Member erstmals eine Verbindung zur Datenbank hergestellt wird.

Vorgehensweise

Zur manuellen Einleitung einer Recovery nach dem Absturz einer Gruppe geben Sie den Befehl **RESTART DATABASE** auf einem der Member ein.

Ergebnisse

Wenn die Recovery nach dem Absturz einer Gruppe abgeschlossen ist, ist das Member, auf dem der Befehl ausgeführt wurde, in der Lage, neue Verbindungen zu akzeptieren. Dabei sind alle Daten mit Ausnahme derjenigen verfügbar, die durch nicht aufgelöste unbestätigte Transaktionen gesperrt bleiben. Nach Abschluss des Befehls **RESTART DATABASE** wird eine Verbindung zur Datenbank beibehalten, wenn der Benutzer über das Zugriffsrecht CONNECT für die Datenbank verfügt. Die Datenbank wird auf keinem anderen Member als dem Member aktiviert, auf dem die Recovery nach dem Absturz einer Gruppe ausgeführt wurde.

Einleiten einer Recovery nach dem Absturz eines Members

Eine Recovery nach dem Absturz eines Members wird in fast allen Fällen automatisch ausgeführt, ohne dass eine Benutzeraktion erforderlich ist. Sie initiieren eine Recovery nach Absturz eines Members nur in solchen Situationen manuell, in denen es der Cluster-Manager zum wiederholten Mal nicht geschafft hat, eine Recovery nach einem Systemabsturz durchzuführen, der Cluster-Manager nicht verfügbar ist (da möglicherweise das Cluster-Management inaktiviert wurde) oder wenn der Konfigurationsparameter **autorestart** auf OFF gesetzt ist.

Informationen zu diesem Vorgang

Wenn die DB2 pureScale-Instanz mehrere Member enthält, die eine Recovery nach dem Absturz eines Members erfordern, führt der Cluster-Manager mehrere Recoverys nach dem Absturz eines Members parallel aus. Dadurch lassen sich Blockadesituationen leichter vermeiden, die entstehen, wenn ein ausgefallenes Member eine Abhängigkeit von Ressourcen hat, die von einem anderen ausgefallenen Member belegt sind.

Vorgehensweise

Zur manuellen Einleitung einer Recovery nach dem Absturz eines Members setzen Sie den Befehl **RESTART DATABASE** ab.
Wenn der Befehl **RESTART DATABASE** abgesetzt wird, ermittelt DB2 for Linux, UNIX and Windows, ob eine Recovery nach Absturz einer Gruppe oder nach Absturz eines Members erforderlich ist.

Ergebnisse

Sobald die Recovery nach dem Absturz eines Members abgeschlossen ist, wird die Datenbank auf dem betreffenden Member aktiviert und zum Empfang von Verbindungen von anderen Anwendungen verfügbar gemacht.

Recovery bei beschädigten Tabellenbereichen

Wenn Tabellenbereichscontainer beschädigt wurden und eine Recovery nach dem Absturz einer Gruppe für die Datenbank erforderlich ist, schlägt die Operation der Recovery nach dem Absturz einer Gruppe fehl. In einer DB2 pureScale-Umgebung mit automatischer Recovery können Sie ein solches Szenario wie folgt lösen.

Vorbereitende Schritte

Dieses Szenario zeigt, wie das Problem beschädigter Tabellenbereiche in einer DB2 pureScale-Umgebung so gelöst werden kann, dass die Recovery nach dem Absturz einer Gruppe fortgesetzt werden kann.

Informationen zu diesem Vorgang

Angenommen, die automatische Recovery ist aktiv, kann jedoch die Datenbank nicht wiederherstellen, da ein Tabellenbereich beschädigt ist. Nach wiederholten Fehlversuchen wird das DB2-Member auf einen anderen Host versetzt und ein Light-Neustart versucht. Nach weiteren Fehlversuchen einer automatischen Recovery der Datenbank löst der Cluster-Manager einen Alert aus. Das Member mit dem Fehler wird nicht gestartet und bleibt inaktiv.

Vorgehensweise

1. Inaktivieren Sie die automatische Recovery, indem Sie den Konfigurationsparameter **autorestart** auf den Wert OFF setzen. Der Konfigurationsparameter **autorestart** ist nicht dynamisch und sein neuer Wert wird beim nächsten Start des Members wirksam. Dieser Befehl setzt den Konfigurationsparameter **autorestart** für eine Datenbank namens WSDB auf den Wert OFF:

   ```
   UPDATE DATABASE CONFIGURATION FOR WSDB USING AUTORESTART OFF
   ```

2. Verwenden Sie den Befehl **db2cluster**, um alle ausstehenden Alerts für alle Member zu löschen. Außerdem setzen die DB2-Cluster-Services automatisch den Befehl **db2start** ab, um die Member auf ihren Benutzerhosts erneut zu starten. Die neu gestarteten Member übernehmen den geänderten Wert für **autorestart** und initialisieren nicht die automatische Recovery.

   ```
   db2cluster -clear -alert
   ```

3. Geben Sie den Befehl **RESTART DATABASE** ein, um die beschädigten Tabellenbereiche zu entfernen. Die beschädigten Tabellenbereiche werden als offline markiert und in den Status 'Löschen anstehend' (DROP PENDING) versetzt.

   ```
   RESTART DATABASE WSDB DROP PENDING TABLESPACES (tbsp1, tbsp2)
   ```

4. Aktivieren Sie die automatische Recovery erneut, indem Sie den Konfigurationsparameter **autorestart** auf den Wert ON setzen. Dieser Befehl setzt beispielsweise den Konfigurationsparameter **autorestart** für die Datenbank WSDB auf den Wert ON:

   ```
   UPDATE DATABASE CONFIGURATION FOR WSDB USING AUTORESTART ON
   ```

   Der Parameter **autorestart** ist nicht dynamisch und sein neuer Wert wird beim nächsten Start des Members wirksam.

Teil 2. Datenrecovery

Eine Recovery ist die Wiederherstellung einer Datenbank oder eines Tabellenbereichs nach einem Problem, zum Beispiel einem Datenträger- oder Speicherfehler, einem Stromausfall oder einem Anwendungsfehler. Wenn Sie Ihre Datenbank oder einzelne Tabellenbereiche mit Backup gesichert haben, können Sie diese erneut erstellen, wenn sie beschädigt oder aus einem anderen Grund unbrauchbar werden.

Vier Arten der Recovery stehen zur Verfügung:

- Eine Recovery nach einem Systemabsturz verhindert, dass eine Datenbank nach unerwartetem Abbruch von Transaktionen bzw. UOWs (Units of Work, Arbeitseinheiten) in einem inkonsistenten oder nicht verwendbaren Zustand verbleibt.
- Unter dem Begriff 'Recovery nach einem Katastrophenfall' werden die Aktivitäten zusammengefasst, die zum Wiederherstellen der Datenbank nach einem Brand, einem Erdbeben, nach Vandalismus oder anderen zerstörerischen Ereignissen erforderlich sind.
- Eine Versionsrecovery ist die Wiederherstellung einer früheren Version der Datenbank mithilfe eines Images der Datenbank, das im Rahmen einer Backup-Operation erstellt wurde.
- Eine aktualisierende Recovery dient zur erneuten Anwendung von Änderungen, die durch Transaktionen nach einem Backup festgeschrieben wurden.

Der DB2-Datenbankmanager startet eine Recovery nach einem Systemabsturz automatisch, um z. B. eine Datenbank nach einem Stromausfall wiederherzustellen. Eine beschädigte Datenbank können Sie durch eine Versionsrecovery oder eine aktualisierende Recovery wiederherstellen.

Kapitel 7. Entwickeln einer Backup- und Recoverystrategie

Eine Datenbank kann aufgrund von Hardware- oder Softwarefehlern (oder beidem) unbrauchbar werden. Irgendwann treten möglicherweise Speicherprobleme, Stromausfälle oder Anwendungsfehler auf und jedes Fehlerszenario erfordert unterschiedliche Recoveryaktionen.

Schützen Sie Ihre Daten vor Verlust, indem Sie eine gut erprobte Recoverystrategie bereithalten.

Bei der Entwicklung Ihrer Recoverystrategie sollten Sie unter anderem folgende Fragen berücksichtigen:

- Soll die Datenbank wiederherstellbar sein?
- Wie viel Zeit steht für die Recovery der Datenbank zur Verfügung?
- In welchen Abständen werden Backup-Operationen durchgeführt?
- Wie viel Speicher kann für Backupkopien und Archivprotokolldateien zugeordnet werden?
- Sind Backups auf Tabellenbereichsebene ausreichend, oder muss die gesamte Datenbank gesichert werden?
- Sollte ein Bereitschaftssystem manuell oder über HADR (High Availability Disaster Recovery) konfiguriert werden?

Eine Strategie zur Recovery der Datenbank sollte sicherstellen, dass alle Informationen verfügbar sind, wenn sie für eine Recovery der Datenbank benötigt werden. Sie sollte einen Zeitplan für regelmäßige Backups enthalten. Im Fall von Umgebungen mit partitionierten Datenbanken sollten auch nach einer Skalierung des Systems, d. h. nach dem Hinzufügen oder Löschen von Datenbankpartitionsservern oder -knoten, Backups durchgeführt werden. Ihre Gesamtstrategie sollte auch Prozeduren zum Wiederherstellen von Befehlsscripts, Anwendungen, benutzerdefinierten Funktionen, Code gespeicherter Prozeduren in Betriebssystembibliotheken sowie von Ladekopien enthalten.

In den folgenden Abschnitten werden verschiedene Recoverymethoden behandelt. Sie können bestimmen, welche Recoverymethode für Ihre Geschäftsumgebung am besten geeignet ist.

Das Konzept eines *Datenbank-Backups* ist mit jedem anderen Daten-Backup vergleichbar: Sie erstellen eine Kopie der Daten und speichern die Kopie anschließend auf einem anderen Datenträger für den Fall eines Ausfalls oder einer Beschädigung der Originaldaten. Den einfachsten Fall eines Backups bildet das Verfahren, bei dem zunächst die Datenbank gestoppt wird, um sicherzustellen, dass keine weiteren Transaktionen auftreten, und anschließend ein Backup-Image der Daten erstellt wird. Sie können die Datenbank dann erneut erstellen, falls sie beschädigt wird.

Das erneute Erstellen der Datenbank wird als *Recovery* bezeichnet. Eine *Versionsrecovery* ist die Wiederherstellung einer früheren Version der Datenbank mithilfe eines Images der Datenbank, das im Rahmen einer Backup-Operation erstellt wurde. Eine *aktualisierende Recovery* ist die erneute Anwendung von in den Datenbankprotokolldateien aufgezeichneten Transaktionen nach dem Restore eines Backup-Images einer Datenbank oder eines Tabellenbereichs.

Eine *Recovery nach einem Systemabsturz* ist die automatische Recovery der Datenbank, wenn ein Fehler auftritt, bevor alle Änderungen einer oder mehrerer UOWs (Transaktionen) beendet und festgeschrieben wurden. Dies geschieht durch den Rollback der unvollständigen Transaktionen und Beenden der festgeschriebenen Aktionen, die sich noch im Hauptspeicher befanden, als der Systemabsturz auftrat.

Die Protokolldateien für die Recovery und die Datei des Recoveryprotokolls werden automatisch erstellt, wenn eine Datenbank erstellt wird (Abb. 11 auf Seite 289). Diese Protokolldateien sind wichtig, falls Sie verlorene oder beschädigte Daten wiederherstellen müssen.

Jede Datenbank enthält *Recoveryprotokolle*, die für die Recovery nach Anwendungs- oder Systemfehlern verwendet werden. In Kombination mit den Datenbankbackups dienen sie zur Recovery der Konsistenz der Datenbank bis exakt zu dem Zeitpunkt, an dem der Fehler auftrat.

Die *Datei des Recoveryprotokolls* enthält eine Zusammenfassung der Backupinformationen, die zur Bestimmung der Recoveryoptionen verwendet werden können, wenn die Datenbank insgesamt oder teilweise bis zu einem bestimmten Zeitpunkt wiederhergestellt werden muss. Sie dient zur Protokollierung recoveryrelevanter Ereignisse wie z. B. Backup- und Restoreoperationen.Diese Datei befindet sich im Datenbankverzeichnis.

Die *Protokolldatei der Tabellenbereichsänderungen*, die sich ebenfalls im Datenbankverzeichnis befindet, enthält Informationen, anhand deren bestimmt werden kann, welche Protokolldateien für die Recovery eines bestimmten Tabellenbereichs erforderlich sind.

Die Datei des Recoveryprotokolls und die Protokolldatei der Tabellenbereichsänderungen können nicht direkt geändert werden. Mit dem Befehl **PRUNE HISTORY** können Sie jedoch Einträge aus den Dateien löschen. Sie haben auch die Möglichkeit, mit dem Datenbankkonfigurationsparameter **rec_his_retentn** die Anzahl der Tage anzugeben, die diese Protokolldateien beibehalten werden sollen.

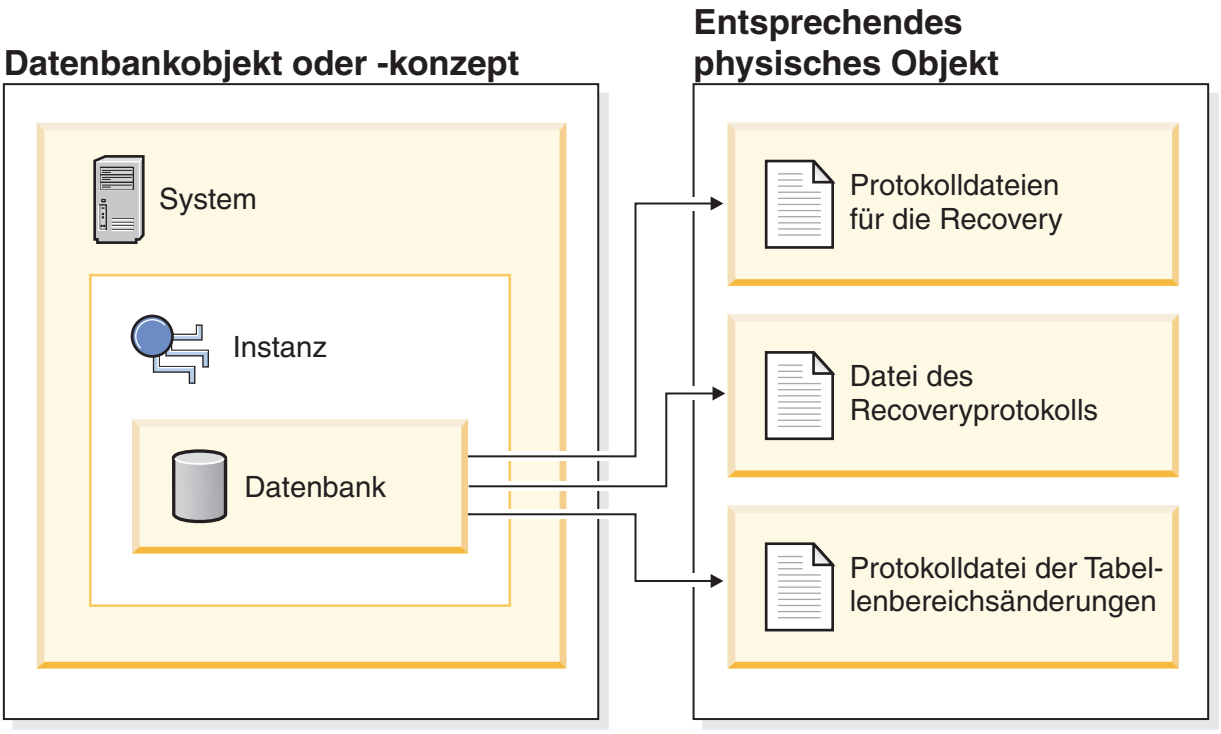


Abbildung 11. Datenbankrecoverydateien

Daten, die sich problemlos erneut erstellen lassen, können in einer nicht wiederherstellbaren Datenbank gespeichert werden. Dazu gehören Daten von einer externen Quelle, die für Anwendungen mit Lesezugriff verwendet werden, und Tabellen, die nicht oft aktualisiert werden und für die der geringe Protokollierungsaufwand nicht die zusätzliche Komplexität der Verwaltung von Protokolldateien sowie die aktualisierende Recovery nach einer Restoreoperation rechtfertigt. Wenn die Datenbankkonfigurationsparameter **logarchmeth1** und **logarchmeth2** beide auf OFF gesetzt sind, ist die Datenbank *nicht wiederherstellbar*. Dies bedeutet, dass lediglich die für die Recovery nach einem Systemabsturz erforderlichen Protokolle beibehalten werden. Diese Protokolle werden als *aktive Protokolldateien* bezeichnet und enthalten aktuelle Transaktionsdaten. Die Versionsrecovery mithilfe von *Offline-Backups* ist das primäre Mittel zur Recovery einer nicht wiederherstellbaren Datenbank. (Ein Offline-Backup bedeutet, dass während der Backup-Operation keine andere Anwendung die Datenbank verwenden kann.) Eine solche Datenbank kann nur offline wiederhergestellt werden. Sie wird in dem Status wiederhergestellt, den sie hatte, als das Backup-Image erstellt wurde. Eine aktualisierende Recovery wird nicht unterstützt.

Daten, die sich *nicht* einfach erneut erstellen lassen, sollten in einer wiederherstellbaren Datenbank gespeichert werden. Hierzu gehören Daten, deren Quelle nach dem Laden der Daten zerstört wird, Daten, die manuell in Tabellen eingegeben werden, sowie Daten, die nach dem Laden in die Datenbank von Anwendungsprogrammen oder Benutzern geändert werden. Bei *wiederherstellbaren Datenbanken* sind die Datenbankkonfigurationsparameter **logarchmeth1** oder **logarchmeth2** auf einen anderen Wert als OFF gesetzt. Aktive Protokolldateien sind weiterhin für die Datenbankrecovery nach einem Systemabsturz verfügbar. Es stehen jedoch auch die *Archivprotokolldateien* zur Verfügung, die festgeschriebene Transaktionsdaten enthalten. Eine solche Datenbank kann nur offline wiederhergestellt werden. Sie wird in dem Status wiederhergestellt, den sie hatte, als das Backup-Image erstellt wurde. Mithilfe der aktualisierenden Recovery (Rollforward Recovery) können Sie jedoch

die Datenbank aktualisierend wiederherstellen (also über den Zeitpunkt hinaus, an dem das Backup-Image erstellt wurde), indem Sie die aktiven und archivierten Protokolle entweder bis zu einem bestimmten Zeitpunkt oder bis zum Ende der aktiven Protokolle anwenden.

Backup-Operationen für wiederherstellbare Datenbanken können entweder offline oder *online* durchgeführt werden (online heißt, dass andere Anwendungen während der Backup-Operation eine Verbindung zur Datenbank herstellen können). Operationen zum Online-Restore eines Tabellenbereichs und zur aktualisierenden Recovery werden nur für wiederherstellbare Datenbanken unterstützt. Wenn die Datenbank nicht wiederherstellbar ist, müssen Operationen zum Restore und zur aktualisierenden Recovery der Datenbank offline ausgeführt werden. Während eines Online-Backups stellt die aktualisierende Recovery sicher, dass *alle* Tabellenänderungen erfasst und erneut angewendet werden, wenn dieses Backup wiederhergestellt wird.

Bei einer wiederherstellbaren Datenbank können Sie anstelle der gesamten Datenbank auch einzelne Tabellenbereiche sichern, wiederherstellen und aktualisierend wiederherstellen. Wenn Sie einen Tabellenbereich online sichern, kann er weiterhin verwendet werden, und gleichzeitig durchgeführte Änderungen werden in den Protokollen aufgezeichnet. Wenn Sie einen Tabellenbereich online wiederherstellen oder online aktualisierend wiederherstellen, kann der Tabellenbereich selbst nicht mehr verwendet werden, bis die Operation beendet ist, Benutzer können jedoch weiterhin auf Tabellen in anderen Tabellenbereichen zugreifen.

Automatisierte Backup-Operationen

Da die Bestimmung, ob und wann Verwaltungsaktivitäten wie zum Beispiel Backup-Operationen auszuführen sind, zeitaufwändig sein kann, haben Sie die Möglichkeit, die Verwaltung automatisch durchzuführen. Bei der automatischen Verwaltung definieren Sie Ihre Verwaltungszielsetzungen, einschließlich der möglichen Ausführungszeiten für die automatische Verwaltung. Anschließend ermittelt DB2 anhand dieser Informationen, ob Verwaltungsaktivitäten erforderlich sind und führt im nächsten verfügbaren Verwaltungsfenster (einem benutzerdefinierten Zeitraum zum Ausführen von automatischen Verwaltungsaktivitäten) nur die erforderlichen Verwaltungsaktivitäten aus.

Anmerkung: Nach der Konfiguration der automatischen Verwaltung haben Sie weiterhin die Möglichkeit, manuelle Backup-Operationen ausführen. DB2 führt automatische Backup-Operationen nur dann aus, wenn sie erforderlich sind.

Häufigkeit von Backups

In Ihrem Recoveryplan sollten regelmäßige Backup-Operationen vorgesehen sein, da das Sichern einer Datenbank Zeit und Systemressourcen in Anspruch nimmt. Ihr Plan kann eine Kombination aus Datenbankgesamtbackups und inkrementellen Backups enthalten. Die Häufigkeit und die Arten von Backups, die Sie vornehmen, wirken sich auch auf die für die Datenbankrecovery benötigte Zeit aus.

Sie sollten in regelmäßigen Abständen Gesamtbackups der Datenbanken erstellen, selbst wenn Sie die Protokolle archivieren, um eine aktualisierende Recovery zu ermöglichen. Für die Recovery einer Datenbank können Sie entweder ein Image eines Datenbankgesamtbackups verwenden, das Backup-Images aller Tabellenbereiche enthält, oder Sie können die Datenbank mithilfe ausgewählter Tabellenbereichsimages erneut erstellen (Rebuildfunktion). Images von Tabellenbereichsbackups sind auch sinnvoll für eine Recovery nach dem Ausfall einer einzel-

nen Platte oder nach einem Anwendungsfehler. In Umgebungen mit partitionierten Datenbanken müssen Sie jeweils nur die Tabellenbereiche wiederherstellen, die sich in den Datenbankpartitionen befinden, die ausgefallen sind. Es müssen nicht alle Tabellenbereiche oder Datenbankpartitionen wiederhergestellt werden.

Obwohl Datenbankgesamtbackups zur Datenbankwiederherstellung nicht länger erforderlich sind, da Sie eine Datenbank mithilfe von Tabellenbereichsimages wiederherstellen können, ist es dennoch sinnvoll, gelegentlich Gesamtbackups der Datenbank zu erstellen.

Außerdem sollten Sie in Betracht ziehen, die Backup-Images und Protokolldateien nicht zu überschreiben, sondern mindestens zwei Gesamtbackup-Images der Datenbank mit zugehörigen Protokollen als weitere Vorsichtsmaßnahme zu sichern.

Wenn die erforderliche Zeit für die Anwendung der archivierten Protokolldateien bei der (aktualisierenden) Recovery einer aktiven Datenbank wichtig ist, sollten Sie den Aufwand häufigerer Datenbank-Backups in Erwägung ziehen. Durch häufigere Backups wird die Anzahl der archivierten Protokolldateien verringert, die Sie bei einer aktualisierenden Recovery anwenden müssen.

Aspekte des Online- und Offline-Backups

Sie können eine Backup-Operation starten, während die Datenbank online oder offline ist. Wenn die Datenbank online ist, können andere Anwendungen oder Prozesse Verbindungen zur Datenbank herstellen sowie Daten lesen und ändern, während die Backup-Operation ausgeführt wird. Wird die Backup-Operation offline ausgeführt, können andere Anwendungen keine Verbindung zur Datenbank herstellen.

Um den Zeitraum, in dem die Datenbank nicht verfügbar ist, möglichst kurz zu halten, sollten Sie Online-Backup-Operationen in Betracht ziehen. Online-Backup-Operationen werden nur unterstützt, wenn die aktualisierende Recovery aktiviert ist. Wenn die aktualisierende Recovery aktiviert ist und Sie über einen kompletten Satz von Recoveryprotokollen verfügen, können Sie die Datenbank im Bedarfsfall wiederherstellen. Sie können ein Online-Backup-Image nur dann für die Recovery verwenden, wenn Sie über die Protokolle verfügen, die sich über den Zeitraum erstrecken, über den die Backup-Operation ausgeführt wurde.

Offline-Backup-Operationen sind schneller als Online-Backup-Operationen, da zwischen den Datendateien keine Konkurrenzsituation auftritt.

Aspekte des Backups ausgewählter Tabellenbereiche

Sie können das Backup-Dienstprogramm auch dazu verwenden, nur ein Backup für ausgewählte Tabellenbereiche durchzuführen. Wenn Sie DMS-Tabellenbereiche verwenden, können Sie verschiedene Datentypen in speziellen Tabellenbereichen speichern, um die für Backup-Operationen benötigte Zeit zu verringern. Sie können die Tabellendaten in einem Tabellenbereich, die Langfeld- und LOB-Daten in einem anderen Tabellenbereich und die Indizes in einem dritten Tabellenbereich unterbringen. Wenn Sie Ihre Daten in verschiedene Tabellenbereiche aufteilen und ein Plattenfehler auftritt, wird der Plattenfehler wahrscheinlich nur einen der Tabellenbereiche betreffen. Zum Restore oder zur aktualisierenden Recovery eines dieser Tabellenbereiche ist weniger Zeit erforderlich als für den Restore eines einzelnen Tabellenbereichs, der alle Daten enthält.

Sie können auch Zeit sparen, indem Sie zu unterschiedlichen Zeitpunkten Backups unterschiedlicher Tabellenbereiche ausführen, solange die an ihnen vorgenommenen Änderungen nicht dieselben sind. Wenn beispielsweise Langfeld- oder LOB-Daten nicht so häufig geändert werden wie die übrigen Daten, müssen Sie diese Tabellenbereiche auch nicht so häufig sichern. Wenn die Langfeld- und LOB-Daten nicht zur Recovery benötigt werden, ist ein Backup dieser Tabellenbereiche möglicherweise gar nicht erforderlich. Wenn die LOB-Daten von einer getrennten Quelle reproduziert werden können, sollten Sie beim Erstellen oder Ändern einer Tabelle zum Hinzufügen von LOB-Spalten die Option NOT LOGGED verwenden.

Wenn Sie Ihre Langfelddaten, LOB-Daten und Indizes in separaten Tabellenbereichen speichern, diese aber nicht gemeinsam sichern, beachten Sie folgenden Hinweis: Wenn Sie einen Tabellenbereich sichern, der nicht alle Tabellendaten enthält, können Sie keine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt für diesen Tabellenbereich durchführen. Alle Tabellenbereiche, die irgendeine Art von Daten für eine Tabelle enthalten, müssen gleichzeitig bis zum selben Zeitpunkt aktualisierend wiederhergestellt werden.

Aspekte der Tabellenreorganisation

Wenn Sie eine Tabelle reorganisieren, sollten Sie die betroffenen Tabellenbereiche nach Beendigung der Operation sichern. Wenn dann der Fall eintritt, dass die Tabellenbereiche wiederhergestellt werden müssen, muss die Datenreorganisation nicht mehr aktualisierend wiederhergestellt werden.

Aspekte des Änderungsstatus von Tabellenbereichen

Durch die Überprüfung des entsprechenden Änderungsstatus können Sie eine fundierte Entscheidung darüber treffen, ob für einen Tabellenbereich ein Backup durchgeführt werden sollte. Mithilfe des Befehls **db2pd -tablespaces trackmodstate** und des Monitorelements **tbsp_trackmode_state** kann der Status des Tabellenbereichs bezüglich des letzten oder nächsten Backups angezeigt werden. Anhand dieser Informationen können Sie feststellen, ob der Tabellenbereich geändert wurde oder ob für den Tabellenbereich ein Backup durchgeführt werden muss.

Aspekte der für die Datenbankrecovery benötigte Zeit

Die Zeit, die für die Recovery einer Datenbank benötigt wird, setzt sich aus zwei Perioden zusammen:

- Die Zeit, die zum Restore der Backup-Kopie benötigt wird.
- Wenn die Datenbank für die aktualisierende Recovery aktiviert ist, die Zeit, die zur Anwendung der Protokolle während der aktualisierenden Recovery benötigt wird.

Berücksichtigen Sie bei der Ausarbeitung eines Recoveryplans diesen Recoveryaufwand und seine Auswirkung auf den Geschäftsbetrieb. Durch Testen des Gesamtrecoveryplans können Sie ermitteln, ob die Zeit, die zur Recovery der Datenbank benötigt wird, in Anbetracht Ihrer Geschäftserfordernisse angemessen ist. Nach jedem Test stellen Sie möglicherweise fest, dass es vorteilhaft ist, die Häufigkeit der Backups zu erhöhen. Wenn Ihre Strategie eine aktualisierende Recovery vorsieht, wird diese erhöhte Häufigkeit von Backups die Anzahl der zwischen den Backups archivierten Protokolle reduzieren und auf diese Weise die für eine aktualisierende Recovery der Datenbank benötigte Zeit nach einer Restoreoperation verringern.

Aspekte des Speicherbedarfs für Recovery

Bei der Entscheidung für die zu verwendende Recoverymethode sollten Sie auch den erforderlichen Speicherplatz in Betracht ziehen. Durch Backups und die Komprimierung archivierter Protokolldateien können die Speicherkosten in der Datenbankumgebung reduziert werden.

Bei der Versionsrecovery wird Speicherplatz für die Backupkopie der Datenbank und für die wiederhergestellte Datenbank benötigt. Bei einer aktualisierenden Recovery wird Speicherplatz für die Backupkopie der Datenbank bzw. der Tabellenbereiche, für die wiederhergestellte Datenbank sowie für die archivierten Datenbankprotokolle benötigt.

Enthält eine Tabelle Langfeld- oder LOB-Spalten, könnte es sinnvoll sein, die betreffenden Daten in einen separaten Tabellenbereich zu stellen. Diese Aktion hat Auswirkungen auf Ihre Überlegungen zum Speicherbedarf sowie auf Ihren Recoveryplan. Wenn Sie einen separaten Tabellenbereich für Langfeld- und LOB-Daten verwenden und Ihnen der Zeitaufwand für das Sichern der Langfeld- und LOB-Daten bekannt ist, können Sie einen Recoveryplan verwenden, bei dem dieser Tabellenbereich nur gelegentlich gesichert wird. Beim Erstellen oder Ändern einer Tabelle, die LOB-Spalten umfasst, können Sie zudem angeben, dass Änderungen an diesen Spalten nicht protokolliert werden sollen. Durch diese Aktion verringert sich die Größe des erforderlichen Protokollspeicherbereichs und somit der erforderliche Speicherplatz für archivierte Protokolldateien.

Um zu verhindern, dass ein Datenträgerfehler die Datenbank beschädigt und Ihnen die Wiederherstellung unmöglich macht, sollten Sie die Backupkopie der Datenbank, die Datenbankprotokolle sowie die Datenbank selbst auf unterschiedlichen Einheiten speichern. Aus diesem Grund wird ausdrücklich empfohlen, nach dem Erstellen der Datenbank die Datenbankprotokolle mithilfe des Konfigurationsparameters *newlogpath* auf eine separate Einheit umzuleiten.

Die Datenbankprotokolle können viel Speicherplatz in Anspruch nehmen. Wenn Sie beabsichtigen, eine aktualisierende Recovery durchzuführen, müssen Sie entscheiden, wie die archivierten Protokolle verwaltet und komprimiert werden sollen. Ihnen stehen folgende Möglichkeiten zur Auswahl:

- Geben Sie unter Verwendung des Konfigurationsparameters LOGARCHMETH1 oder LOGARCHMETH2 eine Archivierungsmethode für Protokolldateien an.
- Aktivieren Sie mithilfe der Konfigurationsparameter LOGARCHCOMPR1 und LOGARCHCOMPR2 die Komprimierung archivierter Protokolldateien.
- Kopieren Sie die Protokolldateien manuell auf eine Speichereinheit bzw. in ein Verzeichnis, die/das nicht mit dem Verzeichnispfad der Datenbankprotokolle übereinstimmt, nachdem die Protokolldateien nicht mehr zur Gruppe der aktiven Protokolldateien gehören.
- Kopieren Sie diese Protokolle mithilfe eines Benutzerexitprogramms auf eine andere Speichereinheit in Ihrer Umgebung.

Backupkomprimierung

Neben den Speichereinsparungen, die Sie durch die Zeilenkomprimierung in Ihrer aktiven Datenbank erzielen können, haben Sie auch die Möglichkeit, die Backupkomprimierung zu verwenden, um die Größe Ihrer Datenbankbackups zu reduzieren.

Während die Zeilenkomprimierung auf Tabellenbasis erfolgt, werden bei der Komprimierung von Backups *alle* Daten im Backup-Image komprimiert, einschließlich der Katalogtabellen, Indexobjekte, LOB-Objekte, Zusatzdatenbankdateien und Datenbankmetadaten.

Sie können die Backupkomprimierung auf Tabellen anwenden, die bereits die Zeilenkomprimierung verwenden. Dabei ist jedoch zu beachten, dass die Backupkomprimierung zusätzliche CPU-Ressourcen und zusätzliche Zeit erfordert. Möglicherweise reicht die Tabellenkomprimierung alleine aus, um eine Reduzierung Ihrer Backupspeicheranforderungen zu erzielen. Wenn Sie bereits die Zeilenkomprimierung verwenden, ist die Backupkomprimierung möglicherweise nur dann sinnvoll, wenn die Speicheroptimierung so wichtig ist, dass die zusätzliche Zeit zur Ausführung des Backups in Kauf genommen werden kann.

Tipp: Eventuell sollte die Backupkomprimierung nur in Tabellenbereichen erfolgen, die keine komprimierten Daten enthalten, wenn folgende Bedingungen zutreffen:

- Daten und Indexobjekte sind von LOB- und Langfelddaten getrennt und
- Sie verwenden die Zeilen- und Indexkomprimierung bei der Mehrheit Ihrer Datentabellen bzw. Indizes.

Um Ihre Backups zu komprimieren, verwenden Sie die Option `COMPRESS` im Befehl **`BACKUP DATABASE`**.

Komprimierung archivierter Protokolldateien

Ab DB2 Version 10.1 können archivierte Protokolldateien komprimiert werden. Diese Funktion reduziert in Verbindung mit der Daten-, Index- und Backupkomprimierung die Menge des erforderlichen Plattenspeicherplatzes in Ihrer Datenbankumgebung.

Archivierte Protokolldateien sind der drittgrößte Speicherkonsument bei aktualisierend wiederherstellbaren Datenbanken. Archivierte Protokolldateien enthalten ein beträchtliches Datenvolumen und diese Archive können schnell wachsen. Wenn sich geänderte Daten bereits in komprimierten Tabellen befinden, reduziert sich die Protokollierung durch das Einfügen komprimierter Satzimages in Protokollsätze. Die Komprimierung archivierter Protokolldateien trägt auch in solchen Umgebungen zu weiteren Speichereinsparungen bei.

Zur Verwendung der Komprimierung für Ihre archivierten Protokolldateien können Sie die Konfigurationsparameter **`logarchcompr1`** und **`logarchcompr2`** mithilfe des Befehls **`UPDATE DB CFG`** auf `ON` setzen.

Einschränkungen

- In den folgenden Situationen können archivierte Protokolldateien nicht komprimiert werden:
  - Die entsprechende Protokollarchivierungsmethode ist nicht auf `DISK`, `TSM` oder `VENDOR` gesetzt. Wenn die entsprechende Protokollarchivierungsmethode wie beschrieben definiert ist, werden die Protokolldateien physisch aus dem Pfad für aktive Protokolldateien oder dem Pfad für Protokollspiegelung verschoben.
  - Wenn die Komprimierung archivierter Protokolldateien aktiviert wird und die entsprechende Protokollarchivierungsmethode auf `OFF`, `LOGRETAIN` oder `USEREXIT` gesetzt ist, ist die Komprimierung wirkungslos. Aktualisierungen der Datenbankkonfigurationsparameter **`logarchmeth1`** und **`logarchmeth2`** oder

der Datenbankkonfigurationsparameter **logarchcompr1** und **logarchcompr2**, die ein solches Szenario zur Folge haben, geben die Warnung SQL1663W zurück.

Anmerkung: Wenn die Datenbank aktiviert ist, wird die Warnung SQL1663W beim Festlegen oder Ändern der Datenbankkonfigurationsparameter für die Komprimierung archivierter Protokolldateien nicht zurückgegeben. Stattdessen wird die Warnung SQL1363W zurückgegeben, bei der es sich um eine Nachricht mit einer höheren Priorität handelt. Falls die Datenbank nicht aktiviert ist, wird die Warnung SQL1663W zurückgegeben.

- Manuelles Archivieren und Abrufen mit **db2adutl**
  - Das Dienstprogramm **db2adutl** führt während der Operationen UPLOAD und EXTRACT keine Komprimierung oder Dekomprimierung durch. Das Verschieben komprimierter Protokolldateien in die bzw. aus der Archivposition wird von **db2adutl** vollständig unterstützt.
  - Wenn Protokolle mit dem Befehl **db2adutl** auf Tivoli Storage Manager hochgeladen werden und archivierte Protokolldateien komprimiert werden sollen, muss die Komprimierung aktiviert sein, wenn die Protokolle in der Plattenposition archiviert werden, bevor sie vom Befehl **db2adutl** ausgewählt werden. Werden komprimierte Protokolle mit dem Befehl **db2adutl** manuell abgerufen, erfolgt die Extrahierung beim ersten Zugriff.
- Die Komprimierung archivierter Protokolldateien wird bei Verwendung von Roheinheiten zur Datenbankprotokollierung nicht unterstützt.
  - Die Komprimierung archivierter Protokolldateien wird außerdem nicht unterstützt, wenn der Datenbankkonfigurationsparameter **logpath** oder **newlogpath** auf eine Roheinheit verweist. Aktualisierungen der Datenbankkonfiguration, die dazu führen, dass die Komprimierung archivierter Protokolldateien aktiviert wird, während der Datenbankkonfigurationsparameter **logpath** oder **newlogpath** auf Roheinheiten verweist, schlagen fehl (SQL1665N).
- Wenn die Komprimierung archivierter Protokolldateien mit den Datenbankkonfigurationsparametern **logarchcompr1** und **logarchcompr2** aktiviert wird, sind bereits in einem Backup-Image gespeicherte Protokolle nicht betroffen.

Gruppieren zusammengehöriger Daten

Das Zusammenfassen zusammengehörender Daten in Gruppen vereinfacht die Datenrecovery.

Im Lauf des Entwurfsprozesses für eine Datenbank entwickeln sich Erkenntnisse über die Beziehungen, die zwischen Tabellen bestehen. Solche Beziehungen können auf folgenden Ebenen bemerkbar machen:

- Auf der Anwendungsebene, wenn Transaktionen mehrere Tabellen aktualisieren
- Auf der Datenbankebene, wenn die referenzielle Integrität zwischen Tabellen zu gewährleisten ist oder sich Trigger einer Tabelle auf eine andere Tabelle auswirken

Diese Beziehungen sollten bei der Entwicklung eines Recoveryplans berücksichtigt werden. Wahrscheinlich ist es sinnvoll, voneinander abhängige Datengruppen gemeinsam zu sichern. Solche zusammengehörigen Datengruppen können entweder auf Tabellenbereichsebene oder auf Datenbankebene eingerichtet werden. Wenn die zusammengehörigen Datengruppen zusammen gesichert werden, können sie bis zu einem Punkt wiederhergestellt werden, an dem alle Daten konsistent sind. Dies ist

besonders wichtig, wenn Sie in der Lage sein wollen, für Tabellenbereiche eine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt (punktuelle Recovery) auszuführen.

Backup- und Restoreoperationen zwischen unterschiedlichen Betriebssystemen und Hardwareplattformen

DB2-Datenbanksysteme unterstützen einige Backup- und Restoreoperationen zwischen unterschiedlichen Betriebssystemen und Hardwareplattformen.

Die unterstützten Plattformen für DB2-Backup- und Restoreoperationen können in eine von drei Familien gruppiert werden:

- Big Endian: Linux und UNIX
- Little Endian Linux und UNIX
- Windows

Ein Datenbankbackup von einer Plattformfamilie kann nur auf einem anderen beliebigen System in derselben Plattformfamilie wiederhergestellt werden. Bei Windows-Betriebssystemen können Sie eine Datenbank, die in Version 9.5 erstellt wurde, in einem DB2 Version 9.7-Datenbanksystem wiederherstellen. Bei Linux- und UNIX-Betriebssystemen können Sie, sofern das Endian-Format (Big Endian oder Little Endian) der Backup- und Restore-Plattformen identisch ist, Backups wiederherstellen, die unter älteren Versionen erstellt wurden.

In der folgenden Tabelle sehen Sie die einzelnen Linux- und UNIX-Plattformen, die DB2 unterstützt; außerdem wird angezeigt, ob die Plattformen das Endian-Format Big Endian oder Little Endian aufweisen:

Tabelle 15. Endian-Formate der unterstützten Linux- und UNIX-Betriebssysteme (von DB2 unterstützt)

Plattform	Endian-Format
AIX	Big Endian
HP unter IA64	Big Endian
Solaris x64	Little Endian
Solaris SPARC	Big Endian
Linux unter zSeries	Big Endian
Linux unter pSeries	Big Endian
Linux unter IA-64	Little Endian
Linux auf AMD64 und Intel EM64T	Little Endian
Linux (32-Bit) unter x86	Little Endian

Auf dem Zielsystem muss dieselbe (oder eine höhere) Version des DB2-Datenbankprodukts installiert sein wie auf dem Quellensystem. Sie können für ein Backup keinen Restore durchführen, das auf einer Version des Datenbankprodukts für ein System erstellt wurde, das eine frühere Version des Datenbankprodukts ausführt. Beispiel: Sie können ein DB2 Version 9.5-Backup auf einem DB2 Version 9.7-Datenbanksystem wiederherstellen, nicht jedoch ein DB2 Version 9.7-Backup auf einem DB2 UDB Version 9.5-Datenbanksystem.

Anmerkung: Sie können eine Datenbank aus einem Backup-Image, das auf einer 32-Bit-Ebene erstellt wurde, in einer 64–Bit-Ebene wiederherstellen, nicht jedoch

umgekehrt. Für das Backup und die Wiederherstellung Ihrer Datenbanken sollten Sie die DB2-Dienstprogramme Backup und Restore verwenden. Das Verschieben einer Dateigruppe von einer Maschine auf eine andere wird nicht empfohlen, da dies zu einer Beeinträchtigung der Datenbankintegrität führen kann.

In Fällen, bei denen bestimmte Backup- und Restorekombinationen nicht zulässig sind, können Sie Tabellen zwischen DB2-Datenbanken verschieben und dazu andere Methoden verwenden:

- Befehl **db2move**
- Dienstprogramm **Export**, gefolgt vom Dienstprogramm **IMPORT** oder **LOAD**

Anmerkung: Datenbankkonfigurationsparameter werden auf ihre Standardwerte gesetzt, falls die im Backup verwendeten Werte außerhalb des zulässigen Bereichs für die Umgebung liegen, in der die Datenbank wiederhergestellt wird. Dies kann bei Speicheroptimierungsparametern der Fall sein, wenn 64-Bit-Datenbanken in 32-Bit-Instanzen wiederhergestellt werden.

Zusammenführung von Protokolldatenströmen und Protokolldateiverwaltung in einer DB2 pureScale-Umgebung

In einer DB2 pureScale-Umgebung verwaltet jedes Member seine eigene Gruppe von Transaktionsprotokolldateien (d. h. einen *Protokolldatenstrom*) auf der gemeinsam genutzten Platte, wobei jede Gruppe einen separaten Protokollpfad aufweist. Die Protokolldateien für ein Member enthalten den Verlauf aller Datenänderungen, die auf dem betreffenden Member vorgenommen wurden.

Mehrere Anwendungen, die gleichzeitig auf unterschiedliche Member zugreifen, generieren während der Laufzeit möglicherweise abhängige Transaktionen. Eine Abhängigkeit zwischen zwei Transaktionen kann auftreten, wenn beispielsweise beide Transaktionen dieselbe Zeile verändern. Damit die Protokollsätze wirksam interpretiert werden, muss der DB2-Datenserver die Datensätze aus allen Protokolldatenströmen untersuchen und die Reihenfolge der Datensätze sortieren, sodass die Reihenfolge der Aktualisierungen wiedergegeben wird, die zur Laufzeit aufgetreten sind. Das Sortieren der Reihenfolge wird als *Protokolldatenstromzusammenführung* bezeichnet. Für einige Operationstypen in einer DB2 pureScale-Umgebung sind Protokolldatenstromzusammenführungen erforderlich. Dazu gehören (unter anderen) die Recovery nach dem Absturz einer Gruppe, aktualisierende Recoverys der Datenbank und aktualisierende Recoverys von Tabellenbereichen.

Konfigurationsparameter für die Protokollierung in einer DB2 pureScale-Umgebung

In Tabelle 16 wird gezeigt, welche protokollbezogenen Datenbankkonfigurationsparameter global gültig sind und welche Parameter dynamisch aktualisierbar sind.

Tabelle 16. Protokollierungsbezogene Datenbankkonfigurationsparameter

Parameter	Global gültig?	Dynamisch aktualisierbar?
archretrydelay	Ja	Ja
blk_log_dsk_ful	Nein	Ja
failarchpath	Ja	Ja
logarchcompr1	Ja	Ja
logarchcompr2	Ja	Ja

Tabelle 16. Protokollierungsbezogene Datenbankkonfigurationsparameter (Forts.)

Parameter	Global gültig?	Dynamisch aktualisierbar?
logarchmeth1	Ja	Ja
logarchmeth2	Ja	Ja
logarchopt1	Ja	Ja
logarchopt2	Ja	Ja
logbufsz	Nein	Ja
logfilsiz	Ja	Nein
logprimary	Ja	Nein
logsecond	Ja	Ja
max_log	Nein	Ja
mirrorlogpath [1]	Ja	Nein
newlogpath [1]	Ja	Nein
num_log_span	Nein	Ja
numarchretry	Ja	Ja
overflowlogpath	Ja	Ja
softmax	Ja	Nein
vendoropt	Ja	Ja

[1] Das erste Member, das eine Verbindung zur Datenbank herstellt oder die Datenbank aktiviert, verarbeitet die Änderungen dieses Protokollpfadparameters. Der DB2-Datenbankmanager bestätigt, dass der Pfad vorhanden ist und sowohl Lese- als auch Schreibzugriff für diesen Pfad vorhanden sind. Außerdem werden memberspezifische Unterverzeichnisse für die Protokolldateien erstellt. Falls eine dieser Operationen fehlschlägt, weist der DB2-Datenbankmanager den angegebenen Pfad zurück und versetzt die Datenbank mithilfe des alten Pfads in den Onlinestatus. Falls der Datenbankmanager den angegebenen Pfad akzeptiert, wird der neue Wert an die übrigen Member weitergegeben. Falls ein Member fehlschlägt, während es versucht, zum neuen Pfad zu wechseln, schlagen nachfolgende Versuche zum Aktivieren der Datenbank oder zum Herstellen einer Verbindung zur Datenbank fehl und die Fehlernachricht SQL5099N wird zurückgegeben. Alle Member müssen denselben Protokollpfad verwenden.

Abrufen von Protokollen für eine Operation zur Protokolldatenstromzusammenführung in einer DB2 pureScale-Umgebung

In dem Pfad für abgerufene Protokolldateien wird ein Unterverzeichnis erstellt. Das Unterverzeichnis hat folgendes Format: *protokollpfad*/LOGSTREAM*xxxx*. Dabei steht *protokollpfad* für den Protokollpfad, den Überlaufprotokollpfad oder den Pfad für Protokollspiegelung und *xxxx* ist eine vierstellige Protokolldatenstromkennung. (Die Protokolldatenstromkennung stimmt nicht unbedingt mit der zugehörigen Member-ID überein.) Wenn ein Member innerhalb dieses Unterverzeichnisses das Abrufen eines Protokolls erfordert, erstellt der DB2-Datenbankmanager eine weitere Unterverzeichnisebene für abgerufene Protokolle der einzelnen Member. Wenn Sie beispielsweise den Überlaufprotokollpfad `/home/dbuser/overflow/` in einem System mit drei Membern angeben und eine Anwendung auf Member 0 Protokolle abrufen muss, deren Eigner andere Member sind, lautet der Pfad für Member 0 `/home/dbuser/overflow/NODE0000/LOGSTREAM0000`, und Unterverzeichnisse unter diesem Pfad enthalten abgerufene Protokolle, deren Eigner andere Member sind, wie im folgenden Beispiel gezeigt:

```
Member 0 retrieves its own logs here:
   /home/dbuser/overflow/NODE0000/LOGSTREAM0000/LOGSTREAM0000
Member 0 retrieves logs that belong to member 1 here:
   /home/dbuser/overflow/NODE0000/LOGSTREAM0000/LOGSTREAM0001
Member 0 retrieves logs that belong to member 2 here:
   /home/dbuser/overflow/NODE0000/LOGSTREAM0000/LOGSTREAM0002
```

Anmerkung: Fügen Sie nicht manuell Protokolldateien in diese Unterverzeichnisse für abgerufene Protokolle ein. Verwenden Sie stattdessen den Überlaufprotokollpfad, wenn Sie Protokolldateien manuell abrufen wollen.

Beim Lesen von archivierten Protokolldateien, deren Eigner andere Member sind, muss ein Member Protokolldateien möglicherweise in den zugehörigen Protokollpfad oder Überlaufprotokollpfad abrufen. In diesem Fall erstellt die Operation zur Protokolldatenstromzusammenführung nach Bedarf für jeden Protokolldatenstrom die Engine-Dispatchable-Unit (EDU) **db2logmgr**.

Wie zuvor erwähnt gibt es drei Pfade, die zum Speichern von Protokolldateien verwendet werden können, deren Eigner andere Member sind, wie in der folgenden Liste gezeigt:

1. Wenn Sie den Datenbankkonfigurationsparameter **overflowlogpath** festlegen, wird der Überlaufprotokollpfad verwendet.

   Tipp: Sie können die Befehlsoptionen **ROLLFORWARD DATABASE** und **RECOVER DATABASE** verwenden, um einen alternativen Überlaufprotokollpfad anzugeben. Die Werte dieser Optionen überschreiben die Datenbankkonfiguration für die Zwecke einer einzelnen Recoveryoperation.
2. Der primäre Protokollpfad
3. Wenn Sie den Datenbankkonfigurationsparameter **mirrorlogpath** festlegen, wird der Pfad für Protokollspiegelung verwendet.

Wenn der DB2-Datenbankmanager eine Protokolldatei nicht im ersten Pfad speichern kann , wird versucht, den nächstfolgenden Pfad in der Liste zu verwenden. Wenn keiner dieser Pfade verfügbar ist, gibt das Dienstprogramm, mit dem die Operation zur Protokolldatenstromzusammenführung aufgerufen wurde, einen Fehler zurück, der für dieses Dienstprogramm spezifisch ist.

Die Ausgabe des Befehls **GET DATABASE CONFIGURATION** in einer DB2 pureScale-Umgebung gibt jeden einzelnen Protokollpfad mit nachfolgendem Namen des Members an. Wenn beispielsweise der Pfad für Protokollspiegelung für Member 2 auf `/home/dbuser/mirrorpath/` festgelegt wurde, zeigt die Ausgabe `/home/dbuser/mirrorpath/NODE0000/LOGSTREAM0002` an.

Wenn Sie Protokolldateien, deren Eigner andere Member sind, manuell abrufen müssen, müssen Sie sicherstellen, dass der Datenbankmanager auf die Protokolldateien mithilfe derselben Verzeichnisstruktur zugreifen kann, die auch automatisch erstellt wird. Damit beispielsweise Protokolle von Member 2 im Überlaufprotokollpfad von Member 1 zur Verfügung stehen, platzieren Sie die Protokolle in das Verzeichnis `/home/dbuser/overflow/NODE0000/LOGSTREAM0001/LOGSTREAM0002`.

Abgerufene Protokolldateien werden automatisch gelöscht, wenn sie nicht mehr benötigt werden. Unterverzeichnisse, die während einer Operation zur Protokolldatenstromzusammenführung erstellt wurden, werden zur zukünftigen Verwendung beibehalten.

Ermittlung fehlender Protokolle während einer Operation zur Protokolldatenstromzusammenführung

Wenn Sie eine Protokolldatei, die für eine Recoveryoperation erforderlich ist, aus Versehen gelöscht, verschoben oder archiviert und verloren haben, können Sie eine aktualisierende Recovery der Datenbank bis zum letzten Konsistenzpunkt vor der verlorenen Protokolldatei durchführen.

Wenn der DB2-Datenbankmanager während einer Operation zur Protokolldatenstromzusammenführung feststellt, dass in einem der Protokolldatenströme eine Protokolldatei fehlt, wird ein Fehler zurückgegeben. Das Dienstprogramm zur aktualisierenden Recovery gibt die Fehlernachricht SQL1273N zurück; die Anwendungsprogrammierschnittstelle db2ReadLog gibt die Fehlernachricht SQL2657N zurück.

In Abb. 12 wird in einem Beispiel gezeigt, wie zwei Member Protokollsätze in die Protokolldateien in ihrem aktiven Protokolldatenstrom schreiben könnten. Jede Protokolldatei wird durch ein Feld dargestellt.

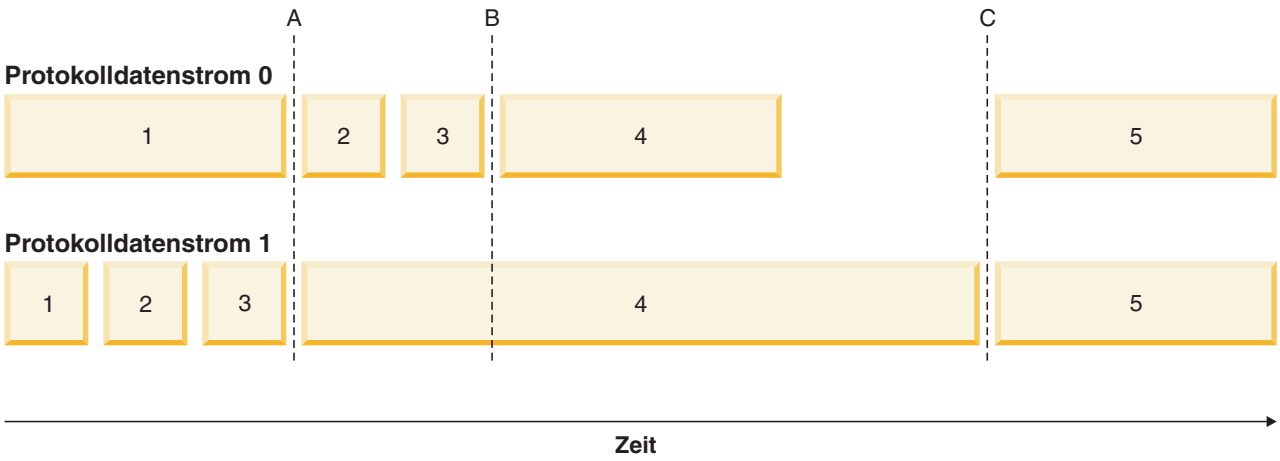


Abbildung 12. Protokolldateien in einer DB2 pureScale-Umgebung

Stellen Sie sich ein Szenario vor, in dem nur Protokolldatei 4 aus dem Protokolldatenstrom 1 fehlt, eine aktualisierende Recovery bis zum Zeitpunkt A erfolgreich ist, während aktualisierende Recoverys bis zum Zeitpunkt B, C oder bis zum Ende der Protokolle fehlschlagen. Der Befehl ROLLFORWARD gibt SQL1273N zurück, da Protokolldatei 4 nicht verfügbar ist. Außerdem wurden die Protokollsätze in den Dateien 2 und 3 im Protokolldatenstrom 0 in demselben Zeitraum geschrieben, in dem Protokolldatei 4 in Protokolldatenstrom 1 begonnen wurde. Daher kann die aktualisierende Recovery die Protokolldateien 2 und 3 erst bearbeiten, wenn Protokolldatei 4 aus Protokolldatenstrom 1 zur Verfügung gestellt wird. Das führt zu dem Ergebnis, dass die aktualisierende Recovery bei Zeitpunkt A stoppt und alle nachfolgenden aktualisierenden Recoverys können nicht über Zeitpunkt A hinaus fortfahren, bis Protokoll 4 aus Protokolldatenstrom 1 zur Verfügung gestellt wird.

Stellen Sie sich ein anderes Szenario vor, in dem während einer aktualisierenden Recovery nur Protokolldatei 4 aus Protokolldatenstrom 0 fehlt. Wenn Sie einen Befehl **ROLLFORWARD** mit der Option **END OF LOGS** absetzen (oder zu einem beliebigen Zeitpunkt nach Zeitpunkt B), stoppt die Operation bei Zeitpunkt B und gibt SQL1273N zurück, weil die Protokolldatei 4 in Protokolldatenstrom 0 fehlt. Eine aktualisierende Recovery kann Protokollsätze aus den Dateien 2 und 3 in Protokolldatenstrom 0 und einige Protokolle aus Datei 4 in Protokolldatenstrom 1 bis

zum Zeitpunkt B anwenden. Die aktualisierende Recovery muss bei Zeitpunkt B stoppen, auch wenn zusätzliche Protokolle aus Protokolldatenstrom 1 verfügbar sind, da es für die Zusammenführung von Protokollen erforderlich ist, dass alle Protokolle aus allen Protokolldatenströmen verfügbar sind.

Wenn Sie die fehlende Protokolldatei wieder auffinden können, stellen Sie sie zur Verfügung und setzen Sie den Befehl **ROLLFORWARD DATABASE** erneut ab. Wenn Sie die fehlende Protokolldatei nicht mehr auffinden können, setzen Sie den Befehl **ROLLFORWARD DATABASE...STOP** ab, um die aktualisierende Recoveryoperation am letzten Konsistenzpunkt vor der fehlenden Protokolldatei zu beenden.

Obwohl die Ermittlung fehlender Protokolle sicherstellt, dass eine Datenbankbeschädigung nicht als Ergebnis fehlender Protokolldateien auftreten wird, verhindert das Vorhandensein fehlender Protokolldateien, dass einige Transaktionen wiedergegeben werden und führt möglicherweise zu Datenverlust, wenn die fehlenden Protokolldateien nicht lokalisiert werden können.

Erforderliche Ressourcen

Für Operationen zur Protokolldatenstromzusammenführung sind zusätzliche EDUs erforderlich. Während der Datenbankaktivierung wird auf jedem Member eine Engine-Dispatchable-Unit (EDU) **db2lfr** erstellt. Wenn eine Operation zum Lesen eines Protokolls eingeleitet wird, die eine Protokolldatenstromzusammenführung erfordert, wird für jeden der Protokolldatenströme eine EDU **db2shred** und eine EDU **db2lfr** erstellt. Obwohl jede **db2lfr**-**db2shred**-Gruppe ihre eigene Gruppe von Puffern für Protokollseiten und Protokollsätze zuordnet, stellt dies keine bedeutende Menge an zusätzlichen Speicher- oder Systemressourcen dar. Jedem Member, das an der Protokolldatenstromzusammenführung beteiligt ist, werden ca. 400 KB zugeordnet.

Während einer Operation zur Protokolldatenstromzusammenführung ruft ein Member Protokolldateien, deren Eigner andere Member sind, in den zugehörigen Überlaufprotokollpfad, den primären Protokollpfad oder den Pfad für Protokollspiegelung ab. In einer DB2 pureScale-Umgebung müssen Sie vor dem Starten einer Operation zur aktualisierenden Recovery sicherstellen, dass im Abrufpfad ausreichend freier Plattenspeicherplatz vorhanden ist. Dadurch kann die Operation ohne Leistungsbeeinträchtigung den größten Teil der Dateien aus dem Archiv abrufen, wie es in einer DB2 pureScale-Umgebung erforderlich ist. Verwenden Sie die folgende Faustregel, um zu berechnen, wie viel Speicherplatz erforderlich ist, um die aktiven Protokolldateien für alle Member abzurufen: (**Anzahl primärer Protokolle** + **Anzahl sekundärer Protokolle**) * Anzahl der Member.

Beispiele

- Aktualisieren Sie den globalen Datenbankkonfigurationsparameter **newlogpath**:

```
db2 update db cfg for db mydb using newlogpath /home/dbuser/logdir
```

- Aktualisieren Sie den für jedes Member erforderlichen Datenbankkonfigurationsparameter **max_log** auf einem einzigen Member:

```
db2 update db cfg for db mydb member 1 using max_log 5
```

- Aktualisieren Sie den Pfad des primären Protokolls:

```
db2 connect to mydb
db2 update db cfg for mydb using newlogpath /home/dbuser/newlogpath
db2 get db cfg for mydb
...
Changed path to log files   (NEWLOGPATH) = /home/dbuser/newlogpath/NODE0000/LOGSTREAM0000/
Path to log files                        = /home/dbuser/dbuser/NODE0000/LOGSTREAM0000/
...
```

Die Änderung wird nicht wirksam, da das Member immer noch aktiv ist.

```
db2 terminate
db2 deactivate db mydb
db2 connect to mydb
db2 get db cfg for mydb
...
Changed path to log files   (NEWLOGPATH) =
Path to log files                       = /home/dbuser/newlogpath/NODE0000/LOGSTREAM0000/
...
```

Jedes Member verwendet den Protokollpfad /home/dbuser/newlogpath/NODE0000/LOGSTREAM*xxxx*, in dem *xxxx* für die Protokolldatenstrom-ID des Protokolldatenstroms steht, der den Pfad verwendet.

- Legen Sie einen neuen Pfad für das primäre Protokoll fest, während Sie ein Backup-Image wiederherstellen:

```
db2 restore db mydb newlogpath '/home/dbuser/newlogpath' without prompting
```

Protokollfolgenummern in DB2 pureScale-Umgebungen

DB2-Datenbanken verwenden die Protokollfolgenummer (LSN - Log Sequence Number), eine Kennung mit 64 Bit, um die Reihenfolge der Operationen zu ermitteln, durch die die Protokollsätze erstellt wurden.

Die Protokollfolgenummer ist ein ständig steigender Wert. Jedes Member schreibt in eine eigene Gruppe von Protokolldateien (den *Protokolldatenstrom*). Die Protokollfolgenummer ist innerhalb eines einzigen Protokolldatenstroms eine eindeutige Nummer.

Da Protokollfolgenummern auf jedem Member unabhängig voneinander generiert werden und mehrere Protokolldatenströme vorhanden sind, ist es möglich, dass Werte für Protokollfolgenummern in mehreren unterschiedlichen Protokolldatenströmen doppelt vorhanden sind. Die Protokollsatzkennung wird verwendet, um Protokollsätze protokolldatenstromübergreifend zu identifizieren. In der Datenbank werden den einzelnen Protokollsätzen in Protokolldatenströmen eindeutige Protokollsatzkennungen zugewiesen. Mit dem Befehl **db2pd** können Sie ermitteln, welche Protokollsatzkennung von einer Recoveryoperation verarbeitet wird.

Kapitel 8. Datei des Recoveryprotokolls

Mit jeder Datenbank wird eine Datei des Recoveryprotokolls (Recovery History File) erstellt. Diese Datei wird bei verschiedenen Operationen automatisch aktualisiert.

Die folgenden Operationen lösen eine Aktualisierung der Datei des Recoveryprotokolls aus:

- Sichern einer Datenbank oder von Tabellenbereichen
- Wiederherstellen einer Datenbank oder von Tabellenbereichen
- Aktualisierendes Wiederherstellen einer Datenbank oder von Tabellenbereichen
- Automatisches erneutes Erstellen einer Datenbank und Wiederherstellen mehrerer Images
- Erstellen eines Tabellenbereichs
- Ändern eines Tabellenbereichs
- Durchführen des Quiesce für einen Tabellenbereich
- Umbenennen eines Tabellenbereichs
- Löschen eines Tabellenbereichs
- Laden einer Tabelle
- Löschen einer Tabelle (wenn die Recovery gelöschter Tabellen aktiviert ist und die Protokollierung für Recovery verwendet wird)
- Reorganisieren einer Tabelle
- Aufrufen der bedarfsgesteuerten Archivierung von Protokolldateien
- Schreiben in eine neue Protokolldatei (wenn die Protokollierung für Recovery verwendet wird)
- Archivieren einer Protokolldatei (wenn die Protokollierung für Recovery verwendet wird)
- Wiederherstellen einer Datenbank

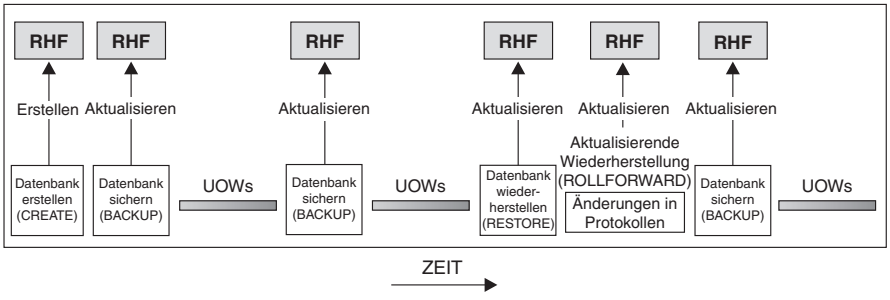


Abbildung 13. Erstellen und Aktualisieren der Datei des Recoveryprotokolls

Sie können die zusammengefassten Backup-Informationen in dieser Datei zur Recovery der gesamten Datenbank oder eines Teils der Datenbank bis zu einem bestimmten Zeitpunkt verwenden. Die Datei enthält folgende Informationen:

- Ein Identifikationsfeld zur eindeutigen Kennzeichnung der einzelnen Einträge
- Den Teil der Datenbank, der kopiert wurde, und Angaben zur Kopiermethode

- Den Zeitpunkt, an dem die Kopie erstellt wurde
- Die Speicherposition der Kopie (mit Informationen zur Einheit und logischen Möglichkeit für den Zugriff auf die Kopie)
- Den Zeitpunkt des letzten Restores
- Den Zeitpunkt, zu dem ein Tabellenbereich umbenannt wurde, sowie den vorherigen und den aktuellen Name des Tabellenbereichs
- Den Status der Backup-Operation: aktiv, inaktiv, abgelaufen oder gelöscht
- Die letzte Protokollfolgenummer, die vom Datenbank-Backup gespeichert oder von einer aktualisierenden Recovery verarbeitet wurde

Verwenden Sie den Befehl **LIST HISTORY**, wenn Sie die Einträge in der Datei des Recoveryprotokolls anzeigen wollen.

Jede Backup-Operation (sowohl für einen Tabellenbereich als auch für die gesamte Datenbank bzw. für ein inkrementelles Backup) schließt eine Kopie der Datei des Recoveryprotokolls mit ein. Die Datei des Recoveryprotokolls ist der Datenbank zugeordnet. Beim Löschen einer Datenbank wird auch die Datei des Recoveryprotokolls gelöscht. Beim Wiederherstellen einer Datenbank an einer neuen Position wird auch die Datei des Recoveryprotokolls wiederhergestellt. Beim Wiederherstellen wird die vorhandene Datei des Recoveryprotokolls nur dann überschrieben, wenn die auf dem Datenträger vorhandene Datei keine Einträge enthält. Ist dies der Fall, wird das Datenbankprotokoll aus dem Backup-Image wiederhergestellt.

Wenn die aktuelle Datenbank unbrauchbar oder nicht verfügbar ist und die zugehörige Datei des Recoveryprotokolls beschädigt oder gelöscht wurde, steht eine Option des Befehls **RESTORE** zur Verfügung, mit der lediglich die Datei des Recoveryprotokolls wiederhergestellt werden kann. Im Anschluss daran kann anhand der Informationen in der Datei des Recoveryprotokolls festgestellt werden, welches Backup zum Restore der Datenbank verwendet werden soll.

Die Größe dieser Datei wird mit dem Konfigurationsparameter **rec_his_retentn** gesteuert, der den Zeitraum (in Tagen) angibt, über den die Einträge in der Datei zu behalten sind. Auch wenn dieser Parameter auf den Wert null (0) gesetzt wurde, werden die Informationen zum aktuellen vollständigen Datenbank-Backup und der zugehörigen Restoregruppe beibehalten. (Diese Kopie kann nur über den Befehl **PRUNE HISTORY** mit der Option **FORCE** gelöscht werden.) Der Aufbewahrungszeitraum beträgt standardmäßig 366 Tage. Mit der Angabe -1 kann für diesen Zeitraum eine unbegrenzte Anzahl von Tagen definiert werden. In diesem Fall muss der Befehl PRUNE für die Datei explizit verwendet werden.

Eintragsstatus der Datei des Recoveryprotokolls

Der Datenbankmanager erstellt in der Datei des Recoveryprotokolls Einträge für Ereignisse wie z. B. eine Backup- oder Restoreoperation, das Erstellen eines Tabellenbereichs usw. Jeder Eintrag in der Datei des Recoveryprotokolls hat einen zugehörigen Status: aktiv (active), inaktiv (inactive), abgelaufen (expired), löschen anstehend (pending_delete), gelöscht (deleted) oder nicht löschen (do_not_delete).

Der Datenbankmanager verwendet den Status eines Eintrags in der Datei des Recoveryprotokolls, um festzustellen, ob die zu diesem Eintrag gehörenden physischen Dateien notwendig sind, um die Datenbank wiederherzustellen. Als Teil der automatischen Bereinigung aktualisiert der Datenbankmanager den Status der Einträge in der Datei des Recoveryprotokolls.

Aktive Datenbankbackups

Ein aktives Datenbankbackup ist ein Backup, das wiederhergestellt und mit aktuellen Protokollen aktualisierend wiederhergestellt werden kann, um wieder den aktuellen Status der Datenbank zu erhalten.

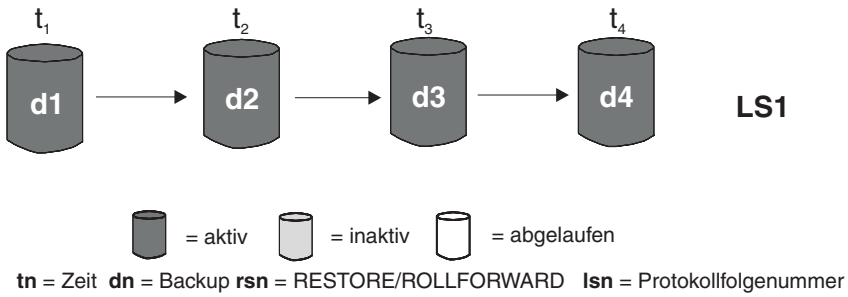


Abbildung 14. Aktive Datenbankbackups. Der Wert von **num_db_backups** wurde auf vier gesetzt.

Inaktive Datenbankbackups

Ein inaktives Datenbankbackup ist ein Backup, bei dessen Restore die Datenbank in einen früheren Zustand zurückversetzt wird.

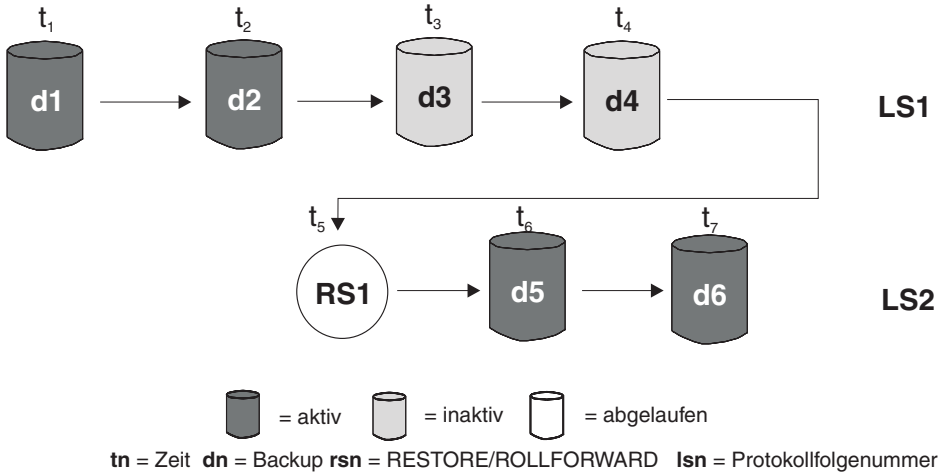


Abbildung 15. Inaktive Datenbankbackups

Abgelaufene Datenbankbackups

Ein abgelaufenes Datenbankbackup-Image wird nicht mehr benötigt, da neuere Backup-Images verfügbar sind.

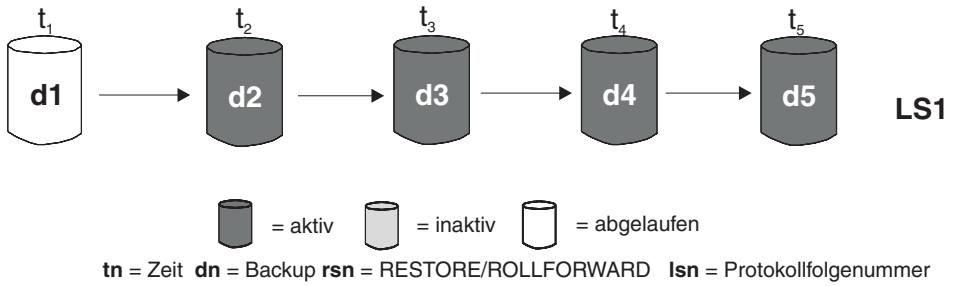


Abbildung 16. Abgelaufene Datenbankbackups

Einträge mit der Markierung do_not_delete

Sie können Einträge in der Datei des Recoveryprotokolls mit dem Befehl **PRUNE HISTORY** oder der API db2Prune entfernen (bereinigen). Der Datenbankmanager entfernt außerdem die Einträge in der Datei des Recoveryprotokolls als Teil der automatischen Bereinigung.

Es gibt lediglich drei Möglichkeiten, Einträge mit der Markierung do_not_delete zu entfernen:

- Rufen Sie den Befehl **PRUNE HISTORY** mit der Option **WITH FORCE** auf
- Rufen Sie die Prozedur ADMIN_CMD mit **PRUNE HISTORY** und der Option **WITH FORCE** auf
- Rufen Sie die API db2Prune mit der Option DB2_PRUNE_OPTION_FORCE auf

Die Einträge mit der Markierung do_not_delete werden ausschließlich dann aus der Datei des Recoveryprotokolls entfernt, wenn Sie eine dieser drei Aktionen ausführen.

Der Status der Einträge in der Datei des Recoveryprotokolls wird nicht vom Datenbankmanager auf do_not_delete gesetzt. Sie können den Status eines Eintrags in der Datei des Recoveryprotokolls mit dem Befehl **UPDATE HISTORY** auf do_not_delete setzen.

Einträge mit der Markierung delete_pending

Ein Eintrag mit der Markierung pending_delete wird gerade entfernt. Ein solcher Eintrag bleibt möglicherweise erhalten, wenn ein Löschvorgang vorzeitig beendet wurde. In einem solchen Fall ist die dem Eintrag zugeordnete Datei noch vorhanden (oder auch nicht), wird aber so behandelt, als ob sie nicht mehr existiert (wie bei gelöschten Einträgen).

Weitere Beispiele für den Status verschiedener Einträge in der Datei des Recoveryprotokolls:

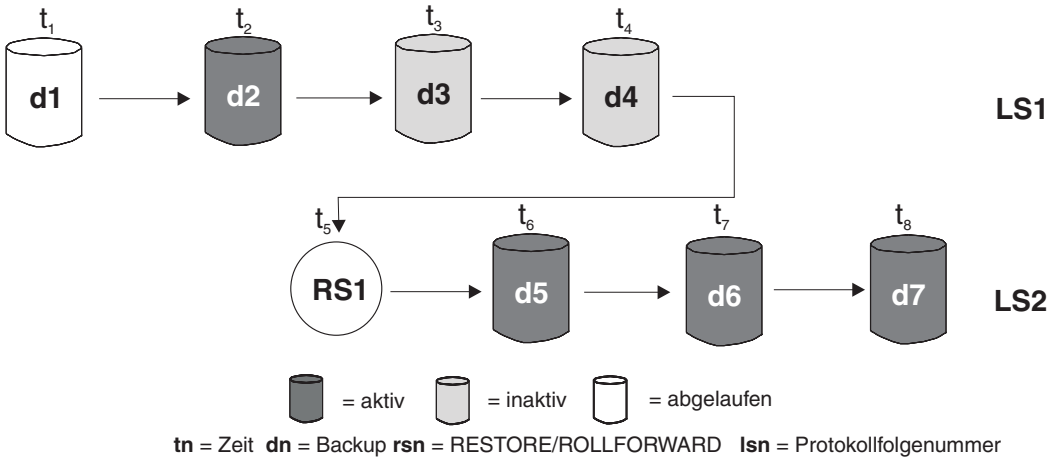


Abbildung 17. Gemischte aktive, inaktive und abgelaufene Datenbankbackups

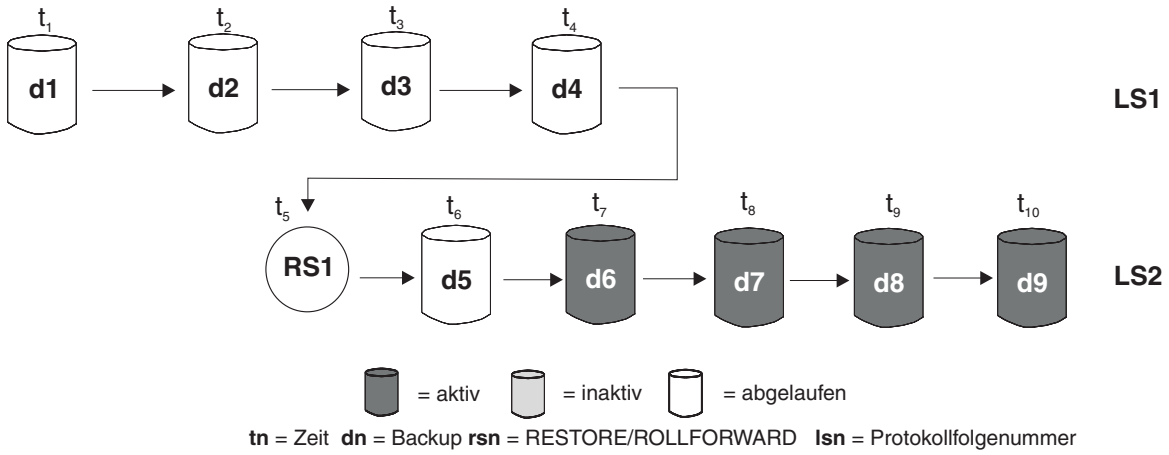


Abbildung 18. Abgelaufene Protokollfolge

Einträge in der Datei des Recoveryprotokolls mithilfe der Verwaltungssicht DB_HISTORY anzeigen

Mithilfe der Verwaltungssicht DB_HISTORY() können Sie auf den Inhalt der Datenbankprotokolldatei zugreifen. Diese Methode stellt eine Alternative zum CLP-Befehl **LIST HISTORY** bzw. den C-Protokoll-APIs dar.

Vorbereitende Schritte

Für die Verwendung dieser Funktion ist eine Datenbankverbindung erforderlich.

Informationen zu diesem Vorgang

Lösch- und Aktualisierungsoperationen in der Datenbankprotokolldatei können nur über die Befehle **PRUNE HISTORY** bzw. **UPDATE HISTORY** ausgeführt werden.

Vorgehensweise

Verwenden Sie die Verwaltungssicht DB_HISTORY() in einer SQL-Anweisung SELECT, um auf die Datenbankprotokolldatei für die Datenbank, zu der die Verbin-

dung besteht, oder für die Datenbankpartition, die durch die Umgebungsvariable **DB2NODE** angegeben wird, zuzugreifen. Geben Sie beispielsweise zum Anzeigen des Inhalts der Protokolldatei Folgendes ein:

```
SELECT * FROM TABLE(DB_HISTORY()) AS LIST_HISTORY
```

Beispiel

Um die Syntax der Verwaltungssicht zu verdecken, können Sie eine Sicht wie folgt erstellen:

```
CREATE VIEW LIST_HISTORY AS
        SELECT * FROM TABLE(DB_HISTORY()) AS LIST_HISTORY
```

Nach der Erstellung dieser Sicht können Sie Abfragen dafür ausführen. Beispiel:

```
SELECT * FROM LIST_HISTORY
```

oder

```
SELECT dbpartitionnum FROM LIST_HISTORY
```

oder

```
SELECT dbpartitionnum, start_time, seqnum, tabname, sqlstate
       FROM LIST_HISTORY
```

Eine Liste der von der Verwaltungssicht DB_HISTORY ausgegebenen Spalten und Spaltendatentypen finden Sie im Abschnitt Verwaltungssicht DB_HISTORY.

Bereinigen der Datei des Recoveryprotokolls

Der Datenbankmanager erstellt in der Datei des Recoveryprotokolls Einträge für Ereignisse wie z. B. eine Backup- oder Restoreoperation, das Erstellen eines Tabellenbereichs usw.

Wenn ein Eintrag in der Datei des Recoveryprotokolls nicht mehr relevant ist, weil die zugehörigen Recoveryobjekte nicht mehr für eine Recovery der Datenbank benötigt werden, empfiehlt es sich, diese Einträge aus der Datei des Recoveryprotokolls zu entfernen (sie zu *bereinigen*).

Vorgehensweise

Verwenden Sie die folgenden Methoden, um die Einträge in der Datei des Recoveryprotokolls zu bereinigen:

- Rufen Sie den Befehl **PRUNE HISTORY** auf
- Rufen Sie die API db2Prune auf
- Rufen Sie die Prozedur ADMIN_CMD mit dem Parameter PRUNE_HISTORY auf

Nächste Schritte

Wenn Sie eine dieser Methoden zum Bereinigen der Datei des Recoveryprotokolls verwenden, entfernt (bereinigt) der Datenbankmanager Einträge aus der Datei des Recoveryprotokolls, die älter als die von Ihnen angegebene Zeitmarke sind.

Wenn ein Eintrag in der Datei des Recoveryprotokolls den von Ihnen festgelegten Kriterien für das Bereinigen entspricht, dieser Eintrag jedoch noch für eine Recove-

ry der Datenbank benötigt wird, entfernt der Datenbankmanager diesen Eintrag nur, wenn Sie den Parameter **WITH FORCE** oder die Markierung DB2PRUNE_OPTION_FORCE verwenden.

Wenn Sie den Parameter **AND DELETE** oder die Markierung DB2PRUNE_OPTION_DELETE verwenden, werden auch Protokolldateien, die den bereinigten Einträgen zugeordnet sind, gelöscht.

Wenn Sie den Datenbankkonfigurationsparameter **AUTO_DEL_REC_OBJ** auf ON setzen und Sie den Parameter **AND DELETE** oder die Markierung DB2PRUNE_OPTION_DELETE verwenden, werden auch Protokolldateien, Backup-Images und Ladekopie-Images, die den bereinigten Einträgen zugeordnet sind, gelöscht.

Bereinigen der Datei des Recoveryprotokolls automatisieren

Sie können den Datenbankmanager so konfigurieren, dass der Status der Einträge in der Datei des Recoveryprotokolls automatisch bereinigt und aktualisiert wird.

Sie können den Status der Einträge in der Datei des Recoveryprotokolls manuell aktualisieren, indem Sie den Befehl **UPDATE HISTORY**, die API 'db2HistoryUpdate' oder die Prozedur ADMIN_CMD zusammen mit dem Parameter UPDATE_HISTORY verwenden. Sie können außerdem den Befehl **PRUNE HISTORY**, die API 'db2Prune' oder die Prozedur ADMIN_CMD mit dem Parameter PRUNE_HISTORY verwenden, um Einträge manuell in der Datei des Recoveryprotokolls zu entfernen (zu 'bereinigen'). Es wird jedoch empfohlen, den Datenbankmanager für die Verwaltung der Datei des Recoveryprotokolls zu konfigurieren, statt die Datei des Recoveryprotokolls manuell zu aktualisieren und zu bereinigen.

Der Datenbankmanager aktualisiert und bereinigt die Einträge in der Datei des Recoveryprotokolls automatisch zu folgenden Zeiten:

- Nachdem eine vollständige (nicht inkrementelle) Datenbanksicherungsoperation oder eine vollständige (nicht inkrementelle) Tabellenbereichsoperation erfolgreich abgeschlossen wurde
- Nachdem eine Datenbankwiederherstellungsoperation, bei der eine aktualisierende Recovery nicht erforderlich ist, erfolgreich abgeschlossen wurde
- Nachdem eine aktualisierende Recoveryoperation für die Datenbank erfolgreich abgeschlossen wurde

Während des automatisierten Bereinigens führt der Datenbankmanager zwei Operationen aus:

1. Aktualisierung des Status der Einträge in der Datei des Recoveryprotokolls
2. Bereinigung abgelaufener Einträge in der Datei des Recoveryprotokolls

Der Datenbankmanager aktualisiert die Einträge der Datei des Recoveryprotokolls auf folgende Art und Weise:

- Alle aktiven Datenbank-Backup-Images, die nicht mehr benötigt werden, werden als abgelaufen markiert.
- Alle Datenbankbackups, die als inaktiv markiert sind und vor dem Zeitpunkt des abgelaufenen Datenbankbackups gespeichert wurden, werden ebenfalls als abgelaufen markiert. Alle zugeordneten inaktiven Backup-Images von Tabellenbereichen und Lade-Backup-Kopien werden ebenfalls als abgelaufen markiert.
- Wenn ein aktives Datenbank-Backup-Image wiederhergestellt wird, es sich aber nicht um das aktuelle in der Protokolldatei aufgezeichnete Datenbank-Backup

handelt, werden alle nachfolgenden Datenbank-Backup-Images, die zur selben Protokollfolge gehören, als inaktiv markiert.

- Wenn ein inaktives Datenbank-Backup-Image wiederhergestellt wird, werden alle inaktiven Datenbankbackups, die zur aktuellen Protokollfolge gehören, wieder als aktiv markiert. Alle aktiven Datenbank-Backup-Images, die nicht länger zur aktuellen Protokollfolge gehören, werden als inaktiv markiert.
- Alle Datenbank- oder Tabellenbereichs-Backup-Images, die nicht der laufenden Protokollfolge (auch laufende Protokollkette genannt) entsprechen, werden als inaktiv markiert.

  Die laufende Protokollfolge wird durch das Datenbank-Backup-Image festgelegt, das wiederhergestellt wurde, und durch die verarbeiteten Protokolldateien. Nach dem Restore eines Datenbank-Backup-Images werden alle zu einem späteren Zeitpunkt erstellten Datenbank-Backup-Images als inaktiv markiert, da mit dem wiederhergestellten Image eine neue Protokollkette beginnt. (Dies trifft zu, wenn das Backup-Image ohne aktualisierende Recovery wiederhergestellt wurde. Wenn eine aktualisierende Recovery durchgeführt wurde, werden alle nach der Unterbrechung der Protokollkette erstellten Datenbankbackups als inaktiv markiert. Es ist vorstellbar, das ein älteres Datenbank-Backup-Image wiederhergestellt werden muss, da das Dienstprogramm zur aktualisierenden Recovery die Protokollfolge verwendet hat, die ein aktuelles beschädigtes Backup-Image enthält.)
- Das Backup-Image eines Tabellenbereichs wird inaktiv, wenn nach seinem Restore der aktuelle Status der Datenbank nicht durch Anwenden der aktuellen Protokollfolge erreicht werden kann.
- Alle Einträge mit dem Status do_not_delete werden nicht bereinigt, und ihre zugehörigen Protokolldateien, Backup-Images und Ladekopie-Images werden ebenfalls nicht gelöscht.
- Wenn für eine Datenbank ein Upgrade durchgeführt wird, werden in der Protokolldatei alle Backup-Einträge für die Datenbank im Onlinestatus und alle Backup-Einträge für den Tabellenbereich im Online- oder Offlinestatus als abgelaufen markiert, damit diese Einträge nicht von einer automatisierten REBUILD-Operation als Images für die erneute Erstellung ausgewählt werden können. Außerdem werden auch Ladekopie-Images und Protokollarchiveinträge als abgelaufen markiert, da diese Eintragstypen nicht für Recoveryzwecke verwendbar sind.

Die folgenden Datenbankkonfigurationsparameter steuern, welche Einträge vom Datenbankmanager bereinigt werden:

num_db_backups
Gibt die Anzahl der Datenbankbackups an, die für eine Datenbank beibehalten werden müssen

rec_his_retentn
Gibt die Anzahl an Tagen an, für die Langzeitinformationen für Backups aufbewahrt werden müssen

auto_del_rec_obj
Gibt an, ob der Datenbankmanager Protokolldateien, Backup-Images und Ladekopie-Images löschen soll, die zu den Einträgen in der Datei des Recoveryprotokolls gehören, welche bereinigt werden

Richten Sie die folgenden Konfigurationsparameter ein, um den Datenbankmanager für die automatisierte Verwaltung der Datei des Recoveryprotokolls zu konfigurieren:

- **num_db_backups**

- **rec_his_retentn**
- **auto_del_rec_obj**

Wenn der Parameter **auto_del_rec_obj** auf ON gesetzt wird und mehr erfolgreiche Datenbankbackup-Einträge als im Konfigurationsparameter **num_db_backups** festgelegt vorhanden sind, bereinigt der Datenbankmanager automatisch alle Einträge in der Datei des Recoveryprotokolls, die älter als die im Parameter **rec_his_retentn** festgelegte Anzahl von Tagen sind.

Einträge in der Datei des Recoveryprotokolls vor der Bereinigung schützen

Sie können verhindern, dass Schlüsseleinträge in der Datei des Recoveryprotokolls bereinigt (gelöscht) und zugehörige Recoveryobjekte gelöscht werden, indem Sie den Status der Einträge in der Datei des Recoveryprotokolls auf do_not_delete (Nicht löschen) setzen.

Informationen zu diesem Vorgang

Sie können Einträge in der Datei des Recoveryprotokolls mit dem Befehl **PRUNE HISTORY**, der Prozedur ADMIN_CMD mit PRUNE HISTORY oder der API db2Prune entfernen (bereinigen). Wenn Sie den Parameter **AND DELETE** zusammen mit **PRUNE HISTORY** oder die Markierung DB2PRUNE_OPTION_DELETE mit db2Prune verwenden und der Datenbankkonfigurationsparameter **auto_del_rec_obj** auf ON gesetzt wurde, werden die zugehörigen Recoveryobjekte ebenfalls physisch gelöscht.

Der Datenbankmanager entfernt außerdem die Einträge in der Datei des Recoveryprotokolls als Teil der automatischen Bereinigung. Wenn der Datenbankkonfigurationsparameter **auto_del_rec_obj** auf ON gesetzt wurde, löscht der Datenbankmanager die Recoveryobjekte, die etwaigen bereinigten Einträgen zugeordnet sind.

Vorgehensweise

Gehen Sie wie folgt vor, um Schlüsseleinträge in der Datei des Recoveryprotokolls sowie zugehörige Recoveryobjekte zu schützen:

Verwenden Sie den Befehl **UPDATE HISTORY**, die API db2HistoryUpdate oder die Prozedur ADMIN_CMD zusammen mit dem Parameter "UPDATE_HISTORY", um den Status von Schlüsseleinträgen in der Datei des Recoveryprotokolls auf 'do_no_delete' zu setzen.
Es gibt drei Möglichkeiten, Einträge mit der Markierung do_not_delete zu bereinigen:

- Rufen Sie den Befehl **PRUNE HISTORY** mit der Option **WITH FORCE** auf
- Rufen Sie die Prozedur ADMIN_CMD mit **PRUNE HISTORY** und der Option **WITH FORCE** auf
- Rufen Sie die API db2Prune mit der Option DB2_PRUNE_OPTION_FORCE **iOption** auf.

Die Einträge mit der Markierung do_not_delete werden nur dann aus der Datei des Recoveryprotokolls entfernt, wenn Sie eine dieser Prozeduren ausführen.
Einschränkungen:

- Sie können nur den Status von Backup-Images, Ladekopie-Images und Protokolldateien auf do_not_delete setzen.

- Der Status eines Backup-Eintrags wird nicht an Ladekopie-Images, nicht inkrementelle Backups oder Protokolldateien weitergegeben, die zu der Backup-Operation gehören. Wenn Sie einen bestimmten Datenbankbackup-Eintrag und seine zugeordneten Protokolldateieinträge sichern möchten, müssen Sie jeweils den Status für den Datenbankbackup-Eintrag und den Eintrag für alle zugehörigen Protokolldateien festlegen.

Kapitel 9. Verwalten von Recoveryobjekten

Wenn Sie regelmäßige Backups Ihrer Datenbank durchführen, können sich sehr große Datenbank-Backup-Images ansammeln, sowie zahlreiche Datenbankprotokolle und Images von Ladekopien. Der Datenbankmanager von IBM Data Server kann die Verwaltung dieser Recoveryobjekte vereinfachen.

Informationen zu diesem Vorgang

Das Speichern von Recoveryobjekten kann sehr viel Speicherplatz beanspruchen. Sobald neue Backup-Operationen ausgeführt werden, können Sie die älteren Recoveryobjekte löschen, da sie nicht mehr für ein Restore der Datenbank notwendig sind. Das Entfernen der älteren Recoveryobjekte kann jedoch zeitaufwendig sein. Außerdem kann es passieren, dass Sie beim Löschen älterer Recoveryobjekte versehentlich Recoveryobjekte löschen, die noch benötigt werden.

Vorgehensweise

Es gibt zwei Möglichkeiten, mit dem Datenbankmanager Recoveryobjekte zu löschen, die nicht mehr für ein Restore der Datenbank erforderlich sind:

- Sie können den Befehl **PRUNE HISTORY** mit dem Parameter **AND DELETE** oder die API db2Prune mit der Markierung DB2PRUNE_OPTION_DELETE aufrufen.
- Sie können den Datenbankmanager so konfigurieren, dass nicht benötigte Recoveryobjekte automatisch gelöscht werden.

Löschen von Datenbankrecoveryobjekten mit dem Befehl PRUNE HISTORY oder der API db2Prune

Sie können den Datenbankkonfigurationsparameter **auto_del_rec_obj** und den Befehl **PRUNE HISTORY** oder die Anwendungsprogrammierschnittstelle (API) db2Prune verwenden, um Recoveryobjekte zu löschen.

Informationen zu diesem Vorgang

Wenn Sie den Befehl **PRUNE HISTORY** verwenden oder die Anwendungsprogrammierschnittstelle db2Prune aufrufen, verhält sich der IBM Data Server-Datenbankmanager wie folgt:

- Löscht Einträge aus der Datei des Recoveryprotokolls, die nicht den Status DB2HISTORY_STATUS_DO_NOT_DEL aufweisen

Wenn Sie den Befehl **PRUNE HISTORY** mit dem Parameter **AND DELETE** oder die Anwendungsprogrammierschnittstelle db2Prune mit der Markierung DB2PRUNE_OPTION_DELETE aufrufen, verhält sich der Datenbankmanager wie folgt:

- Löscht Einträge aus der Datei des Recoveryprotokolls, die älter als eine von Ihnen festgelegte Zeitmarke sind und nicht den Status DB2HISTORY_STATUS_DO_NOT_DEL aufweisen
- Löscht die physischen Protokolldateien, die zu den bereinigten (gelöschten) Einträgen gehören

Wenn Sie den Datenbankkonfigurationsparameter **auto_del_rec_obj** auf ON setzen und Sie den Befehl **PRUNE HISTORY** mit dem Parameter **AND DELETE** oder die Anwen-

dungsprogrammierschnittstelle db2Prune mit der Markierung DB2PRUNE_OPTION_DELETE aufrufen, verhält sich der Datenbankmanager wie folgt:

- Löscht Einträge aus der Datei des Recoveryprotokolls, die nicht den Status DB2HISTORY_STATUS_DO_NOT_DEL aufweisen
- Löscht die physischen Protokolldateien, die zu den bereinigten (gelöschten) Einträgen gehören
- Löscht die Backup-Images, die den bereinigten (gelöschten) Einträgen zugeordnet sind
- Löscht die Images der Ladekopien, die den bereinigten (gelöschten) Einträgen zugeordnet sind

Vorgehensweise

Gehen Sie wie folgt vor, um nicht benötigte Recoveryobjekte zu löschen:

1. Setzen Sie den Datenbankkonfigurationsparameter **auto_del_rec_obj** auf ON.
2. Rufen Sie den Befehl **PRUNE HISTORY** mit dem Parameter **AND DELETE** oder die API db2Prune mit der Markierung DB2PRUNE_OPTION_DELETE auf.

Automatisieren der Verwaltung von Datenbankrecoveryobjekten

Sie können den Datenbankkonfigurationsparameter **auto_del_rec_obj** und die automatische Bereinigung der Datei des Recoveryprotokolls verwenden, um den IBM Data Server-Datenbankmanager so zu konfigurieren, dass er nicht benötigte Recoveryobjekte nach jeder vollständigen Datenbankbackup-Operation automatisch löscht.

Informationen zu diesem Vorgang

Nach jeder erfolgreichen vollständigen (nicht inkrementellen) Datenbankbackup-Operation entfernt der Datenbankmanager die Datei des Recoveryprotokolls in Übereinstimmung mit dem Wert der Konfigurationsparameter **num_db_backup** und **rec_his_retentn**:

- Sind mehr Datenbankbackup-Einträge in der Datei des Recoveryprotokolls vorhanden, als der Wert des Konfigurationsparameters **num_db_backup** zulässt, entfernt der Datenbankmanager alle Einträge aus der Datei des Recoveryprotokolls, die älter als der Wert des Konfigurationsparameters **rec_his_retentn** sind und nicht den Status DB2HISTORY_STATUS_DO_NOT_DEL aufweisen.

Wenn Sie den Datenbankkonfigurationsparameter **auto_del_rec_obj** auf ON setzen, führt der Datenbankmanager folgende Aktionen zusätzlich zum Löschen der Einträge in der Datei des Recoveryprotokolls aus:

- Löscht die physischen Protokolldateien, die zu den bereinigten (gelöschten) Einträgen gehören
- Löscht die Backup-Images, die den bereinigten (gelöschten) Einträgen zugeordnet sind
- Löscht die Images der Ladekopien, die den bereinigten (gelöschten) Einträgen zugeordnet sind

Wenn für die Berücksichtigung im aktuellen Recovery-Protokoll keine Datenbankgesamtbackup-Images verfügbar sind (z. B. weil sie niemals erstellt wurden), werden alle Images, die älter als der in **rec_his_retentn** angegebene Zeitraum sind, gelöscht.

Wenn der Datenbankmanager eine Datei nicht löschen kann, weil die Datei sich nicht mehr in der Speicherposition befindet, die in der Datei des Recoveryprotokolls aufgelistet ist, löscht der Datenbankmanager den Protokolleintrag.

Wenn der Datenbankmanager eine Datei nicht löschen kann, weil zwischen dem Datenbankmanager und dem Speichermanager oder der Speichereinheit ein Kommunikationsfehler aufgetreten ist, löscht der Datenbankmanager den Protokolleintrag nicht. Wenn der Fehler beseitigt worden ist, kann die betreffende Datei während der nächsten automatischen Bereinigung gelöscht werden.

Vorgehensweise

Gehen Sie wie folgt vor, um den Datenbankmanager so zu konfigurieren, dass nicht benötigte Recoveryobjekte automatisch gelöscht werden:

1. Setzen Sie den Datenbankkonfigurationsparameter **auto_del_rec_obj** auf ON.
2. Richten Sie die Konfigurationsparameter **rec_his_retentn** und **num_db_backups** so ein, dass die automatische Bereinigung der Datei des Recoveryprotokolls aktiviert ist.

Schutz von Recoveryobjekten vor dem Löschen

Die automatische Verwaltung von Recoveryobjekten spart Verwaltungszeit und Speicherbereich. Es kann jedoch sinnvoll sein, bestimmte Recoveryobjekte davor zu schützen, automatisch gelöscht zu werden. Sie können verhindern, dass Schlüsselrecoveryobjekte gelöscht werden, indem Sie den Status der zugehörigen Einträge in der Datei des Recoveryprotokolls auf do_not_delete (Nicht löschen) setzen.

Informationen zu diesem Vorgang

Wenn Sie den Datenbankkonfigurationsparameter **auto_del_rec_obj** auf ON setzen, werden Recoveryobjekte gelöscht, wenn ihre zugehörigen Einträge in der Datei des Recoveryprotokolls bereinigt (gelöscht) werden. Die Einträge in der Datei des Recoveryprotokolls werden bereinigt, wenn eines der folgenden Ereignisse eintritt:

- Sie rufen den Befehl **PRUNE HISTORY** zusammen mit dem Parameter **AND DELETE** auf
- Sie rufen die API db2Prune mit der Markierung DB2PRUNE_OPTION_DELETE auf
- Der Datenbankmanager bereinigt automatisch die Datei des Recoveryprotokolls, was nach jedem erfolgreichen Gesamtbackup eines Tabellenbereichs oder einer Datenbank geschieht.

Unabhängig davon, ob Sie den Befehl **PRUNE HISTORY** oder die API db2Prune aufrufen, oder den Datenbankmanager so konfigurieren, dass automatisch die Einträge in der Datei des Recoveryprotokolls bereinigt werden, werden Einträge mit der Markierung do_not_delete nicht bereinigt, und die zugehörigen Recoveryobjekte werden nicht gelöscht.

Einschränkungen

- Sie können nur den Status von Backup-Images, Ladekopie-Images und Protokolldateien auf do_not_delete setzen.
- Der Status eines Backup-Eintrags wird nicht an Ladekopie-Images, nicht inkrementelle Backups oder Protokolldateien weitergegeben, die zu der Backup-Operation gehören. Wenn Sie einen bestimmten Datenbankbackup-Eintrag und seine

zugeordneten Protokolldateieinträge sichern möchten, müssen Sie jeweils den Status für den Datenbankbackup-Eintrag und den Eintrag für alle zugehörigen Protokolldateien festlegen.

Vorgehensweise

Verwenden Sie den Befehl **UPDATE HISTORY**, um den Status von zugehörigen Einträgen in der Datei des Recoveryprotokolls auf do_no_delete zu setzen.

Verwalten von Momentaufnahmebackup-Objekten

Zur Verwaltung von Momentaufnahmebackup-Objekten müssen Sie den Befehl **db2acsutil** verwenden. Versetzen oder verschieben Sie keine Momentaufnahmebackup-Objekte mit Dateisystemdienstprogrammen.

Vorbereitende Schritte

Zur Ausführung von Backup- und Restoreoperationen für Momentaufnahmen benötigen Sie einen DB2 ACS-API-Treiber für Ihre Speichereinheit. Eine Liste der unterstützten Speicherhardware für den integrierten Treiber enthält die Tivoli-Dokumentation im Abschnitt Unterstützte Speichersubsysteme.

Bevor Sie mit der Verwaltung von Momentaufnahmebackup-Objekten beginnen können, müssen Sie DB2 Advanced Copy Services (ACS) aktivieren. Siehe hierzu „Aktivieren von DB2 Advanced Copy Services (ACS)" auf Seite 480.

Einschränkungen

Der Befehl **db2acsutil** wird derzeit nur unter AIX und Linux unterstützt.

Vorgehensweise

1. Verwenden Sie den Parameter **QUERY**, um eine Liste der verfügbaren Momentaufnahmebackup-Objekte aufzurufen.

   Verwenden Sie beispielsweise die folgende Syntax, um die verfügbaren Momentaufnahmebackup-Objekte für die Datenbankmanagerinstanz `dbminst1` aufzulisten:

   ```
   db2acsutil query instance dbminst1
   ```

2. Verwenden Sie den Parameter **STATUS**, um den Fortschritt einer bestimmten Momentaufnahmebackup-Operation zu überprüfen.

   Verwenden Sie beispielsweise die folgende Syntax, um den Fortschritt von Momentaufnahmebackup-Operationen anzuzeigen, die momentan möglicherweise auf der Datenbank `database1` ausgeführt werden:

   ```
   db2acsutil query status db database1
   ```

3. Verwenden Sie den Parameter **DELETE**, um ein bestimmtes Momentaufnahmebackup-Objekt zu löschen.

   Verwenden Sie beispielsweise die folgende Syntax, um alle Momentaufnahmebackup-Objekte für die Datenbank `database1` zu löschen, die älter als 10 Tage sind:

   ```
   db2acsutil delete older than 10 days ago db database1
   ```

Hochladen von Backup-Images und Protokolldateien in TSM

Sie haben die Möglichkeit, ein Backup in einer relativ schnellen Operation zunächst auf Platte vorzunehmen und das Backup-Image sowie die Protokolldateien zu einem späteren Zeitpunkt in Tivoli Storage Manager (TSM) hochzuladen. Hierbei bleiben die Informationen des Recoveryprotokolls genauso erhalten wie bei einem direkten Backup in TSM.

Diese Strategie ist vor allem in Situationen geeignet, in denen Backup-Images schneller erzeugt als von TSM geschrieben werden können.

Beispiel 1: Übernahmestrategie

Im Rahmen Ihres Recoveryplans beschließen Sie, zur Vereinfachung der Recovery eine bestimmte Gruppe von Images und Protokollen auf Platte beizubehalten. In einem festgelegten Intervall (in diesem Fall wöchentlich) werden die ältesten Protokolle und Images in TSM hochgeladen. Das Verfahren besteht darin, zunächst die Datei des Recoveryprotokolls nach dem ältesten Backup-Image abzufragen und anschließend dieses Image und seine Protokolle in TSM hochzuladen.

1. Sie fragen die Protokolldatei mit dem folgenden Befehl nach verfügbaren Protokollen und Images ab:

   ```
   db2 list history all for db sample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   List History File for sample

   Number of matching file entries = 100

   ...
   ...
   ...

   Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
   ---- ----- ---------------------------- ------- ----- --------------------
   X D 20110403134938 1 D S0000003.LOG C0000000
   ------------------------------------------------------------------------------
   Comment:
   Start Time: 20110403134938
   End Time: 20110403135204
   Status: A
   ------------------------------------------------------------------------------
   EID: 5 Location: /home/logdir/log1/diwu/SAMPLE/NODE0000/C0000000/S0000003.LOG

   ...
   ...
   ...

   Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
   ---- ----- ---------------------------- ------ ----- --------------------
   B D 20110404135750001 F D S0000000.LOG S0000007.LOG
   ------------------------------------------------------------------------------
   Contains 2 tablespace(s):

   00001 SYSCATSPACE
   00002 USERSPACE1
   ------------------------------------------------------------------------------
   Comment: DB2 BACKUP SAMPLE OFFLINE
   Start Time: 20110404135750
   End Time: 20110404135755
   Status: A
   ------------------------------------------------------------------------------
   ```

```
EID: 10 Location: /home/backupdir

...
...
...
```

2. Sie wählen die älteste Protokolldatei mit dem folgenden Befehl für das Hochladen aus:

   ```
   db2adutl upload logs between s3 and s3 db sample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   ======================================================
   | Upload Summary: |
   ======================================================

   1 / 1 logs were successfully uploaded

   Logs successfully uploaded:
   /home/logdir/log1/diwu/SAMPLE/NODE0000/C0000000/S0000003.LOG
   ```

3. Sie laden das älteste Protokoll und seine Protokolle mit dem folgenden Befehl hoch:

   ```
   db2adutl upload images taken at 20110404135750 with logs db sample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Match found, but S0000003.LOG is already on TSM

   ======================================================
   | Upload Summary: |
   ======================================================

   1 / 1 backup images were successfully uploaded
   4 / 4 logs were successfully uploaded

   Backup Images successfully uploaded:
   /home/backupdir/SAMPLE.0.diwu.NODE0000.CATN0000.20110404135750.001

   Logs successfully uploaded:
   /home/logdir/log1/diwu/SAMPLE/NODE0000/C0000001/S0000004.LOG
   /home/logdir/log1/diwu/SAMPLE/NODE0000/C0000001/S0000005.LOG
   /home/logdir/log1/diwu/SAMPLE/NODE0000/C0000002/S0000006.LOG
   /home/logdir/log1/diwu/SAMPLE/NODE0000/C0000002/S0000007.LOG
   ```

4. Sie prüfen die Ergebnisse:
   a. Durch eine Abfrage der Protokolldatei mit dem folgenden Befehl:

      ```
      db2 connect to sample
      ```

      Die folgenden Informationen werden zurückgegeben:

      ```
      Database server = DB2/LINUXX8664 9.7.5
      SQL authorization ID = DIWU
      Local database alias = SAMPLE
      ```

      ```
      db2 select OPERATION, OBJECTTYPE, START_TIME, SEQNUM, FIRSTLOG, LASTLOG, LOCATION,
       DEVICETYPE from table(ADMIN_LIST_HIST()) as T
      ```

      Die folgenden Informationen werden zurückgegeben:

      ```
      OPERATION OBJECTTYPE START_TIME SEQNUM FIRSTLOG LASTLOG LOCATION DEVICETYPE
      ---------------- ----------------- ----------------------- ------------- ------------------
      X D 20110403134938 - S0000003.LOG C0000000 adsm/libtsm.a A
      ...
      ...
      B D 20110404135750 1 S0000000.LOG S0000007.LOG adsm/libtsm.a A
      ```

   b. Durch eine Abfrage von TSM mit dem folgenden Befehl:

```
db2adutl query db sample
```

Die folgenden Informationen werden zurückgegeben:

```
Query for database SAMPLE


   Retrieving FULL DATABASE BACKUP information.
1 Time: 20110404135750 Oldest log: S0000007.LOG DB Partition Number: 0 Sessions: 1


   Retrieving INCREMENTAL DATABASE BACKUP information.
No INCREMENTAL DATABASE BACKUP images found for SAMPLE


   Retrieving DELTA DATABASE BACKUP information.
No DELTA DATABASE BACKUP images found for SAMPLE


   Retrieving TABLESPACE BACKUP information.
No TABLESPACE BACKUP images found for SAMPLE


   Retrieving INCREMENTAL TABLESPACE BACKUP information.
No INCREMENTAL TABLESPACE BACKUP images found for SAMPLE


   Retrieving DELTA TABLESPACE BACKUP information.
No DELTA TABLESPACE BACKUP images found for SAMPLE


   Retrieving LOAD COPY information.
No LOAD COPY images found for SAMPLE


   Retrieving LOG ARCHIVE information.
Log file: S0000003.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-04-21.38.28
Log file: S0000004.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-04-21.38.29
Log file: S0000005.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-04-21.38.30
Log file: S0000006.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-04-21.38.30
Log file: S0000007.LOG, Chain Num: 1, DB Partition Number: 0, Taken at: 2011-04-04-21.38.31
```

5. In der nächsten Woche laden Sie das älteste Backup-Image mit dem folgenden Befehl hoch:

```
db2adutl upload images taken at 20110409155645 with logs db sample
```

Die folgenden Informationen werden zurückgegeben:

```
Match found, but S0000003.LOG is already on TSM

============================================================
| Upload Summary: |
============================================================

1 / 1 backup images were successfully uploaded
2 / 2 logs were successfully uploaded

Backup Images successfully uploaded:
/home/backupdir/SAMPLE.0.diwu.NODE0000.CATN0000.20110409155645.001

Logs successfully uploaded:
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000000/S0000008.LOG
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000000/S0000009.LOG
```

6. Sie prüfen die Ergebnisse, indem Sie TSM mit dem folgenden Befehl abfragen:

```
db2adutl query db sample
```

Die folgenden Informationen werden zurückgegeben:

```
Query for database SAMPLE


   Retrieving FULL DATABASE BACKUP information.
1 Time: 20110404135750 Oldest log: S0000007.LOG DB Partition Number: 0 Sessions: 1
2 Time: 20110409155645 Oldest log: S0000009.LOG DB Partition Number: 0 Sessions: 1
```

```
   Retrieving INCREMENTAL DATABASE BACKUP information.
No INCREMENTAL DATABASE BACKUP images found for SAMPLE


   Retrieving DELTA DATABASE BACKUP information.
No DELTA DATABASE BACKUP images found for SAMPLE


   Retrieving TABLESPACE BACKUP information.
No TABLESPACE BACKUP images found for SAMPLE


   Retrieving INCREMENTAL TABLESPACE BACKUP information.
No INCREMENTAL TABLESPACE BACKUP images found for SAMPLE


   Retrieving DELTA TABLESPACE BACKUP information.
No DELTA TABLESPACE BACKUP images found for SAMPLE


   Retrieving LOAD COPY information.
No LOAD COPY images found for SAMPLE


   Retrieving LOG ARCHIVE information.
Log file: S0000003.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-04-21.38.28
Log file: S0000004.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-04-21.38.29
Log file: S0000005.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-04-21.38.30
Log file: S0000006.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-04-21.38.30
Log file: S0000007.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-04-21.38.31
Log file: S0000008.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-09-20.21.50
Log file: S0000009.LOG, Chain Num: 0, DB Partition Number: 0, Taken at: 2011-04-09-20.21.51
```

Beispiel 2: Hochladen und Entfernen eines lokalen Backup-Image

1. Sie erstellen mit dem folgenden Befehl ein Backup Ihrer Datenbank:

   ```
   db2 backup db sample to /home/backupdir
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Backup successful. The timestamp for this backup image is: 20110401135620
   ```

2. Zu einem späteren Zeitpunkt beschließen Sie, das Backup-Image hochzuladen und von der Platte zu entfernen. Hierzu verwenden Sie den folgenden Befehl:

   ```
   db2adutl upload and remove images taken at 20110401135620 db sample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   File /home/backupdir/SAMPLE.0.diwu.NODE0000.CATN0000.20110401135620.001 is uploaded successfully.
   Would you really like to remove the original file (Y/N)
   ```

3. Sie geben `Y` ein.

   Anmerkung: Falls Sie beim Hochladen keine Systemanfrage vor dem Entfernen des Backup-Image von der Platte erhalten möchten, verwenden Sie den folgenden Befehl:

   ```
   db2adutl upload and remove images taken at 20110401135620 db sample without prompting
   ```

Die folgenden Informationen werden zurückgegeben:

```
/home/backupdir/SAMPLE.0.diwu.NODE0000.CATN0000.20110401135620.001 is successfully removed.

======================================================
| Upload Summary: |
======================================================

1 / 1 backup images were successfully uploaded
```

```
Backup Images successfully uploaded:
/home/backupdir/SAMPLE.0.diwu.NODE0000.CATN0000.20110401135620.001
```

Beispiel 3: Hochladen eines Image ohne Zeitmarke

1. Es wird die Frage ausgegeben, ob Sie das neueste Image hochladen möchten oder nicht:

   ```
   Upload the most recent backup image?
   ```

2. Sie laden ein Backup-Image ohne Angabe einer Zeitmarke oder eines Dateinamens mit dem folgenden Befehl hoch:

   ```
   db2adutl upload images db sample
   ```

3. Sie geben `Y` ein.

Die folgenden Informationen werden zurückgegeben:

```
============================================================
| Upload Summary: |
============================================================
1 / 1 backup images were successfully uploaded
Backup Images successfully uploaded:
/home/backupdir/SAMPLE.0.diwu.NODE0000.CATN0000.20110401160128.001
```

Falls das neueste Backup-Image bereits in TSM vorhanden ist, werden die folgenden Informationen zurückgegeben:

```
The most recent image is already on TSM.
```

Beispiel 4: Hochladen eines bestimmten Image mit Protokollen

Sie wollen ein bestimmtes Backup-Image hochladen und seine Protokolle einschließen. Hierzu geben Sie den folgenden Befehl aus:

```
db2adutl upload images taken at 20110401155645 with logs db sample
```

Die folgenden Informationen werden zurückgegeben:

```
============================================================
| Upload Summary: |
============================================================

1 / 1 backup images were successfully uploaded
5 / 5 logs were successfully uploaded

Backup Images successfully uploaded:
/home/backupdir/SAMPLE.0.diwu.NODE0000.CATN0000.20110401155645.001

Logs successfully uploaded:
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000000/S0000000.LOG
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000000/S0000001.LOG
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000000/S0000002.LOG
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000000/S0000003.LOG
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000001/S0000004.LOG
```

Falls mit diesem Image eine bestimmte Gruppe von Protokollen hochgeladen werden soll, geben Sie den Bereich der Folgenummern wie im folgenden Befehl gezeigt an:

```
db2adutl upload images taken at 20110401155645 logs between s3 and s7 db sample
```

Die folgenden Informationen werden zurückgegeben:

```
============================================================
| Upload Summary: |
============================================================
```

```
1 / 1 backup images were successfully uploaded
5 / 5 logs were successfully uploaded

Backup Images successfully uploaded:
/home/backupdir/SAMPLE.0.diwu.NODE0000.CATN0000.20110401155645.001

Logs successfully uploaded:
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000000/S0000003.LOG
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000001/S0000004.LOG
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000001/S0000005.LOG
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000002/S0000006.LOG
/home/logdir/log1/diwu/SAMPLE/NODE0000/C0000002/S0000007.LOG
```

Kapitel 10. Überwachen des Fortschritts von Restoreoperationen

Mit dem **LIST UTILITIES** können Restoreoperationen für eine Datenbank überwacht werden.

Vorgehensweise

Setzen Sie den Befehl **LIST UTILITIES** ab und geben Sie den Parameter **SHOW DETAIL** an.

```
LIST UTILITIES SHOW DETAIL
```

Ergebnisse

Für Restoreoperationen wird kein geschätzter Anfangswert angegeben. Stattdessen wird UNKNOWN angegeben. Während die einzelnen Puffer aus dem Image gelesen werden, wird die tatsächliche Menge gelesener Byte aktualisiert. Für automatische inkrementelle Restoreoperationen, bei denen möglicherweise mehrere Images wiederhergestellt werden, wird der Fortschritt anhand von Phasen überwacht. Jede Phase stellt ein Image dar, das aus der Kette von inkrementellen Backups wiederhergestellt werden soll. Zu Beginn wird nur eine Phase angegeben. Nachdem das erste Image wiederhergestellt wurde, wird die Gesamtanzahl der Phasen angezeigt. Beim Wiederherstellen der einzelnen Images werden die Anzahl der abgeschlossenen Phasen und die Anzahl der verarbeiteten Byte aktualisiert.

Beispiel

Es folgt eine Beispielausgabe der Leistungsüberwachung einer Restoreoperation:

```
ID                                = 6
Typ                               = RESTORE
Datenbankname                     = SAMPLE
Partitionsnummer                  = 0
Beschreibung                      = db
Startzeit                         = 08/04/2011 12:24:47.494191
Status                            = Wird ausgeführt
Aufruftyp                         = Benutzer
   Fortschrittsüberwachung:
      Abgeschlossene Arbeit       = 4096 Byte
      Startzeit                   = 08/04/2011 12:24:47.494197
```

Kapitel 11. Backup - Übersicht

Sie können ein Backup der DB2-Datenbank und der zugehörigen gespeicherten Daten erstellen, um den Verlust von Daten im Falle eines Datenbankserviceausfalls zu verhindern. Für die Ausführung des Backupprozesses steht eine Reihe von Tools zur Verfügung.

In der einfachsten Form des DB2-Befehls **BACKUP DATABASE** müssen Sie lediglich den Aliasnamen der Datenbank angeben, die Sie sichern möchten. Beispiel:

```
db2 backup db sample
```

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Backup von Datenbanken. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Wird der Befehl erfolgreich ausgeführt, wird ein neues Backup-Image in dem Verzeichnis erstellt, in dem der Befehl abgesetzt wurde. Das Image wird in diesem Verzeichnis erstellt, da der Befehl in diesem Beispiel kein bestimmtes Zielverzeichnis angibt. Backup-Images, die mit DB2 Version 9.5 oder einer neueren Version erstellt werden, werden mit dem Dateimodus 600 generiert. Dies bedeutet, dass unter UNIX nur der Instanzeigner über Lese- und Schreibzugriffsrechte verfügt und dass unter Windows nur Mitglieder der Gruppe DB2ADMNS (und Administratoren) über Zugriff auf die Backup-Images verfügen.

Anmerkung: Wenn sich der DB2-Client und der -Server nicht auf demselben System befinden, ermitteln DB2-Datenbanksysteme das aktuelle Arbeitsverzeichnis auf der Clientmaschine und verwenden dieses Verzeichnis als Zielverzeichnis für das Backup auf dem Server. Aus diesem Grund wird empfohlen, dass Sie für das Backup-Image ein Zielverzeichnis angeben.

Backup-Images werden an der Zielposition erstellt, die Sie beim Aufrufen des Backup-Dienstprogramms angeben. Folgende Positionen sind möglich:

- Ein Verzeichnis (für Backups auf Platte oder Diskette)
- Eine Einheit (für Backups auf Band)
- Ein TSM-Server (TSM = Tivoli Storage Manager)
- Der Server eines anderen Lieferanten

Die Datei des Recoveryprotokolls wird jedes Mal automatisch mit den Ergebnisinformationen aktualisiert, wenn Sie eine Backup-Operation für eine Datenbank starten. Diese Datei wird im selben Verzeichnis wie die Datenbankkonfigurationsdatei erstellt.

Wenn Sie alte Backup-Images löschen wollen, die nicht länger benötigt werden, können Sie die Dateien entfernen, wenn die Backups in Form von Dateien gespeichert sind. Wenn Sie nachfolgend den Befehl **LIST HISTORY** mit der Option **BACKUP** ausführen, werden auch Informationen zu den gelöschten Backup-Images zurückgegeben. Um diese Einträge aus der Datei des Recoveryprotokolls zu entfernen, müssen Sie den Befehl **PRUNE** ausführen.

Wenn Ihre Recoveryobjekte mit Tivoli Storage Manager (TSM) gespeichert wurden, können Sie die Recoveryobjekte mithilfe des Dienstprogramms **'db2adutl'** abfragen, extrahieren, prüfen und löschen. Unter Linux und UNIX befindet sich dieses Dienstprogramm im Verzeichnis `sqllib/adsm`; bei Windows-Betriebssystemen finden Sie es im Verzeichnis `sqllib\bin`. Verwenden Sie für Momentaufnahmen das Dienstprogramm **db2acsutil** im Verzeichnis `sqllib/bin`.

Unter allen Betriebssystemen setzen sich die Dateinamen für auf Platte erstellte Backup-Images aus verschiedenen durch Punkte voneinander getrennten Elementen zusammen:

`DB_alias.Type.Inst_name.DBPART`*nnn*`.timestamp.Seq_num`

Beispiel:

`STAFF.0.DB201.DBPART000.19950922120112.001`

Datenbankaliasname (DB_alias)
Der aus 1 bis 8 Zeichen bestehende Datenbankaliasname, der beim Aufrufen des Backup-Dienstprogramms angegeben wurde.

Typ (Type)
Der Typ der Backup-Operation. Dabei gilt Folgendes: `0` steht für ein Backup der gesamten Datenbank, `3` steht für ein Backup auf Tabellenbereichsebene und `4` steht für ein Backup-Image, das mit dem Befehl **LOAD COPY TO** generiert wird.

Instanzname (inst_name)
Der aus 1 bis 8 Zeichen bestehende Name der aktuellen Instanz, der der Umgebungsvariablen **DB2INSTANCE** entnommen wird.

Nummer der Datenbankpartition (DBPART)
In Umgebungen mit nur einer Datenbankpartition lautet diese Nummer stets `DBPART000`. In Umgebungen mit partitionierten Datenbanken lautet die Nummer `DBPART`*xxx*, wobei *xxx* die Nummer angibt, die der Datenbankpartition in der Datei `db2nodes.cfg` zugeordnet ist.

Zeitmarke (timestamp)
Ein Wert aus 14 Zeichen, der das Datum und die Uhrzeit des Zeitpunkts angibt, zu dem das Backup durchgeführt wurde. Die Zeitmarke hat das Format *jjjjmmtthhnnss*. Dabei gilt Folgendes:

- *jjjj* steht für das Jahr (`1995` bis `9999`)
- *mm* steht für den Monat `01` bis `12`)
- *tt* steht für den Tag des Monats (`01` bis `31`)
- *hh* steht für die Stunde (`00` bis `23`)
- *mm* steht für die Minuten (`00` bis `59`)
- *ss* Steht für die Sekunden (`00` bis `59`)

Folgenummer (Seq_num)
Eine dreistellige Zahl, die als Dateierweiterung verwendet wird.

Wenn ein Backup-Image auf Band geschrieben wird, gilt Folgendes:

- Es werden keine Dateinamen erstellt, aber die zuvor beschriebenen Informationen werden im Backup-Header zu Prüfzwecken gespeichert.
- Es muss eine Bandeinheit über die Standardschnittstelle des Betriebssystems verfügbar sein. In eine großen Umgebung mit partitionierten Datenbanken ist es jedoch unter Umständen nicht praktisch, eine dedizierte Bandeinheit für jeden Datenbankpartitionsserver bereitzuhalten. Sie können die Bandeinheiten mit einem

oder mehreren TSM-Servern verbinden, sodass der Zugriff auf diese Bandeinheiten jedem Datenbankpartitionsserver ermöglicht wird.
- In einer Umgebung mit partitionierten Datenbanken können Sie darüber hinaus Produkte verwenden, die Funktionen für virtuelle Bandeinheiten implementieren, wie zum Beispiel REELlibrarian 4.2 oder CLIO/S. Mithilfe dieser Produkte können Sie auf eine mit anderen Knoten (Datenbankpartitionsservern) verbundene Bandeinheit über eine Pseudobandeinheit zugreifen. Der Zugriff auf die ferne Bandeinheit wird transparent ermöglicht, während der Zugriff auf die Pseudobandeinheit über die Standardschnittstelle des Betriebssystems erfolgen kann.

Ein Backup für eine Datenbank, die sich nicht im normalen Status oder im Status 'Backup anstehend' befindet, ist nicht möglich. Für einen Tabellenbereich, der sich im normalen Status oder im Status 'Backup anstehend' befindet, kann ein Backup durchgeführt werden. Wenn sich der Tabellenbereich nicht im normalen Status oder im Status 'Backup anstehend' befindet, kann ein Backup zugelassen werden.

Gleichzeitig ablaufende Backup-Operationen für denselben Tabellenbereich sind nicht zulässig. Sobald eine Backup-Operation für einen Tabellenbereich eingeleitet wurde, schlagen alle nachfolgenden Versuche fehl (SQL2048N).

Wenn eine Datenbank oder ein Tabellenbereich nur teilweise wiederhergestellt wurde, da während der Restoreoperation ein Systemabsturz auftrat, müssen Sie die Datenbank bzw. den Tabellenbereich erfolgreich wiederherstellen, bevor Sie ein Backup-Image davon erstellen können.

Das Backup schlägt fehl, wenn die Liste der zu sichernden Tabellenbereiche den Namen eines temporären Tabellenbereichs enthält.

Das Backup-Dienstprogramm ermöglicht die Steuerung des gemeinsamen Zugriffs für mehrere Prozesse, die Backupkopien verschiedener Datenbanken anlegen. Diese Steuerung des gemeinsamen Zugriffs gewährleistet, dass die Zieleinheiten für das Backup bis zur Beendigung aller Backup-Operationen geöffnet bleiben. Wenn während der Backup-Operation ein Fehler auftritt und ein geöffneter Container nicht geschlossen werden kann, erhalten andere Backup-Operationen mit demselben Ziellaufwerk eventuell Nachrichten über Zugriffsfehler. Zur Behebung etwaiger Zugriffsfehler müssen Sie die Backup-Operation, die den Fehler verursachte, beenden und die Verbindung zur Zieleinheit trennen. Wenn Sie das Backup-Dienstprogramm für gleichzeitig ablaufende Backup-Operationen auf Band verwenden, müssen Sie sicherstellen, dass diese Operationen nicht dasselbe Band ansteuern.

Anzeigen von Backup-Informationen

Mit dem Dienstprogramm **db2ckbkp** können Sie Informationen zu vorhandenen Backup-Images anzeigen. Dieses Dienstprogramm bietet folgende Möglichkeiten:
- Testen der Integrität des Backup-Images und Feststellen, ob ein Wiederherstellen möglich ist.
- Anzeigen der im Backup-Header gespeicherten Informationen.
- Anzeigen von Informationen zu den Objekten und des Protokolldateiheaders im Backup-Image.

Backup von Daten

Verwenden Sie den Befehl **BACKUP DATABASE**, um eine Kopie der Daten einer Datenbank zu erstellen und sie auf einem anderen Datenträger zu speichern. Die Backup-Daten können im Falle eines Verlusts der Originaldaten oder eines Fehlers verwendet werden.

Sie können eine gesamte Datenbank, eine Datenbankpartition oder nur ausgewählte Tabellenbereiche mit Backup sichern.

Vorbereitende Schritte

Es braucht noch keine Verbindung zu der zu sichernden Datenbank zu bestehen: Das Backup-Datenbankdienstprogramm stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der Backup-Operation beendet wird. Wenn bereits eine Verbindung zu der zu sichernden Datenbank besteht, wird diese Verbindung beim Absetzen des Befehls **BACKUP DATABASE** unterbrochen, und die Backup-Operation wird fortgesetzt.

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln. Das Backup-Image bleibt auf dem Datenbankserver, sofern Sie kein Speicherverwaltungsprodukt wie Tivoli Storage Manager (TSM) oder DB2 Advanced Copy Services (ACS) verwenden.

Wenn Sie ein Offline-Backup durchführen und Sie die Datenbank mit dem Befehl **ACTIVATE DATABASE** aktiviert haben, müssen Sie die Datenbank inaktivieren, bevor Sie das Offline-Backup durchführen. Wenn die Datenbank über aktive Verbindungen verfügt, muss ein Benutzer mit der Berechtigung SYSADM eine Verbindung zu der Datenbank herstellen und die folgenden Befehle absetzen, um die Datenbank erfolgreich zu inaktivieren:

```
CONNECT TO aliasname_der_datenbank
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
TERMINATE;
DEACTIVATE DATABASE aliasname-der-datenbank
```

In einer Umgebung mit partitionierten Datenbanken haben Sie die folgenden Möglichkeiten: Mit dem Befehl **BACKUP DATABASE** können Sie Datenbankpartitionen einzeln sichern, mit dem Befehlsparameter **ON DBPARTITIONNUM** können Sie mehrere der Datenbankpartitionen auf einmal sichern und mit dem Parameter **ALL DBPARTITIONNUMS** können Sie alle Datenbankpartitionen gleichzeitig sichern. Mit dem Befehl **LIST DBPARTITIONNUMS** können Sie die Datenbankpartitionen identifizieren, in denen Benutzertabellen enthalten sind, für die das Ausführen eines Backups sinnvoll ist.

Wenn Sie ein *Offline-Backup* in einer Umgebung mit partitionierten Datenbanken ausführen und kein SSV-Backup (SSV = Single System View, Einzelsystemsicht) verwenden, sollten Sie die Katalogpartition von allen anderen Datenbankpartitionen getrennt sichern. Zum Beispiel können Sie die Katalogpartition zuerst sichern und anschließend die anderen Datenbankpartitionen sichern. Diese Aktion ist notwendig, weil die Backup-Operation möglicherweise eine exklusive Datenbankverbindung mit der Katalogpartition erfordert, während deren die anderen Datenbankpartitionen keine Verbindung herstellen können. Wenn Sie ein *Online-Backup* ausführen, können alle Datenbankpartitionen (einschließlich der Katalogpartition) gleichzeitig und in beliebiger Reihenfolge gesichert werden.

Auf einem System, in dem mit verteilten Anforderungen gearbeitet wird, werden die Backup-Operationen auf die Datenbank für verteilte Anforderungen sowie auf die Metadaten angewendet, die im Katalog dieser Datenbank gespeichert sind (Wrapper, Server, Kurznamen usw.). Datenquellenobjekte (Tabellen und Sichten) werden nicht gesichert, es sei denn, diese Objekte werden in der Datenbank für verteilte Anforderungen gespeichert.

Wenn eine Datenbank mit einem vorherigen Release des Datenbankmanagers erstellt wurde, aber für sie noch kein Upgrade auf das aktuelle Release durchgeführt wurde, müssen Sie das entsprechende Upgrade für die Datenbank vornehmen. Andernfalls ist es nicht möglich, eine Backup-Kopie der Datenbank zu erstellen.

Einschränkungen

Für das Backup-Dienstprogramm gelten die folgenden Einschränkungen:

- Es ist nicht möglich, gleichzeitig ein Backup eines Tabellenbereichs und einen Restore eines Tabellenbereichs durchzuführen. Dies gilt auch dann, wenn Backup und Restore für unterschiedliche Tabellenbereiche erfolgen.
- Wenn Sie in der Lage sein wollen, in einer Umgebung mit partitionierten Datenbanken eine aktualisierende Recovery durchzuführen, müssen Sie die Datenbank auf den Knoten der Liste regelmäßig sichern. Außerdem müssen Sie über mindestens ein Backup-Image der übrigen Knoten im System verfügen (auch von Knoten, die keine Benutzerdaten für diese Datenbank enthalten). In zwei Situationen ist ein Backup-Image einer Datenbankpartition auf einem Datenbankpartitionsserver erforderlich, der keine Benutzerdaten für die Datenbank enthält:
  - Sie haben dem Datenbanksystem einen Datenbankpartitionsserver hinzugefügt, nachdem das letzte Backup erstellt wurde, und müssen eine aktualisierende Recovery auf diesem Datenbankpartitionsserver durchführen.
  - Die punktuelle Recovery wird verwendet, wozu alle Datenbankpartitionen im System den Status 'aktualisierende Recovery anstehend' aufweisen müssen.
- Online-Backup-Operationen für DMS-Tabellenbereiche sind mit den folgenden Operationen nicht kompatibel:
  - Laden
  - Reorganisation (online und offline)
  - Löschen von Tabellenbereichen
  - Tabellenverkürzung
  - Indexerstellung
  - Verwenden des Parameters NOT LOGGED INITIALLY (wird mit den Anweisungen CREATE TABLE und ALTER TABLE verwendet)
- Wenn Sie versuchen, ein Offline-Backup einer Datenbank durchzuführen, die momentan aktiv ist, empfangen Sie einen Fehler. Sie können vor einem Offline-Backup sicherstellen, dass die Datenbank nicht aktiv ist, indem Sie den Befehl **DEACTIVATE DATABASE** ausführen.

Vorgehensweise

Gehen Sie wie folgt vor, um das Backup-Dienstprogramm aufzurufen:

- Setzen Sie den Befehl **BACKUP DATABASE** im Befehlszeilenprozessor ab.
- Führen Sie die Prozedur ADMIN_CMD mit dem Parameter `BACKUP DATABASE` aus.
- Verwenden Sie die Anwendungsprogrammierschnittstelle (API) db2Backup.

- Öffnen Sie den Taskassistenten in IBM Data Studio für den Befehl **BACKUP DATABASE**.

Beispiel

Das folgende Beispiel zeigt einen Befehl **BACKUP DATABASE**, der über den CLP abgesetzt wird:

```
db2 backup database sample to c:\DB2Backups
```

Nächste Schritte

Wenn Sie ein Offline-Backup durchgeführt haben, müssen Sie nach der Ausführung die Datenbank reaktivieren:

```
ACTIVATE DATABASE sample
```

Durchführen eines Momentaufnahmebackups

Bei einer Momentaufnahmebackup-Operation wird zum Kopieren von Daten die leistungsstarke Kopiertechnologie einer Speichereinheit genutzt.

Vorbereitende Schritte

Zur Ausführung von Backup- und Restoreoperationen für Momentaufnahmen benötigen Sie einen DB2 ACS-API-Treiber für Ihre Speichereinheit. Eine Liste der unterstützten Speicherhardware für den integrierten Treiber enthält die Tivoli-Dokumentation im Abschnitt Unterstützte Speichersubsysteme.

Vor der Durchführung eines Momentaufnahmebackups muss DB2 Advanced Copy Services (ACS) aktiviert werden. Siehe hierzu: „Aktivieren von DB2 Advanced Copy Services (ACS)” auf Seite 480.

Einschränkungen

Einzelne Tabellenbereiche können mit Momentaufnahmebackups nicht wiederhergestellt werden.

Wenn Sie integrierte Momentaufnahmebackups verwenden, können Sie keinen umgeleiteten Restore durchführen. Bei einem FlashCopy-Restore wird die gesamte Datenträgergruppe mit allen Datenbankpfaden auf einen früheren Zeitpunkt zurückgesetzt.

Vorgehensweise

Verwenden Sie eine der folgenden Methoden, um ein Momentaufnahmebackup auszuführen:

- Setzen Sie den Befehl **BACKUP DATABASE** mit dem Parameter **USE SNAPSHOT** ab. Beispiel:

  ```
  db2 backup db sample use snapshot
  ```

- Rufen Sie die Prozedur ADMIN_CMD mit den Parametern **BACKUP DB** und **USE SNAPSHOT** auf. Beispiel:

  ```
  CALL SYSPROC.ADMIN_CMD
     ('backup db sample use snapshot')
  ```

- Rufen Sie die API db2Backup mit dem Datenträgertyp SQLU_SNAPSHOT_MEDIA auf. Beispiel:

```
int sampleBackupFunction( char dbAlias[],
                                                                  char user[],
                                                 char pswd[],
                          char workingPath[] )
{
  db2MediaListStruct mediaListStruct = { 0 };

  mediaListStruct.locations = &workingPath;
    mediaListStruct.numLocations = 1;
  mediaListStruct.locationType = SQLU_SNAPSHOT_MEDIA;


  db2BackupStruct backupStruct = { 0 };

  backupStruct.piDBAlias = dbAlias;
  backupStruct.piUsername = user;
  backupStruct.piPassword = pswd;
  backupStruct.piVendorOptions = NULL;
  backupStruct.piMediaList = &mediaListStruct;

  db2Backup(db2Version950, &backupStruct, &sqlca);

  return 0;
}
```

Verwenden einer geteilten Spiegeldatenbank als Backup-Image

Verwenden Sie die folgende Prozedur, um eine geteilte Spiegeldatenbank einer Datenbank an einer anderen Position in demselben System zu erstellen, die als Backup-Image außerhalb einer DB2 pureScale-Umgebung verwendet werden soll. Diese Prozedur kann anstelle von Datenbank-Backup-Operationen für die Datenbank verwendet werden.

Vorgehensweise

Gehen Sie wie folgt vor, um eine geteilte Spiegeldatenbank als Backup-Image zu verwenden:

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zur Primärdatenbank her:

   ```
   db2 connect to <datenbankname>
   ```

2. Setzen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl aus:

   ```
   db2 set write suspend for database
   ```

   Anmerkung: Führen Sie keine weiteren Dienstprogramme oder Tools aus, während sich die Datenbank im ausgesetzten Status befindet. Das Erstellen einer Datenbankkopie sollte die einzige Aktivität sein. Optional können Sie vor Eingabe des Befehls **SET WRITE** SUSPEND alle Pufferpools löschen, um das Recoveryfenster zu minimieren. Hierfür können Sie die Anweisung FLUSH BUFFERPOOLS ALL verwenden.

3. Erstellen Sie aus der Primärdatenbank mithilfe der entsprechenden Befehle auf Betriebssystem- bzw. Speicherebene mindestens eine geteilte Spiegeldatenbank.

   Anmerkung:

   - Stellen Sie sicher, dass Sie das gesamte Datenbankverzeichnis einschließlich des Datenträgerverzeichnisses kopieren. Sie müssen auch das Protokollverzeichnis und alle Containerverzeichnisse kopieren, die außerhalb des Datenbankverzeichnisses vorhanden sind. Wenn Sie mehrere Speichergruppen ver-

wenden, müssen Sie alle Pfade einschließlich Dateien und Unterverzeichnissen in diesen Pfaden kopieren. Eine Zusammenstellung dieser Informationen finden Sie in der Verwaltungssicht DBPATHS, in der alle Dateien und Verzeichnisse der Datenbank angezeigt werden, die geteilt werden müssen.

- Wenn Sie `EXCLUDE LOGS` im Befehl **`SET WRITE`** angegeben haben, dürfen Sie die Protokolldateien nicht in die Kopie einbeziehen.

4. Nehmen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl wieder auf:

   ```
   db2 set write resume for database
   ```

Falls ein Fehler im System auftritt, führen Sie die folgenden Schritte aus, um die Datenbank mithilfe der geteilten Spiegeldatenbank als Backup wiederherzustellen:

1. Stoppen Sie die Datenbankinstanz mit dem folgenden Befehl:

   ```
   db2stop
   ```

2. Kopieren Sie die geteilten Daten mit Befehlen auf Betriebssystemebene.

   Wichtig: Kopieren Sie nicht die geteilten Protokolldateien, da die ursprünglichen Protokolle für die aktualisierende Recovery benötigt werden.

3. Starten Sie die Datenbankinstanz mit folgendem Befehl:

   ```
   db2start
   ```

4. Initialisieren Sie die Primärdatenbank:

   ```
   db2inidb aliasname_der_datenbank as mirror
   ```

   Dabei steht *aliasname_der_datenbank* für den Aliasnamen der Datenbank.

5. Stellen Sie die Datenbank bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt aktualisierend wieder her, und beenden Sie das Restore.

Verwenden einer geteilten Spiegeldatenbank als Backup-Image in einer DB2 pureScale-Umgebung

Verwenden Sie die folgende Prozedur, um eine geteilte Spiegeldatenbank einer Datenbank an einer anderen Position in demselben System zu erstellen, die als Backup-Image in einer DB2 pureScale-Umgebung verwendet werden soll. Diese Prozedur kann anstelle von Datenbank-Backup-Operationen für die Datenbank verwendet werden.

Vorgehensweise

Gehen Sie wie folgt vor, um eine geteilte Spiegeldatenbank als Backup-Image zu verwenden:

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zur Primärdatenbank her:

   ```
   db2 connect to <datenbankname>
   ```

2. Konfigurieren Sie das General Parallel File System (GPFS) auf dem sekundären Cluster durch Extrahieren und Importieren der Einstellungen des primären Clusters. Führen Sie auf dem primären Cluster den folgenden GPFS-Befehl aus:

   ```
   mmfsctl dateisystem syncFSconfig -n datei-des-fernen-knotens
   ```

   Dabei steht *datei-des-fernen-knotens* für die Liste der Hosts im sekundären Cluster.

3. Setzen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl aus:

```
db2 set write suspend for database
```

Anmerkung: Führen Sie keine weiteren Dienstprogramme oder Tools aus, während sich die Datenbank im ausgesetzten Status befindet. Das Erstellen einer Datenbankkopie sollte die einzige Aktivität sein. Optional können Sie vor Eingabe des Befehls **SET WRITE SUSPEND** alle Pufferpools löschen, um das Recoveryfenster zu minimieren. Hierfür können Sie die Anweisung FLUSH BUFFERPOOLS ALL verwenden.

4. Ermitteln Sie mithilfe des folgenden Befehls, welche Dateisysteme ausgesetzt und kopiert werden müssen:
   ```
   db2cluster -cfs -list -filesystem
   ```
5. Setzen Sie mit folgendem Befehl alle GPFS-Dateisysteme aus, die Containerdaten oder Protokolldaten enthalten:
   ```
   /usr/lpp/mmfs/bin/mmfsctl dateisystem suspend-write
   ```

   Dabei steht *dateisystem* für ein Dateisystem, das Daten oder Protokolldaten enthält.

   Anmerkung: Während die GPFS-Dateisysteme ausgesetzt sind, sind Schreiboperationen blockiert. Führen Sie in diesem Zeitraum ausschließlich die Operationen für die geteilte Spiegeldatenbank aus, um den Zeitraum zu minimieren, in dem Operationen blockiert sind.
6. Erstellen Sie aus der Primärdatenbank mithilfe der entsprechenden Befehle auf Betriebssystem- bzw. Speicherebene mindestens eine geteilte Spiegeldatenbank.

   Anmerkung:
   - Stellen Sie sicher, dass Sie das gesamte Datenbankverzeichnis einschließlich des Datenträgerverzeichnisses kopieren. Sie müssen auch das Protokollverzeichnis und alle Containerverzeichnisse kopieren, die außerhalb des Datenbankverzeichnisses vorhanden sind. Wenn Sie mehrere Speichergruppen verwenden, müssen Sie alle Pfade einschließlich Dateien und Unterverzeichnissen in diesen Pfaden kopieren. Eine Zusammenstellung dieser Informationen finden Sie in der Verwaltungssicht DBPATHS, in der alle Dateien und Verzeichnisse der Datenbank angezeigt werden, die geteilt werden müssen.
   - Wenn Sie EXCLUDE LOGS im Befehl **SET WRITE** angegeben haben, dürfen Sie die Protokolldateien nicht in die Kopie einbeziehen.
7. Setzen Sie den Betrieb der ausgesetzten GPFS-Dateisysteme fort. Verwenden Sie dazu den folgenden Befehl für jedes ausgesetzte Dateisystem:
   ```
   /usr/lpp/mmfs/bin/mmfsctl dateisystem resume
   ```

   Dabei steht *dateisystem* für ein ausgesetztes Dateisystem, das Daten oder Protokolldaten enthält.
8. Nehmen Sie die Schreiboperationen für die Ein-/Ausgabe auf der Primärdatenbank mit folgendem Befehl wieder auf:
   ```
   db2 set write resume for database
   ```

Führen Sie die folgenden Schritte aus, falls eine Situation auftritt, in der es erforderlich ist, die geteilte Spiegeldatenbank als Backup-Image zu verwenden, um ein Restore für die Datenbank durchzuführen:

1. Stoppen Sie die Primärdatenbankinstanz mit folgendem Befehl:
   ```
   db2stop
   ```
2. Listen Sie die Cluster-Manager-Domäne mit folgendem Befehl auf:

```
db2cluster -cm -list -domain
```

3. Stoppen Sie den Cluster-Manager auf jedem Host im Cluster mit folgendem Befehl:

   ```
   db2cluster
   -cm -stop -host host -force
   ```

   Anmerkung: Sie müssen den Host, von dem aus Sie diesen Befehl absetzen, als Letztes beenden.
4. Stoppen Sie den GPFS-Cluster auf der Primärdatenbankinstanz mit dem folgenden Befehl:

   ```
   db2cluster -cfs -stop -all
   ```
5. Kopieren Sie die geteilten Daten der Primärdatenbank mithilfe entsprechender Befehle auf Betriebssystemebene.

   Wichtig: Kopieren Sie nicht die geteilten Protokolldateien, da die ursprünglichen Protokolle für die aktualisierende Recovery benötigt werden.
6. Starten Sie den GPFS-Cluster auf der Primärdatenbankinstanz mit folgendem Befehl:

   ```
   db2cluster -cfs -start -all
   ```
7. Starten Sie den Cluster-Manager mit folgendem Befehl:

   ```
   db2cluster -cm -start -domain domäne
   ```
8. Starten Sie die Datenbankinstanz mit folgendem Befehl:

   ```
   db2start
   ```
9. Initialisieren Sie die Primärdatenbank mit folgendem Befehl:

   ```
   db2inidb aliasname_der_datenbank as mirror
   ```
10. Stellen Sie die Primärdatenbank bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt aktualisierend wieder her, und beenden Sie das Restore.

Backup auf Band

Wenn Sie Ihre Datenbank oder Ihren Tabellenbereich sichern wollen, müssen Sie die Block- und die Puffergröße korrekt definieren. Dies gilt besonders dann, wenn Sie mit variablen Blockgrößen arbeiten (z. B. unter AIX, wenn die Blockgröße auf null gesetzt wurde).

Beim Sichern gilt eine Einschränkung für die Anzahl der festen Blockgrößen, die verwendet werden können. Diese Einschränkung besteht, weil DB2-Datenbanksysteme den Backup-Image-Header als 4-KB-Block ausgeben. Die einzigen festen Blockgrößen, die von DB2-Datenbanksystemen unterstützt werden, sind 512, 1024, 2048 und 4096 Byte. Wenn Sie mit einer festen Blockgröße arbeiten, können Sie für das Backup eine beliebige Puffergröße angeben. Es ist jedoch möglich, dass Ihre Backup-Operation nicht erfolgreich abgeschlossen werden kann, wenn die feste Blockgröße nicht mit einem der von DB2-Datenbanksystemen unterstützten Werte übereinstimmt.

Wenn Ihre Datenbank umfangreich ist und Sie eine feste Blockgröße verwenden, benötigen Ihre Backup-Operationen möglicherweise mehr Zeit als erwartet. Zur Verbesserung der Leistung können Sie eine variable Blockgröße verwenden.

Anmerkung: Wenn Sie eine variable Blockgröße verwenden, stellen Sie sicher, dass Sie über erprobte Verfahren verfügen, die Ihnen eine erfolgreiche Recovery (ein-

schließlich explizit angegebener Puffergrößen für die Befehle **BACKUP** und **RESTORE**) unter Verwendung von Backup-Images ermöglichen, die mit variabler Blockgröße erstellt wurden.

Bei der Verwendung variabler Blockgrößen müssen Sie eine Backup-Puffergröße angeben, die kleiner als der obere Grenzwert für die verwendete Bandeinheit oder gleich diesem Wert ist. Die Puffergröße muss dem oberen Grenzwert für die Blockgröße der verwendeten Einheit entsprechen, wenn Sie optimale Leistungswerte erzielen wollen.

Bevor eine Bandeinheit unter einem Windows-Betriebssystem verwendet werden kann, müssen Sie den folgenden Befehl absetzen:

```
db2 initialize tape on einheit using blkgröße
```

Dabei gilt Folgendes:

device Ein gültiger Bandeinheitenname. Der Standardwert bei Windows-Betriebssystemen ist `\\.\TAPE0`.

blkgröße
Der Blockungsfaktor für das Band. Er muss ein Faktor oder ein Vielfaches von 4096 sein. Der Standardwert ist die Standardblockgröße für die Einheit.

Wenn zum Restore ein mit variabler Blockgröße erstelltes Backup-Image verwendet wird, kann dies zu einer Fehlermeldung führen. In diesem Fall muss das Image eventuell mit einer passenden Blockgröße erneut geschrieben werden. Im Folgenden sehen Sie ein Beispiel für AIX:

```
tctl -b 0 -Bn -f /dev/rmt0 read > backup-datei.datei
   dd if=backup-datei.datei of=/dev/rmt0 obs=4096 conv=sync
```

Ein Speicherauszug des Backup-Images wird in einer Datei mit dem Namen `backup-datei.datei` erstellt. Der Befehl **dd** schreibt einen Speicherauszug des Images unter Verwendung einer Blockgröße von 4096 Byte zurück auf das Band.

Bei dieser Methode treten jedoch Probleme auf, wenn die Images zu umfangreich sind, um einen entsprechenden Speicherauszug in einer Datei zu speichern. Eine mögliche Lösung besteht in der Verwendung des Befehls **dd**. Mit diesem Befehl können Speicherauszüge von Images von einer Bandeinheit auf die andere gespeichert werden. Dieses Verfahren ist möglich, solange das Image nicht mehrere Bänder umfasst. Bei der Verwendung von zwei Bandeinheiten hat der Befehl **dd** folgendes Format:

```
dd if=/dev/rmt1 of=/dev/rmt0 obs=4096
```

Wenn der Einsatz von zwei Bandeinheiten nicht möglich ist, können Sie den Imagespeicherauszug mit dem Befehl **dd** auf einer Roheinheit speichern und ihn anschließend von dieser Roheinheit auf ein Band schreiben. Das Problem bei diesem Ansatz besteht darin, dass der Befehl **dd** die auf der Roheinheit gespeicherte Blockanzahl verfolgen *muss*. Diese Blockanzahl muss beim Zurückversetzen des Images auf ein Band angegeben werden. Wird der Befehl **dd** dazu verwendet, einen Imagespeicherauszug von einer Roheinheit auf ein Band zu schreiben, wird der gesamte Inhalt der Roheinheit auf das Band übertragen. Der Befehl **dd** kann nicht feststellen, wie viel Speicherplatz auf der Roheinheit zum Speichern des Images verwendet wird.

Wenn Sie das Backup-Dienstprogramm verwenden, müssen Sie den oberen Grenzwert für die Blockgröße der verwendeten Bandeinheit(en) kennen. Nachfolgend sind einige Beispiele aufgeführt:

Einheit	Anhang	Blockgrößen-grenzwert	Grenzwert für DB2-Puffergröße (in 4-KB-Seiten)
8 mm	scsi	131.072	32
3420	s370	65.536	16
3480	s370	61.440	15
3490	s370	61.440	15
3490E	s370	65.536	16
7332 (4 mm)[1]	scsi	262.144	64
3490e	scsi	262.144	64
3590[2]	scsi	2.097.152	512
3570 (Magstar MP)		262.144	64

Anmerkung:

1. Die Einheit 7332 implementiert keinen Blockgrößengrenzwert. Bei 256 KB handelt es sich lediglich um einen vorgeschlagenen Wert. Der geltende Blockgrößengrenzwert richtet sich nach dem übergeordneten Adapter.
2. Während auf der Einheit 3590 eine Blockgröße von 2 MB unterstützt wird, können Sie auch niedrigere Werte (z. B. 256 KB) ausprobieren, sofern die hierbei erzielten Leistungen Ihren Anforderungen gerecht werden.
3. Informationen zu den Grenzwerten Ihrer Einheit können Sie der Einheitendokumentation entnehmen bzw. beim Lieferanten der Einheit einholen.

Überprüfen der Kompatibilität der Bandeinheit

Bei UNIX-, Linux- und AIX-Betriebssystemen (ausschließlich) können Sie wie folgt bestimmen, ob Ihre Bandeinheit für eine Sicherung der DB2-Datenbanken unterstützt wird:

Führen Sie als Eigner der Datenbankmanagerinstanz den Betriebssystembefehl **dd** zum Lesen oder Beschreiben Ihrer Bandeinheit aus. Kann der Befehl **dd** erfolgreich ausgeführt werden, können Sie die DB2-Datenbanken auf der Bandeinheit sichern.

Backup in benannten Pipes

Unter UNIX-Betriebssystemen ist die Unterstützung für das Sichern in (und Wiederherstellen aus) lokalen benannten Pipes verfügbar.

Vorbereitende Schritte

Das Aus- und Eingabeprogramm der benannten Pipe müssen sich auf derselben Maschine befinden. Die Pipe muss in einem lokalen Dateisystem vorhanden sein. Da die benannte Pipe als lokale Einheit behandelt wird, muss nicht speziell angegeben werden, dass es sich bei dem Ziel um eine benannte Pipe handelt.

Vorgehensweise

1. Erstellen Sie eine benannte Pipe. Im Folgenden sehen Sie ein Beispiel für AIX:

```
mkfifo /u/dmcinnis/meinepipe
```

2. Wenn dieses Backup-Image vom Dienstprogramm RESTORE verwendet werden soll, muss die Restoreoperation *vor* der Backup-Operation aufgerufen werden, sodass alle Daten erfasst werden:

   ```
   db2 restore db sample from /u/dmcinnis/mypipe into mynewdb
   ```

3. Verwenden Sie diese Pipe als Ziel für die Backup-Operation einer Datenbank:

   ```
   db2 backup db sample to /u/dmcinnis/meinepipe
   ```

Backup von partitionierten Datenbanken

Ein Backup einer Datenbank in einer Umgebung mit partitionierten Datenbanken durchzuführen kann die folgenden Schwierigkeiten bereiten: Verfolgen, ob das Backup für alle Datenbankpartitionen erfolgreich war, Verwalten mehrerer Protokolldateien und Backup-Images und Sicherstellen, dass die Protokolldateien und Backup-Images für alle Datenbankpartitionen die erforderliche Mindestrecoveryzeit umfassen, die für ein Restore der Datenbank erforderlich ist. Ein SSV-Backup (SSV = single system view, Einzelsystemsicht) ist die einfachste Möglichkeit, für eine partitionierte Datenbank ein Backup durchzuführen.

Informationen zu diesem Vorgang

Es gibt drei Möglichkeiten, ein Backup für eine Datenbank in einer Umgebung mit partitionierten Datenbanken durchzuführen:

- Aufeinander folgendes Backup der einzelnen Datenbankpartitionen mit dem Befehl **BACKUP DATABASE**, dem Befehl **BACKUP DATABASE** zusammen mit der Prozedur ADMIN_CMD oder der API-Funktion db2Backup.
- Verwenden des Befehls **db2_all** zusammen mit dem Befehl **BACKUP DATABASE**, um für alle von Ihnen angegebenen Datenbankpartitionen ein Backup durchzuführen.
- Ausführen eines SSV-Backups, um für einige oder alle Datenbankpartitionen gleichzeitig ein Backup durchführen zu können.
- Verwenden Sie einen Taskassistenten in IBM Data Studio, um den Prozess des Datenbankbackups zu durchlaufen.

Ein Backup jeweils nur für eine Datenbankpartition auf einmal auszuführen, ist zeitaufwendig und fehlerträchtig. Ein Backup für alle Partitionen mit dem Befehl **db2_all** ist einfacher, als für alle Datenbankpartitionen einzeln ein Backup durchzuführen, da Sie generell nur einen Befehl aufrufen müssen. Wenn Sie jedoch **db2_all** für ein Backup einer partitionierten Datenbank verwenden, müssen Sie **db2_all** in manchen Fällen trotzdem mehrfach aufrufen, da für die Datenbankpartition, die den Katalog enthält, nicht gleichzeitig ein Backup mit Datenbankpartitionen ohne Katalog durchgeführt werden kann. Unabhängig davon, ob Sie jeweils nur für eine Datenbankpartition ein Backup durchführen oder **db2_all** verwenden, ist das Verwalten von Backup-Images, die mit einer der beiden Methoden erstellt wurden, schwierig, da die jeweilige Zeitmarke für das Backup-Image der Datenbankpartition unterschiedlich ist und das Koordinieren der Mindestrecoveryzeit über die Backup-Images der Datenbankpartitionen hinweg ebenfalls schwierig ist.

Aus den zuvor genannten Gründen wird empfohlen, ein Backup für eine Datenbank in einer Umgebung mit partitionierten Datenbanken als SSV-Backup durchzuführen.

Vorgehensweise

Gehen Sie wie folgt vor, um ein Backup für einige oder alle Datenbankpartitionen einer partitionierten Datenbank gleichzeitig in Form eines SSV-Backups durchzuführen:

1. Optional: Die Datenbank kann im Online-Status bleiben oder offline geschaltet werden.

   Während Sie für eine partitionierte Datenbank ein Backup durchführen, kann die Datenbank online oder offline geschaltet sein. Ist die Datenbank online, übernimmt das Backup-Dienstprogramm gemeinsam genutzte Verbindungen zu den anderen Datenbankpartitionen, sodass die Benutzeranwendungen in der Lage sind, eine Verbindung zu einer Datenbankpartition herzustellen, während für diese ein Backup durchgeführt wird.
2. Rufen Sie in der Datenbankpartition, die den Datenbankkatalog enthält, das Backup mit den geeigneten Parametern für partitionierte Datenbanken auf.
   - Führen Sie den Befehl **BACKUP DATABASE** mit dem Parameter **ON DBPARTITIONNUMS** aus.
   - Führen Sie den Befehl **BACKUP DATABASE** mit dem Parameter **ON DBPARTITIONNUMS** unter Verwendung der Prozedur ADMIN_CMD aus.
   - Rufen Sie die API db2Backup mit dem Parameter **iAllNodeFlag** auf.
   - Öffnen Sie den Taskassistenten in IBM Data Studio für den Befehl **BACKUP DATABASE**.
3. Optional: Beziehen Sie die für die Recovery erforderlichen Protokolldateien in die Backup-Images ein.

   Standardmäßig sind die Protokolldateien in Backup-Images enthalten, wenn Sie ein SSV-Backup durchführen (d. h. bei Angabe des Parameters **ON DBPARTITIONNUM**). Wenn Sie nicht möchten, dass die Protokolldateien in die Backup-Images einbezogen werden, verwenden Sie beim Ausführen des Backups den Befehlsparameter **EXCLUDE LOGS**. Bei anderen Backups als SSV-Backups sind die Protokolldateien standardmäßig nicht in den Backup-Images enthalten.

   Der Abschnitt „Einschließen von Protokolldateien in ein Backup-Image" auf Seite 184 enthält weitere Informationen.
4. Optional: Löschen Sie ältere Backup-Images. Die Methode, die Sie zum Löschen älterer Backup-Images verwenden, hängt davon ab, wie Sie die Backup-Images speichern. Wenn Sie z. B. die Backup-Images auf Platte speichern, können Sie die Dateien löschen; wenn Sie die Backup-Images mit Tivoli Storage Manager speichern, können Sie das Dienstprogramm **db2adutl** verwenden, um die Backup-Images zu löschen. Wenn Sie DB2 Advanced Copy Services (ACS) verwenden, können Sie mit **db2acsutil** Momentaufnahmebackup-Objekte löschen.

Backup partitionierter Tabellen mit IBM Tivoli Space Manager Hierarchical Storage Management

Das Hierarchical Storage Manager-Clientprogramm (HSM) von Tivoli Space Manager führt eine automatische Migration in Frage kommender Dateien auf sekundäre Speichereinheiten durch, um bestimmte Mindestgrößen an freiem Speicher in lokalen Dateisystemen sicherzustellen.

Bei einer Tabellenpartitionierung werden Tabellendaten auf mehrere Speicherobjekte verteilt, die als Datenpartitionen bezeichnet werden. HSM unterstützt ein Backup einzelner Datenpartitionen auf sekundären Speichereinheiten.

Bei SMS-Tabellenbereichen stellt sich jeder Datenpartitionsbereich als eine Datei im entsprechenden Verzeichnis dar. Daher ist eine Migration einzelner Datenbereiche (Datenpartitionen) auf sekundäre Speichereinheiten sehr einfach.

Bei DMS-Tabellenbereichen wird jeder Container durch eine Datei dargestellt. In diesem Fall empfiehlt es sich, weniger häufig genutzte Datenbereiche in einem eigenen Tabellenbereich zu speichern. Verwenden Sie bei der Eingabe der Anweisung CREATE TABLE mit der Klausel EVERY auch die Klausel NO CYCLE, um sicherzustellen, dass die Anzahl der in der IN-Klausel auf Tabellenebene aufgelisteten Tabellenbereiche mit der Anzahl der Datenpartitionen identisch ist, die erstellt werden. Das folgende Beispiel veranschaulicht diese Methode:

Beispiel 1

```
CREATE TABLE t1 (c INT) IN tbsp1, tbsp2, tbsp3 NO CYCLE
  PARTITION BY RANGE(c)
  (STARTING FROM 2 ENDING AT 6 EVERY 2);
```

Aktivieren des automatischen Backups

Eine Datenbank kann durch eine Vielzahl möglicher Hard- und Softwarefehler unbrauchbar werden. Die Bereithaltung eines möglichst aktuellen Gesamtbackups Ihrer Datenbank ist ein integraler Bestandteil der Planung und Implementierung einer Strategie zur Wiederherstellung eines Systems nach einem Katastrophenfall.

Verwenden Sie das automatische Datenbankbackup im Rahmen Ihrer Wiederherstellungsstrategie, um DB2 zu veranlassen, Ihre Datenbank ordnungsgemäß und regelmäßig mit BACKUP zu sichern.

Informationen zu diesem Vorgang

Sie können das automatische Backup über die Befehlszeilenschnittstelle oder die gespeicherte Systemprozedur AUTOMAINT_SET_POLICY konfigurieren. Sie müssen außerdem den Diagnoseanzeiger db.db_backup_req aktivieren, der standardmäßig aktiviert ist. Beachten Sie, dass nur aktive Datenbanken für die Auswertung herangezogen werden.

Vorgehensweise

- Setzen Sie die folgenden Datenbankkonfigurationsparameter auf `ON`, um das automatische Backup über die Befehlszeilenschnittstelle zu konfigurieren:
  - **`AUTO_MAINT`**
  - **`AUTO_DB_BACKUP`**
- Zur Konfiguration des automatischen Backups mit IBM Data Studio müssen Sie mit der rechten Maustaste auf die Datenbank klicken und dann den Taskassistenten auswählen, um das automatische Backup zu konfigurieren.
- Gehen Sie wie folgt vor, um das automatische Backup über die gespeicherte Systemprozedur AUTOMAINT_SET_POLICY zu konfigurieren:
  1. Erstellen Sie XML-Konfigurationseingaben, in denen Einzelheiten wie Backup-Datenträger angegeben werden, ob das Backup online oder offline erfolgen sollte, sowie die Häufigkeit des Backups.

     Sie können den Inhalt der Beispieldatei `DB2DefaultAutoBackupPolicy.xml` in das Verzeichnis `SQLLIB/samples/automaintcfg` kopieren und die XML so modifizieren, dass sie Ihren Konfigurationsanforderungen entspricht.
  2. Optional: Erstellen Sie eine XML-Eingabedatei, die Ihre XML-Konfigurationseingaben enthält.

3. Rufen Sie AUTOMAINT_SET_POLICY mit den folgenden Parametern auf:
   – Verwaltungstyp: AutoBackup
   – XML-Konfigurationseingaben: entweder ein großes Binärobjekt, das Ihren XML-Konfigurationseingabetext enthält, oder der Name der Datei, die Ihre XML-Konfigurationseingaben enthält.

   Weitere Informationen zur Verwendung der gespeicherten Systemprozedur AUTOMAINT_SET_POLICY finden Sie im Abschnitt „Konfigurieren einer Richtlinie für automatische Verwaltung mit SYSPROC.AUTOMAINT_SET-_POLICY oder SYSPROC.AUTOMAINT_SET_POLICYFILE" auf Seite 80.

Automatisches Datenbank-Backup

Eine Datenbank kann durch eine verschiedene Hard- und Softwarefehler unbrauchbar werden. Das automatische Datenbank-Backup vereinfacht die Verwaltung von Datenbank-Backup-Tasks für den Datenbankadministrator (DBA), indem sie fortlaufend sicherstellt, dass bei Bedarf ein neues Gesamtbackup der Datenbank durchgeführt wird.

Sie stellt auf der Grundlage eines oder mehrerer der folgenden Kriterien fest, ob eine Backup-Operation ausgeführt werden muss:

- Sie haben nie ein Gesamtbackup der Datenbank durchgeführt.
- Die abgelaufene Zeit seit dem letzten Gesamtbackup ist länger als die angegebene Anzahl von Stunden.
- Der Speicherplatz für Transaktionsprotokolle seit dem letzten Backup ist größer als die angegebene Anzahl von 4-KB-Seiten (nur im Protokollarchivierungsmodus).

Schützen Sie Ihre Daten, indem Sie eine Strategie zur Recovery nach einem Katastrophenfall für Ihr System planen und implementieren. Wenn dies Ihren Anforderungen entgegenkommt, können Sie die Funktion für das automatische Datenbank-Backup in Ihre Backup- und Recoverystrategie integrieren.

Wenn die Datenbank für die aktualisierende Recovery (Protokollarchivierung) eingerichtet ist, kann das automatische Datenbank-Backup entweder für das Online- oder das Offline-Backup aktiviert werden. Ansonsten ist nur das Offline-Backup verfügbar. Das automatische Datenbank-Backup unterstützt Platten, Bänder, Tivoli Storage Manager (TSM) und DLL-Datenträgertypen von Fremdanbietern.

Wenn ein Backup auf Platte ausgewählt wird, löscht die automatische Backup-Funktion die Backup-Images regelmäßig aus dem in der Konfiguration für das automatische Datenbankbackup angegebenen Verzeichnis. Daher steht zu einem bestimmten Zeitpunkt immer nur das letzte Backup-Image zur Verfügung, unabhängig von der Anzahl an Gesamtbackups, die in der Richtliniendatei für automatische Backups angegeben sind. Es wird empfohlen, dieses Verzeichnis ausschließlich für die automatische Backup-Funktion zu reservieren und nicht zur Speicherung anderer Backup-Images zu verwenden.

Die Funktion für das automatische Datenbank-Backup kann mithilfe der Datenbankkonfigurationsparameter **auto_db_backup** und **auto_maint** aktiviert bzw. inaktiviert werden. In einer Umgebung mit partitionierten Datenbanken wird das automatische Datenbank-Backup in jeder Datenbankpartition ausgeführt, wenn die Datenbankkonfigurationsparameter für die jeweilige Datenbankpartition aktiviert sind.

Sie können das automatische Backup auch mit den gespeicherten Systemprozeduren AUTOMAINT_SET_POLICY oder AUTOMAINT_SET_POLICYFILE konfigurieren.

Backup- und Restoreoperationen in einer DB2 pureScale-Umgebung

In einer DB2 pureScale-Umgebung wird durch einmaliges Absetzen eines Befehls **BACKUP DATABASE** oder **RESTORE DATABASE** für ein beliebiges Member eine Backup- oder Restoreoperation für alle Member eingeleitet.

Da eine DB2 pureScale-Umgebung nur eine einzige Datenbankpartition aufweisen kann, muss eine Backup-Operation nur eine einzige Datengruppe verarbeiten und generiert nur ein Backup-Image für die gesamte Gruppe. Im Falle der übrigen Member müssen nur die Metadaten und die Transaktionsprotokolle der Datenbank verarbeitet werden; diese sind in dem einen Backup-Image enthalten.

Ein Backup-Image enthält Daten aus den angegebenen Tabellenbereichen und sämtliche erforderliche Metadaten und Konfigurationsinformationen für alle aktuell definierten Member. Sie müssen für weitere Member in der DB2 pureScale-Instanz keine zusätzlichen Backup-Operationen ausführen. Darüber hinaus benötigen Sie lediglich einen einzigen Befehl **RESTORE DATABASE**, um die Datenbank und die memberspezifischen Metadaten aller Member wiederherzustellen. Sie müssen für weitere Member keine zusätzlichen Restoreoperationen ausführen, um den Cluster wiederherzustellen. Die Zeitmarken aufeinanderfolgender Backup-Images sind eindeutige, steigende Werte, unabhängig davon, welches Member sie generiert hat.

Alle Member müssen konsistent sein, bevor eine Offline-Backup-Operation eingeleitet werden kann. Es kann jeweils nur eine Offline-Backup-Operation ausgeführt werden, da das Backup-Dienstprogramm memberübergreifend absolut exklusiven Zugriff auf die Datenbank erfordert. Obwohl gleichzeitig ablaufende Online-Backup-Operationen unterstützt werden, können andere Backup-Operationen nicht gleichzeitig dieselben Tabellenbereiche kopieren, sondern müssen warten, bis sie an der Reihe sind.

Jegliche Lesevorgänge von Daten und Metadaten aus der Datenbank und sämtliche Schreibvorgänge in ein Backup-Image finden auf einem einzigen Member statt. Interaktionen zwischen Backup- oder Restoreoperationen und anderen Member sind auf das Kopieren oder Aktualisieren von Datenbankmetadaten (z. B. Tabellenbereichsdefinitionen, Protokolldateiheader und der Datenbankkonfiguration) beschränkt.

Anmerkung: Bevor Sie ein Backup ausführen, müssen Sie sicherstellen, dass für den Protokollarchivierungspfad ein gemeinsam genutztes Verzeichnis festgelegt ist, damit alle Member in der Lage sind, bei nachfolgenden aktualisierenden Recoverys auf die Protokolle zuzugreifen. Kann ein Member, für das die aktualisierende Recovery durchgeführt wird, nicht auf den Archivpfad zugreifen, wird die Nachricht SQL1273N zurückgegeben. Der folgende Befehl ist ein Beispiel dafür, wie das gemeinsam genutzte Verzeichnis als Protokollpfad festgelegt wird:

```
db2 update db cfg using logarchmeth1 DISK:/db2fs/gpfs1/svtdbm5/svtdbm5/ArchiveLOGS
```

(Hierbei ist *gpfs1* das gemeinsam genutzte Verzeichnis für die Member und *ArchiveLOGS* das eigentliche Verzeichnis, in dem die Protokolle archiviert werden.)

Online-Backup-Operationen können erfolgreich fortgeführt werden, wenn ein anderes Member offline ist, offline gesetzt wird oder wieder in den Onlinestatus versetzt wird, während die Operation ausgeführt wird (Tabelle 17 auf Seite 342). Ob-

wohl der Status der übrigen Member keine Auswirkung auf Restoreoperationen für die Datenbank hat, müssen Backup-Operationen möglicherweise für eine kurze Zeit warten, während die Recovery nach dem Absturz eines Members ausgeführt wird, das offline oder inkonsistent ist.

Tabelle 17. Auswirkungen des Status der übrigen Member in einer DB2 pureScale-Instanz bei Backup- und Restoreoperationen für die Datenbank

	Status übriger Member	
Operation	**Offline und konsistent**	**Offline und inkonsistent**
Online-Backup	Die Backup-Operation wird fortgeführt. Das andere Member kann nicht in den aktiven Status versetzt werden, während das Backup-Dienstprogramm gegen Beginn der Backup-Operation auf den Protokolldateiheader (LFH - Log File Header) zugreift oder während das Backup-Dienstprogramm gegen Ende der Backup-Operation auf den Protokolldatenstrom zugreift.	Die Backup-Operation wird fortgeführt, muss jedoch warten, bis die Recovery nach dem Absturz eines Members beendet ist und das andere Member entweder in den aktiven oder konsistenten Status versetzt wird. Das andere Member kann nicht in den aktiven Status versetzt werden, während das Backup-Dienstprogramm gegen Beginn der Backup-Operation auf den Protokolldateiheader (LFH) zugreift oder während das Backup-Dienstprogramm gegen Ende der Backup-Operation auf den Protokolldatenstrom zugreift.
Restore	Die Restoreoperation wird normal beendet.	Die Restoreoperation wird normal beendet.

Image- und Archivbenennung

Dateinamen für Backup-Images, die Sie auf Platte erstellen, setzen sich aus verschiedenen, durch Punkte voneinander getrennten Elementen zusammen:

datenbankaliasname.*typ*.*instanzname*.DBPART*nnn*.*zeitmarke*.*folgenr*

datenbankaliasname
Der Datenbankaliasname, den Sie beim Aufrufen des Backup-Dienstprogramms angegeben haben.

Typ Der Typ der Backup-Operation. Dabei gilt Folgendes: 0 steht für ein Backup der gesamten Datenbank, 3 steht für ein Backup eines Tabellenbereichs und 4 steht für ein Backup-Image, das durch den Befehl **LOAD** mit der Option **COPY** NO generiert wird.

instanzname
Der Name der aktuellen Instanz, der dem Wert der Umgebungsvariable **DB2INSTANCE** entspricht.

nnn Die Datenbankpartitionsnummer. In einer DB2 pureScale-Umgebung lautet diese Nummer stets '000'.

zeitmarke
Ein Wert aus 14 Zeichen, der das Datum und die Uhrzeit angibt, zu dem die Backup-Operation durchgeführt wurde. Die Zeitmarke hat das Format *jjjjmmtthhnnss*. Dabei gilt Folgendes:

- *jjjj* steht für das Jahr.
- *mm* steht für den Monat 01 bis 12).
- *tt* steht für den Tag des Monats (01 bis 31).
- *hh* steht für die Stunde (00 bis 23).
- *nn* steht für die Minuten (00 bis 59).
- *ss* steht für die Sekunden (00 bis 59).

folgenr Eine dreistellige Zahl, die als Dateierweiterung verwendet wird.

Beispiel:

```
SAMPLE.0.krodger.DBPART000.200802241234.001
```

Online-Backup mit INCLUDE LOGS

Eine Online-Backup-Operation mit der Option **INCLUDE LOGS** (dies ist der Standard) generiert ein Backup-Image, in dem die Protokolldateien enthalten sind, die erforderlich sind, um den Restore und die aktualisierende Recovery der Datenbank bis zur Mindestrecoveryzeit auszuführen. Wenn dieses Backup-Image anschließend verwendet wird, um (möglicherweise während einer Recovery nach einem Katastrophenfall) ein Restore in eine neue Datenbank durchzuführen und wenn daraufhin während einer nachfolgenden Operation zur aktualisierenden Recovery nur die Protokolle des Backup-Images zur Verfügung stehen, gibt ein Befehl **ROLLFORWARD DATABASE** mit dem Parameter **TO END OF LOGS** häufig eine Fehlernachricht bezüglich einer fehlenden Protokolldatei zurück (SQL1273N). Dies kann in einigen Situationen erwartet werden, da der Datenbankmanager möglicherweise entdeckt hat, dass nach der Backup-Operation zusätzliche Protokolle geschrieben wurden, diese Protokolle aber nicht für die aktuelle Operation zur aktualisierenden Recovery zur Verfügung stehen. Es kann auch der Fall eintreten, dass eines oder mehrere dieser Protokolle, die für die aktualisierende Recovery der Datenbank bis zu einem konsistenten Punkt erforderlich sind, fehlen. In beiden Fällen müssen Sie überprüfen, ob der Endpunkt der Operation zur aktualisierenden Recovery zulässig ist, und Sie müssen anschließend einen Befehl **ROLLFORWARD DATABASE** mit dem Parameter **AND STOP** eingeben. Wenn die aktualisierende Recovery die Mindestrecoveryzeit trotz der fehlenden Protokolldatei erreicht hat, müsste der Befehl **ROLLFORWARD DATABASE** mit dem Parameter **AND STOP** erfolgreich beendet werden. Ist dies nicht der Fall, wird die Fehlernachricht SQL1276N (die besagt, dass die aktualisierende Recovery mithilfe dieses Backup-Images nicht die Mindestrecoveryzeit erreicht hat) ausgegeben.

Recovery nach einem Katastrophenfall und Hochverfügbarkeit durch Protokollübertragung in einer DB2 pureScale-Umgebung

Bei der *Protokollübertragung* handelt es sich um das Kopieren ganzer Protokolldateien auf eine Bereitschaftsmaschine, entweder von einer Archivierungseinheit aus oder mithilfe eines Benutzerexitprogramms, das für die primäre Datenbank ausgeführt wird. Sie können eine Bereitschaftsdatenbank auf dem aktuellen Stand halten, indem Sie die Protokolle, die archiviert werden, auf diese Datenbank anwenden, oder Sie können die Datenbank- oder Tabellenbereichs-Backup-Images und die Protokollarchive am Bereitschaftsstandort lagern und einen Restore bzw. eine aktualisierende Recovery nur durchführen, wenn ein Katastrophenfall eingetreten ist. In beiden Fällen wird durch die aktualisierende Recovery am Bereitschaftsstandort möglicherweise festgestellt, dass eine oder mehrere Protokolldateien fehlen und es wird die Fehlernachricht SQL1273N ausgegeben. Überprüfen Sie, dass die aktualisierende Recovery eine zulässige Zeitmarke erreicht hat oder nehmen Sie entsprechende Aktionen vor, um das Problem zu beheben.

Wenn der DB2-Datenbankmanager während einer Operation zur Protokolldatenstromzusammenführung feststellt, dass in einem der Protokolldatenströme eine Protokolldatei fehlt, wird ein Fehler zurückgegeben. Das Dienstprogramm zur aktualisierenden Recovery gibt die Fehlernachricht SQL1273N zurück; die Anwendungsprogrammierschnittstelle db2ReadLog gibt die Fehlernachricht SQL2657N zurück. Wenn Sie auswählen, eine Bereitschaftsdatenbank auf dem aktuellen Stand zu halten, indem Sie die Protokolle, die archiviert werden, auf sie anwenden, wird durch aktualisierende Recoverys häufig festgestellt, dass einige Protokolle fehlen.

In Abb. 19 wird in einem Beispiel gezeigt, wie zwei Member Protokollsätze in die Protokolldateien in ihrem aktiven Protokolldatenstrom schreiben könnten. Jede Protokolldatei wird durch ein Feld dargestellt. Stellen Sie sich ein Szenario vor, indem aus Gründen der hohen Verfügbarkeit sowohl ein Primärstandort als auch ein Bereitschaftsstandort konfiguriert wurden. Ein Befehl **ROLLFORWARD DATABASE** mit der Option **END OF LOGS** wird an dem Bereitschaftsstandort an den Zeitpunkten A, B und C versucht. Für jeden beliebigen Zeitpunkt sind alle vor diesem Zeitpunkt geschlossenen Protokolldateien archiviert worden und sind am Bereitschaftsstandort zugänglich. Andernfalls ist die Protokolldatei am Primärstandort noch aktiv und sie steht noch nicht für den Bereitschaftsstandort zur Verfügung (wie für Protokolldatei 4 im Protokolldatenstrom 1 zum Zeitpunkt B dargestellt).

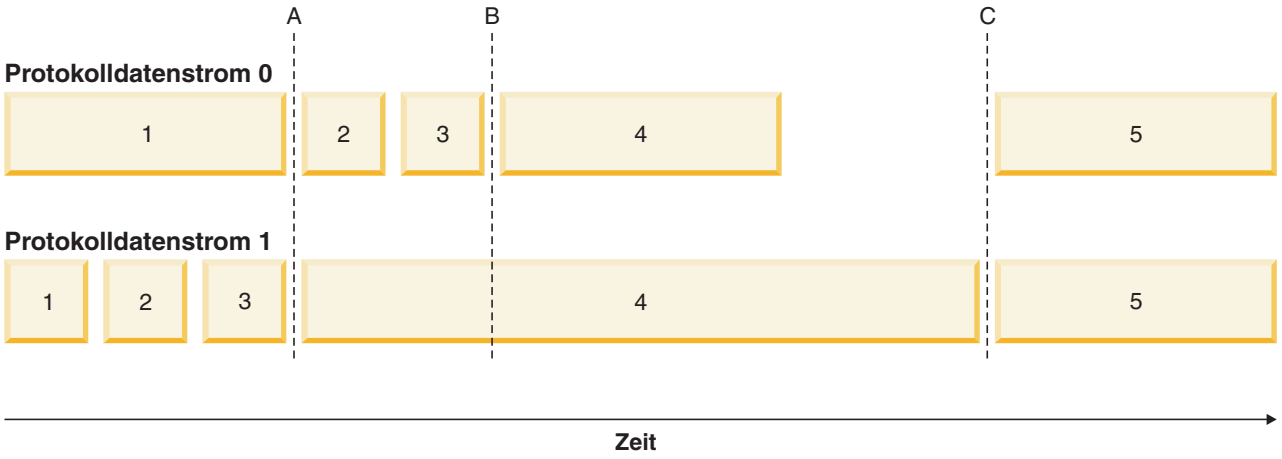


Abbildung 19. Protokolldateien in einer DB2 pureScale-Umgebung

Zum Zeitpunkt A wird der Befehl **ROLLFORWARD DATABASE** erfolgreich abgeschlossen, da die Protokolldatei 1 des Protokolldatenstroms 0 zum selben Zeitpunkt wie Protokolldatei 3 des Protokolldatenstroms 1 geschlossen und archiviert wurde. Zum Zeitpunkt B gibt der Befehl **ROLLFORWARD DATABASE** jedoch SQL1273N zurück. Dies geschieht, da zu dem Zeitpunkt, zu dem der Befehl am Bereitschaftsstandort abgesetzt wird, der Bereitschaftsstandort über Zugriff auf die Protokolldateien 2 und 3 des Protokolldatenstroms 0 verfügt, jedoch nicht auf die Protokolldatei 4 des Protokolldatenstroms 1, da die Protokolldatei noch am primären Standort offen und aktiv ist. Darüber hinaus wurden die Protokollsätze in den Dateien 2 und 3 im Protokolldatenstrom 0 im selben Zeitraum geschrieben, in dem Protokolldatei 4 in Protokolldatenstrom 1 begonnen wurde. Daher kann die aktualisierende Recovery die Protokolldateien 2 und 3 erst bearbeiten, wenn Protokolldatei 4 aus Protokolldatenstrom 1 zur Verfügung gestellt wird. Zum Zeitpunkt C, als die Protokolldatei 4 geschlossen und in Protokolldatenstrom 1 archiviert ist, wird ein Befehl **ROLLFORWARD DATABASE** erfolgreich ausgeführt. Ein Abschneiden und Archivieren von Dateien aus allen Protokolldatenströmen kann mit dem Befehl **ARCHIVE LOG** erzwungen werden oder durch memberübergreifendes Inaktivieren der Datenbank.

Im Falle des Befehls **ARCHIVE LOG** werden die aktuellen Protokolldateien in jedem Protokolldatenstrom unabhängig voneinander abgeschnitten. Es gibt keine Garantie dafür, dass dies bei allen Membern zu genau demselben Zeitpunkt geschieht. Daher ist es auch bei Verwendung des Befehls **ARCHIVE LOG** möglich, dass bei Ausführung des Befehls **ROLLFORWARD DATABASE** ein Fehler SQL1273N zurückgegeben wird.

Während es häufig vorkommt, dass bei der Protokollübertragung in einer DB2 pureScale-Umgebung Protokolle fehlen und dies erwartet wird, geht eine aktualisierende Recovery am Bereitschaftsstandort in den meisten Fällen weiter als der letzte Befehl **ROLLFORWARD DATABASE** (auch wenn SQL1273N zurückgegeben wird). Daher sollte der Fehler selbst häufig erwartet werden. Es ist jedoch möglich, dass am primären Standort Probleme beim Archivieren einer Datei für einen bestimmten Protokolldatenstrom bestehen, während Protokolle für die übrigen Protokolldatenströme erfolgreich archiviert werden. Dies könnte das Ergebnis eines temporären Problems beim Zugriff auf den Archivierungsspeicher für einen Protokolldatenstrom sein. Solche Probleme können dazu führen, dass Protokolldatenstromzusammenführung und -wiedergabe auf der Bereitschaftsdatenbank angehalten werden, wodurch sich die Anzahl der Transaktionen erhöht, die im Katastrophenfall möglicherweise verloren gehen. Stellen Sie sicher, dass das Bereitschaftssystem auf dem neuesten Stand ist, indem Sie einen Befehl **ROLLFORWARD DATABASE** mit dem Parameter **QUERY STATUS** nach jeder aktualisierenden Recoveryoperation, die SQL1273N zurückgibt, eingeben, und sich vergewissern, dass Verarbeitungsfortschritte erzielt werden. Wenn Sie feststellen, dass eine aktualisierende Recovery auf der Bereitschaftsdatenbank über einen längeren Zeitraum keine Verarbeitungsfortschritte erzielt, müssen Sie ermitteln, warum die als fehlend gemeldete Protokolldatei auf dem Bereitschaftssystem nicht verfügbar ist, und das Problem beheben. Der Befehl **ARCHIVE LOG** kann verwendet werden, um die Protokolldateien abzuschneiden, die aktuell auf jedem Member aktualisiert werden, damit sie für die Archivierung und die nachfolgende Wiedergabe auf dem Bereitschaftssystem infrage kommen.

Bei einem Katastrophenfall (beispielsweise Feuer, Erdbeben, mutwillige Zerstörung oder andere Katastrophen) kann Ihr Plan für die Recovery so aussehen, dass mit den verbleibenden Protokollen eine aktualisierende Recovery oder mit den verfügbaren Protokollen ein Restore und eine aktualisierende Recovery ausgeführt wird. Wie bereits zuvor erwähnt, wird durch die aktualisierende Recovery möglicherweise festgestellt, dass eine oder mehrere Protokolldateien fehlen, da Protokolldateien auf der primären Datenbank geschrieben wurden, zum Zeitpunkt des Katastrophenfalls jedoch noch nicht archiviert waren (SQL1273N). Es ist auch möglich, dass ein archiviertes Protokoll vom Dienstprogramm zur aktualisierenden Recovery aus unerwarteten Gründen nicht gefunden werden kann; auch dies führt möglicherweise dazu, dass das Dienstprogramm zur aktualisierenden Recovery die Fehlernachricht SQL1273N zurückgibt. Es ist wichtig, den Endpunkt einer aktualisierenden Recovery mit dem Befehl **ROLLFORWARD DATABASE** und dem Parameter **QUERY STATUS** zu überprüfen und zu entscheiden, ob die Fehlerbedingung bezüglich des fehlenden Protokolls erwartet ist. Wenn die Fehlerbedingung bezüglich des fehlenden Protokolls erwartet bzw. der Endpunkt akzeptabel ist, können Sie einen Befehl **ROLLFORWARD DATABASE** mit dem Parameter **STOP** eingeben, um den Prozess der aktualisierenden Recovery abzuschließen.

Einschränkungen

Backup- und Restoreoperationen zwischen einer Umgebung, in der DB2 pureScale Feature installiert ist, und einer Umgebung, in der DB2 pureScale Feature nicht installiert ist, werden nicht unterstützt.

Nach einer Topologieänderung, bei der ein Member hinzugefügt oder gelöscht wurde, können Sie aktualisierende Recoverys nicht über den Zeitpunkt hinaus ausführen, zu dem die Topologieänderung vorgenommen wurde. Wenn Sie ein Member hinzufügen oder löschen, wird die Datenbank in den Status "Backup anstehend" versetzt, und Sie müssen ein Datenbankgesamtbackup durchführen, bevor eine Verbindung zu dieser Datenbank hergestellt werden kann. Für die Wiederherstellung führen Sie ein Restore dieses Backup-Images und eine aktualisierende Recovery bis zum Ende der Protokolle durch. Wenn Sie ein Restore eines Backup-Images von einem Zeitpunkt vor der Topologieänderung durchführen müssen, können Sie eine aktualisierende Recovery nur bis zu dem Zeitpunkt durchführen, zu dem die Topologieänderung vorgenommen wurde. Geben Sie hierzu den Befehl **ROLLFORWARD DATABASE** mit dem Parameter **TO END OF LOGS** (gibt SQL1546N zurück) und anschließend den Befehl **ROLLFORWARD DATABASE** mit dem Parameter **STOP** ein. Mit dieser Operation werden keine Transaktionen wiederhergestellt, durch die die Datenbank nach der Topologieänderung geändert wurde.

In einer DB2 pureScale-Umgebung sind die Parameter **ON ALL DBPARTITIONNUMS** und **ON DBPARTITION (0)** des Befehls **BACKUP DATABASE** gültig. Wenn Sie eine andere Datenbankpartitionsnummer als 0 angeben, wird jedoch ein Fehler (SQL0270N) zurückgegeben, da keine weiteren Datenbankpartitionen vorhanden sind.

Die folgende Einschränkung gilt für dieses Release:

- Eine Datenbank, die sich außerhalb einer DB2 pureScale-Umgebung befindet, kann in eine DB2 pureScale-Umgebung migriert werden. Für die Migration einer solchen Datenbank in eine DB2 pureScale-Umgebung können Sie keine Restoreoperationen für Datenbanken verwenden.

Beispiele

- Führen Sie für eine Datenbank mit dem Namen SAMPLE, die vier Member aufweist, ein Backup von einem beliebigen Member aus durch:

  ```
  BACKUP DB SAMPLE
  ```

- Führen Sie für eine Datenbank mit dem Namen SAMPLE, die ein einziges Member aufweist, ein Restore durch:

  ```
  RESTORE DB SAMPLE
  ```

- Verwenden Sie den Befehl **RECOVER DATABASE**, um einen Restore und eine aktualisierende Recovery für eine Datenbank mit dem Namen SAMPLE von einem beliebigen Member aus durchzuführen:

  ```
  RECOVER DB SAMPLE TO END OF LOGS
  ```

  Wenn die Datenbank nicht vorhanden ist, verwenden Sie die Befehle **RESTORE DATABASE** und **ROLLFORWARD DATABASE** statt des Befehls **RECOVER DATABASE**, da für die erfolgreiche Beendigung des Befehls **RECOVER DATABASE** eine vorhandene Datenbank mit einem vollständigen Datenbankprotokoll erforderlich ist.

Überwachen von Backup-Operationen

Sie können den Befehl **LIST UTILITIES** verwenden, um den Fortschritt von Backup-Operationen für eine Datenbank zu überwachen.

Vorgehensweise

Setzen Sie den Befehl **LIST UTILITIES** ab und geben Sie dabei den Parameter **SHOW DETAIL** an:

```
list utilities show detail
```

Ergebnisse

Für Backup-Operationen wird eine erste geschätzte Anzahl zu verarbeitender Byte angegeben. Während der Backup-Operation wird die geschätzte Anzahl zu verarbeitender Byte aktualisiert. Die angezeigten Byte entsprechen nicht der Größe des Images und sollten nicht zur Schätzung der Größe des Backup-Images verwendet werden. Das tatsächliche Image kann wesentlich kleiner sein, abhängig davon, ob es sich um ein inkrementelles Backup oder ein komprimiertes Backup handelt.

Beispiel

Es folgt eine Beispielausgabe der Leistungsüberwachung einer Offline-Backup-Operation für eine Datenbank:

```
ID                                = 3
Typ                               = BACKUP
Datenbankname                     = SAMPLE
Partitionsnummer                  = 0
Beschreibung                      = offline db
Startzeit                         = 08/04/2011 12:16:23.248367
Status                            = Wird ausgeführt
Aufruftyp                         = Benutzer
Drosselung:
      Priorität                   = Ungedrosselt
Fortschrittsüberwachung:
   Geschätzte Fertigstellung (%) = 31
      Gesamte Arbeit              = 123147277 Byte
      Abgeschlossene Arbeit       = 37857269 bytes
      Startzeit                   = 08/04/2011 12:16:23.248377
```

Optimieren der Leistung des Backup-Dienstprogramms

Wenn Sie eine Backup-Operation ausführen, wählt der DB2-Datenbankmanager automatisch einen optimalen Wert für die Anzahl der Puffer, die Puffergröße und die Parallelitätseinstellungen aus. Die Werte basieren auf der Kapazität des für Dienstprogramme verfügbaren Zwischenspeichers, der Anzahl der verfügbaren Prozessoren und der Datenbankkonfiguration.

Daher sollten Sie je nach der Größe der auf Ihrem System verfügbaren Speicherkapazität in Betracht ziehen, mehr Speicher zuzuordnen, indem Sie den Wert des Konfigurationsparameters **util_heap_sz** erhöhen.

Die Zielsetzung dabei ist, die zum Ausführen einer Backup-Operation erforderliche Zeit zu minimieren. Wenn Sie für die folgenden Parameter des Befehls **BACKUP DATABASE** nicht explizit Werte angeben, wählt der DB2-Datenbankmanager einen Wert für die Parameter aus:

- **WITH** *anzahl_puffer* **BUFFERS**
- **PARALLELISM** *n*
- **BUFFER** *puffergröße*

Wenn die Anzahl der Puffer und die Puffergröße nicht angegeben werden, sodass der DB2-Datenbankmanager die Werte festlegt, sollte sich dies auf große Datenbanken nur minimal auswirken. Bei kleinen Datenbanken kann dies hingegen die Backup-Images prozentual beträchtlich vergrößern. Selbst wenn der letzte Datenpuffer, der auf Platte geschrieben wird, wenig Daten enthält, wird trotzdem der vollständige Puffer in das Image geschrieben. In einer kleinen Datenbank bedeutet dies, dass ein beträchtlicher Prozentsatz der Image-Größe möglicherweise leer gelassen wird.

Sie können auch eine der folgenden Aktionen ausführen, um die für eine Backup-Operation erforderliche Zeit zu reduzieren:

- Geben Sie ein Tabellenbereichsbackup an.

  Sie können einen Teil einer Datenbank sichern (und anschließend wiederherstellen), indem Sie die Option **TABLESPACE** des Befehls **BACKUP DATABASE** verwenden. Dies vereinfacht die Verwaltung von Tabellendaten, Indizes und Langfeld- bzw. LOB-Daten in separaten Tabellenbereichen.
- Erhöhen Sie den Wert des Parameters **PARALLELISM** des Befehls **BACKUP DATABASE**, bis er der Anzahl der zu sichernden Tabellenbereiche entspricht.

  Der Parameter **PARALLELISM** definiert die Anzahl der Prozesse bzw. Threads, die zum Lesen von Daten aus der Datenbank und zum Komprimieren von Daten während einer Operation für komprimiertes Backup gestartet werden. Jeder Prozess oder Thread wird einem bestimmten Tabellenbereich zugeordnet, d. h. es besteht kein erzielbarer Nutzen, wenn Sie für den Parameter **PARALLELISM** einen Wert angeben, der die Anzahl der Tabellenbereiche übersteigt, für die ein Backup durchgeführt wird. Nach dem Backup eines Tabellenbereichs wird ein anderer Bereich angefordert. Beachten Sie jedoch, dass für jeden Prozess bzw. Thread sowohl Hauptspeicher- als auch CPU-Systemaufwand anfallen.
- Erhöhen Sie die Backup-Puffergröße.

  Die ideale Puffergröße für ein Backup ist ein Vielfaches der Speicherbereichsgröße (EXTENTSIZE) des Tabellenbereichs plus eine Seite. Wenn Sie mehrere Tabellenbereiche mit verschiedenen Angaben für die Speicherbereichsgröße haben, geben Sie als Wert ein Vielfaches des Werts für die Speicherbereichsgrößen plus eine Seite an.
- Erhöhen Sie die Anzahl der Puffer.

  Verwenden Sie mindestens doppelt so viele Puffer wie Backup-Ziele (oder Sitzungen), um sicherzustellen, dass die Backup-Zieleinheiten nicht auf Daten warten müssen.
- Verwenden Sie mehrere Zieleinheiten.

Backup und Restore - Statistik

Jede erfolgreiche Backup- und Restoreoperation generiert einen einzelnen Datensatz in der Datei `db2diag.log`, der Informationen zur Durchführung dieser Operation liefert. Der Protokolldatensatz dient zur Information und wird mit der Stufe DIAGLEVEL 3 (der Standardeinstellung) und DIAGLEVEL 4 protokolliert.

Beispiel

Die Protokolldatensätze für Backup- und Restorestatistikdaten bestehen aus einer Zeile pro Engine-Dispatchable-Unit (EDU) für den Backup- und Restorepuffermanipulator (db2bm) und einer Zeile pro Engine-Dispatchable-Unit (EDU) für den Backup- und Restoredatenträgercontroller (db2med):

```
2012-07-30-15.41.30.012922-240 E15775E1464          LEVEL: Info
PID     : 15882                TID : 46913126656320  KTID : 16001
PROC    : db2sysc
INSTANCE: krodger              NODE : 000            DB   : SAMPLE
APPHDL  : 0-18                 APPID: *LOCAL.krodger.120730194119
AUTHID  : KRODGER              HOSTNAME: hotel74
EDUID   : 49                   EDUNAME: db2agent (SAMPLE)
FUNCTION: DB2 UDB, database utilities, sqluxLogDataStats, probe:377
MESSAGE : Performance statistics
DATA #1 : String, 951 bytes

Number of buffers = 4
```

```
Buffer size       = 16781312 (4097 4kB pages)

BM#    Total      I/O      MsgQ      WaitQ      Buffers   kBytes
---  --------  --------  --------  --------    --------  --------
000      6.30      0.02      0.00      6.18           4       640
001      5.88      4.48      0.00      1.33           9    139536
---  --------  --------  --------  --------    --------  --------
TOT     12.18      4.51      0.00      7.51          13    140176

MC#    Total      I/O      MsgQ      WaitQ      Buffers   kBytes
---  --------  --------  --------  --------    --------  --------
000      6.36      0.34      5.94      0.00           9    114748
001      6.29      0.18      5.60      0.10           6     81944
---  --------  --------  --------  --------    --------  --------
TOT     12.66      0.53     11.55      0.10          15    196692
```

Bedeutung der Spalten:

BM ID der db2bm-EDU

Total
Zeitraum, in dem jede EDU vorhanden war

I/O
Zeit für die Durchführung von Lese- oder Schreib-E/A-Operationen in den oder aus dem stabilen Speicher

MsgQ
Zeit für das Warten auf einen E/A-Puffer

WaitQ
Zeit für das Warten auf eine Statusmaschinensteuernachricht

Buffers
Anzahl der verarbeiteten E/A-Puffer

KBytes
Menge der verarbeiteten Daten

MC ID der db2med-EDU

Beispiel

Bei komprimierten Backups enthält der Protokolldatensatz zwei zusätzliche Spalten mit Leistungsinformationen zur Komprimierungsoperation:

```
2012-07-30-15.41.47.228766-240 E38419E1913           LEVEL: Info
PID     : 15882                TID : 46913126656320  KTID : 16081
PROC    : db2sysc
INSTANCE: krodger              NODE : 000            DB   : SAMPLE
APPHDL  : 0-29                 APPID: *LOCAL.krodger.120730194132
AUTHID  : KRODGER              HOSTNAME: hotel74
EDUID   : 80                   EDUNAME: db2agent (SAMPLE)
FUNCTION: DB2 UDB, database utilities, sqluxLogDataStats, probe:377
MESSAGE : Performance statistics
DATA #1 : String, 1399 bytes

Number of buffers = 4
Buffer size       = 16781312 (4097 4K pages)
                                                                                  Compr
BM#    Total      I/O      Compr     MsgQ      WaitQ      Buffers   kBytes     kBytes
---  --------  --------  --------- --------  --------    --------  --------  ---------
000     12.08      4.36      7.18      0.00      0.37           5    139536     144941
001     11.87      0.01      0.01      0.00     11.79           1       640        640
---  --------  --------  --------- --------  --------    --------  --------  ---------
TOT     23.96      4.38      7.19      0.00     12.17           6    140176     145581
```

```
MC#    Total      I/O              MsgQ      WaitQ      Buffers   kBytes
---  --------  --------           --------  --------    --------  --------
000     12.07      0.11             11.76      0.00           4     49168
001     12.10      0.07             11.84      0.15           4     32808
---  --------  --------           --------  --------    --------  --------
TOT     24.18      0.19             23.61      0.15           8     81976
```

Compr
Zeit für die Durchführung der Komprimierungsoperation

Compr Bytes
Menge nicht komprimierter Daten, die komprimiert wurden

Erforderliche Zugriffsrechte und Berechtigungen zur Verwendung von Backup

Sie benötigen die Berechtigung SYSADM, SYSCTRL oder SYSMAINT, um das Backup-Dienstprogramm verwenden zu können.

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf diese zuzugreifen. Berechtigungsstufen stellen eine Methode dar, um Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zusammenzufassen. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte.

Benutzer können nur auf die Objekte zugreifen, für die sie zugriffsberechtigt sind, d. h., für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Kompatibilität von Online-Backup und anderen Dienstprogrammen

Einige Dienstprogramme können zur gleichen Zeit wie ein Online-Backup ausgeführt werden, während dies bei anderen nicht möglich ist.

Die folgenden Dienstprogramme sind mit einem Online-Backup kompatibel:

- **EXPORT**
- **INSPECT**

Die folgenden SQL-Anweisungen und Dienstprogramme sind mit einem Online-Backup nur unter bestimmten Umständen kompatibel:

- CREATE INDEX

  Im SMS-Modus können Sie die Online-Indexerstellung und das Online-Backup aufgrund der ALTER TABLE-Sperre nicht gleichzeitig ausführen. Die Online-Indexerstellung aktiviert diese Sperre im Exklusivmodus, während das Online-Backup sie im SHARE-Modus aktiviert.

  Im DMS-Modus können eine Online-Indexerstellung und ein Online-Backup in den meisten Fällen gleichzeitig ausgeführt werden. Wenn der Tabellenbereich, in dem Sie den Index erstellen, eine große Anzahl von Tabellen enthält, besteht jedoch die Möglichkeit, dass die Online-Indexerstellung intern eine Online-Backup-Sperre aktiviert, die mit jedem gleichzeitig ausgeführten Online-Backup in Konflikt gerät.
- REORG INDEX mit der Option ONLINE

  Ebenso wie die Online-Indexerstellung kann die Online-Indexreorganisation im SMS-Modus wegen der ALTER TABLE-Sperre nicht gleichzeitig mit einem Online-Backup ausgeführt werden. Die Online-Indexreorganisation aktiviert diese Sperre im Exklusivmodus, während das Online-Backup sie im SHARE-Modus

aktiviert. Darüber hinaus führt eine Operation zur Online-Indexreorganisation vor der Umschaltphase ein Quiesce für die Tabelle aus und aktiviert eine Sperre des Modus Z, die ein Online-Backup verhindert. Allerdings sollte die ALTER TABLE-Sperre die gleichzeitige Ausführung eines Online-Backups verhindern, bevor die Tabellensperre des Modus Z angefordert wurde.

Im DMS-Modus können eine Online-Indexreorganisation und ein Online-Backup gleichzeitig ausgeführt werden.

Darüber hinaus führt eine Online-Indexreorganisation vor der Umschaltphase ein Quiesce für die Tabelle aus und aktiviert eine Sperre des Modus Z, die ein Online-Backup verhindert.

- **IMPORT**

  Das Dienstprogramm IMPORT ist mit einem Online-Backup kompatibel, sofern der Befehl **IMPORT** nicht mit dem Parameter **REPLACE** abgesetzt wird. In diesem Fall aktiviert das Dienstprogramm IMPORT eine Sperre des Modus Z für die Tabelle und verhindert die gleichzeitige Ausführung eines Online-Backups.

- TRUNCATE TABLE

  Die Anweisung TRUNCATE ist nicht kompatibel mit dem Online-Backup, da sie eine Sperre des Modus Z für die Tabelle aktiviert und die gleichzeitige Ausführung eines Online-Backup verhindert.

- **ALLOW READ ACCESS LOAD**

  Ladeoperationen mit **ALLOW READ ACCESS** sind nicht mit einem Online-Backup kompatibel, wenn der Befehl **LOAD** mit dem Parameter **COPY NO** abgesetzt wird. In diesem Modus ändern beide Dienstprogramme den Tabellenbereichsstatus, sodass eines der Dienstprogramme einen Fehler meldet.

  Ladeoperationen mit **ALLOW READ ACCESS** sind mit einem Online-Backup kompatibel, wenn der Befehl **LOAD** mit der Option **COPY YES** ausgeführt wird, obwohl Kompatibilitätsprobleme auftreten können. Im SMS-Modus können die Dienstprogramme gleichzeitig ausgeführt werden. Sie arbeiten jedoch mit inkompatiblen Tabellensperrmodi, sodass es eventuell zu Wartezeiten für Tabellensperren kommt. Im DMS-Modus arbeiten beide Dienstprogramme mit inkompatiblen Sperrmodi "Internal-B" (OLB), sodass es zu Wartezeiten für Sperren dieses Modus kommen kann. Wenn die Dienstprogramme gleichzeitig für denselben Tabellenbereich ausgeführt werden, wird das Dienstprogramm LOAD möglicherweise gezwungen, zu warten, bis das Backup-Dienstprogramm die Verarbeitung des Tabellenbereichs beendet hat, bevor das Dienstprogramm LOAD fortfahren kann.

- REORG TABLE mit der Option ONLINE

  Die Bereinigungsphase der Online-Tabellenreorganisation kann nicht starten, solange ein Online-Backup ausgeführt wird. Sie können die Tabellenreorganisation, falls erforderlich, anhalten, sodass das Online-Backup abgeschlossen werden kann, bevor die Online-Tabellenreorganisation wieder aufgenommen wird.

  Sie können ein Online-Backup eines DMS-Tabellenbereichs starten, während eine Tabelle im gleichen Tabellenbereich online reorganisiert wird. Während der Abschneidephase kann es im Zusammenhang mit der Reorganisationsoperation jedoch zu Sperrenwartezeiten kommen.

  Sie können ein Online-Backup eines SMS-Tabellenbereichs nicht starten, während eine Tabelle im gleichen Tabellenbereich online reorganisiert wird. Beide Operationen erfordern eine exklusive Sperre.

- DDL-Anweisungen, die eine Sperre im Modus Z erfordern (wie ALTER TABLE, DROP TABLE und DROP INDEX)

  Ein Online-Backup eines DMS-Tabellenbereichs ist mit DDL-Anweisungen kompatibel, die eine Sperre im Modus Z erfordern.

Ein Online-Backup eines SMS-Tabellenbereichs muss warten, bis die Sperre des Modus Z freigegeben wird.

- Speichergruppen-DDLs

  Wenn Sie die Speichergruppen der Datenbank durch Ausführen einer der folgenden Anweisungen ändern wollen, dann sollten Sie diese Operation unbedingt auf den Zeitplan für Online-Backups abstimmen:

  – CREATE STOGROUP
  – ALTER STOGROUP
  – DROP STOGROUP
  – RENAME STOGROUP
  – ALTER DATABASE

  Wenn gerade ein Online-Backup ausgeführt wird, dann wartet die Speichergruppen-DDL, bis sie die erforderliche Sperre erhält. Dieser Vorgang kann einige Zeit in Anspruch nehmen. In ähnlicher Weise wartet ein Online-Backup, bis für alle momentan ausgeführten Speichergruppen-DDLs eine Commit- oder Rollback-Operation abgeschlossen ist.

- **RUNSTATS** mit der Option `ALLOW WRITE` oder `ALLOW READ`

  Der Befehl **RUNSTATS** ist mit dem Online-Backup kompatibel, es sei denn, der Systemkatalogtabellenbereich ist ein SMS-Tabellenbereich. Wenn sich der Systemkatalog in einem SMS-Tabellenbereich befindet, halten der Befehl **RUNSTATS** und das Online-Backup nicht kompatible Sperren für die Tabelle, die den Wartestatus für Sperren zur Folge haben.

- ALTER TABLESPACE

  Operationen, die die Funktion zur automatischen Größenänderung aktivieren bzw. inaktivieren oder Container für die Funktion zur automatischen Größenänderung ändern, sind während des Online-Backups eines Tabellenbereichs nicht zulässig.

- ALTER TABLESPACE mit der Option REBALANCE

  Wenn ein Online-Backup und die Neuausgleichsfunktion gleichzeitig ausgeführt werden, hält das Online-Backup die Neuausgleichsfunktion an, ohne zu warten, bis sie abgeschlossen ist.

Die folgenden Dienstprogramme sind mit einem Online-Backup nicht kompatibel:

- **REORG TABLE**
- **RESTORE DATABASE**
- **ROLLFORWARD DATABASE**
- **LOAD** mit der Option **ALLOW NO ACCESS**
- **SET WRITE**
- **BACKUP DATABASE** mit der Option `ONLINE`

  Gültig für Online-Backups auf Datenbankebene und auf Tabellenbereichsebene (falls diese denselben Tabellenbereich bzw. dieselben Tabellenbereiche betreffen).

Backup - Beispiele

Dieser Abschnitt enthält einige Beispiele zu unterschiedlichen Sicherungsstrategien.

Backup auf TSM

Im folgenden Beispiel wird die Datenbank SAMPLE in zwei gleichzeitig ablaufenden TSM-Clientsitzungen auf einem TSM-Server gesichert. Das Backup-Dienstprogramm berechnet die optimale Anzahl von Puffern. Die optimale Puffergröße in 4-KB-Seiten wird automatisch auf Basis der verfügbaren Speichermenge und der Anzahl verfügbarer Zieleinheiten berechnet. Die Parallelitätseinstellung wird ebenfalls automatisch berechnet und basiert auf der Anzahl der verfügbaren Prozessoren und der Anzahl der zu sichernden Tabellenbereiche.

```
db2 backup database sample use tsm open 2 sessions with 4 buffers

db2 backup database payroll tablespace (syscatspace, userspace1) to
/dev/rmt0, /dev/rmt1 with 8 buffers without prompting
```

Inkrementelles Backup

Im Folgenden sehen Sie ein Beispiel für eine wöchentliche inkrementelle Backup-Strategie für eine wiederherstellbare Datenbank. Diese Strategie umfasst ein wöchentliches Gesamtbackup der Datenbank, ein tägliches nicht kumulatives Backup (Delta-Backup) sowie ein kumulatives Backup (inkrementell) in der Wochenmitte:

```
(So) db2 backup db kdr use tsm
(Mo) db2 backup db kdr online incremental delta use tsm
(Di) db2 backup db kdr online incremental delta use tsm
(Mi) db2 backup db kdr online incremental use tsm
(Do) db2 backup db kdr online incremental delta use tsm
(Fr) db2 backup db kdr online incremental delta use tsm
(Sa) db2 backup db kdr online incremental use tsm
```

Backup auf Band

Setzen Sie den folgenden Befehl ab, um in einer Windows-Umgebung eine Backup-Operation für eine Bandeinheit einzuleiten:

```
db2 backup database sample to \\.\tape0
```

Kapitel 12. Recovery - Übersicht

Das Recoverydienstprogramm führt auf Basis der in der Datei des Recoveryprotokolls vorhandenen Informationen die erforderlichen Operationen zum Wiederherstellen und aktualisierenden Wiederherstellen einer Datenbank bis zu einem bestimmten Zeitpunkt aus.

Wenn Sie dieses Dienstprogramm verwenden, geben Sie an, ob die Datenbank bis zu einem bestimmten Zeitpunkt oder bis zum Ende der Protokolldateien wiederhergestellt werden soll. Das Dienstprogramm wählt anschließend das am besten geeignete Backup-Image aus und führt die Recoveryoperationen aus.

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Recovery für Datenbanken durchführen. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Das Recoverydienstprogramm bietet keine Unterstützung für die folgenden Optionen des Befehls **RESTORE DATABASE**:

- **TABLESPACE** *tabellenbereichsname*. Restoreoperationen für Tabellenbereiche werden nicht unterstützt.
- **INCREMENTAL**. Operationen zum inkrementellen Restore werden nicht unterstützt.
- **OPEN** *anzahl_sitzungen* **SESSIONS**. Die Anzahl E/A-Sitzungen, die mit TSM oder einem Programm eines anderen Lieferanten verwendet werden sollen, kann nicht angegeben werden.
- **BUFFER** *puffergröße*. Die Größe des für die Restoreoperation verwendeten Puffers kann nicht angegeben werden.
- **DLREPORT** *dateiname*. Es kann kein Dateiname angegeben werden für Berichtsdateien, deren Verknüpfung aufgehoben wurde.
- **WITHOUT ROLLING FORWARD**. Es kann nicht angegeben werden, dass die Datenbank nicht in den Status *Aktualisierende Recovery anstehend* versetzt werden soll, nachdem sie erfolgreich wiederhergestellt wurde.
- **PARALLELISM** *n*. Es kann kein Grad der Parallelität für die Restoreoperation angegeben werden.
- **WITHOUT PROMPTING**. Es kann nicht angegeben werden, dass die Restoreoperation nicht überwacht ausgeführt werden soll.

Darüber hinaus lässt das Recoverydienstprogramm nicht zu, dass Sie eine der **REBUILD**-Optionen angeben. Das Recoverydienstprogramm verwendet jedoch automatisch die geeignete **REBUILD**-Option, wenn es auf der Basis der Informationen in der Datei des Recoveryprotokolls keine Datenbank-Backup-Images lokalisieren kann.

Für den Befehl **RECOVER DATABASE** können Sie nicht die Option `TABLESPACE` oder die Option `INCREMENTAL` aus dem Befehl **RESTORE DATABASE** verwenden.

Für den Befehl **RECOVER DATABASE** ist die Option `restore` automatisiert. Gleiches gilt für die Option `REBUILD` des Befehls **RESTORE**.

Recovery von Daten

Mit dem Befehl **RECOVER DATABASE** können Sie für eine Datenbank und alle Speichergruppen eine Recovery bis zu einem bestimmten Zeitpunkt durchführen, indem Sie Informationen verwenden, die in der Recoveryprotokolldatei enthalten sind.

Vorbereitende Schritte

Wenn Sie den Befehl **RECOVER DATABASE** nach einer unvollständigen Recoveryoperation ausführen, die in der Phase der aktualisierenden Recovery (Rollforward) beendet wurde, versucht das Recoverydienstprogramm, die vorherige Recoveryoperation fortzusetzen, ohne die Restorephase zu wiederholen. Wenn Sie das Recoverydienstprogramm zwingen wollen, die Restorephase zu wiederholen, geben Sie den Befehl **RECOVER DATABASE** mit der Option **RESTART** ein. Dies veranlasst das Recoverydienstprogramm, jede frühere Recoveryoperation, die nicht vollständig abgeschlossen wurde, zu ignorieren. Wenn Sie die Anwendungsprogrammierschnittstelle (API) verwenden, geben Sie die Aufruferaktion DB2RECOVER_RESTART für das Feld **iRecoverAction** an, um das Recoverydienstprogramm zu veranlassen, die Restorephase zu wiederholen.

Wenn der Befehl **RECOVER DATABASE** während der Restorephase unterbrochen wird, kann er nicht fortgesetzt werden. Sie müssen den Befehl **RECOVER DATABASE** erneut absetzen.

Es sollte noch keine Verbindung zu der wiederherzustellenden Datenbank bestehen: Das Dienstprogramm für die Datenbankrecovery stellt automatisch eine Verbindung zu der angegebenen Datenbank her. Diese Verbindung wird nach Abschluss der Recoveryoperation beendet.

Informationen zu diesem Vorgang

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken muss das Recoverydienstprogramm von der Katalogpartition der Datenbank aus aufgerufen werden.

Vorgehensweise

Verwenden Sie eine der beiden folgenden Möglichkeiten, um das Recoverydienstprogramm aufzurufen:

- Befehl **RECOVER DATABASE**
- Anwendungsprogrammierschnittstelle (API) db2Recover

Beispiel

Das folgende Beispiel zeigt, wie Sie den Befehl **RECOVER DATABASE** über den CLP verwenden:

```
db2 recover db sample
```

Datenrecovery mit 'db2adutl'

Sie können eine knotenübergreifende Recovery mit dem Befehl **db2adutl** sowie mithilfe der Datenbankkonfigurationsparameter **logarchopt1** und **vendoropt** aus-

führen. Diese Recovery wird in Beispielen in verschiedenen TSM-Umgebungen (TSM = Tivoli Storage Manager) ausgeführt.

In den folgenden Beispielen besitzt Computer 1 den Namen bar und wird unter dem Betriebssystem AIX betrieben. Der Benutzer dieser Maschine ist roecken. Die Datenbank auf der Maschine bar heißt zample. Computer 2 besitzt den Namen dps. Dieser Computer wird ebenfalls unter dem Betriebssystem AIX von dem Benutzer regress9 ausgeführt.

Beispiel 1: Automatische Kennwortverwaltung durch den TSM-Server (die Option PASSWORDACCESS ist auf GENERATE gesetzt)

Dieses Beispiel für knotenübergreifende Recovery zeigt die Vorgehensweise zum Einrichten von zwei Computern für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter mit der Option PASSWORDACCESS=GENERATE verwaltet werden.

Anmerkung: Nach dem Aktualisieren der Datenbankkonfiguration müssen Sie möglicherweise ein Offline-Backup der Datenbank erstellen.

1. Um die Datenbank für Protokollarchivierung für den Computer bar auf dem TSM-Server zu aktivieren, aktualisieren Sie den Datenbankkonfigurationsparameter **logarchmeth1** für die Datenbank zample mit dem folgenden Befehl:

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
   ```

2. Trennen Sie mit dem folgenden Befehl alle Benutzer und Anwendungen von der Datenbank:

   ```
   db2 force applications all
   ```

3. Stellen Sie mit dem folgenden Befehl sicher, dass keine Anwendungen mit der Datenbank verbunden sind:

   ```
   db2 list applications
   ```

   Sie müssten eine Nachricht empfangen, die besagt, dass keine Daten zurückgegeben wurden.

   Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen.

4. Erstellen Sie mit dem folgenden Befehl ein Backup der Datenbank auf dem TSM-Server:

   ```
   db2 backup db zample use tsm
   ```

   Informationen ähnlich den folgenden werden zurückgegeben:

   ```
   Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: 20090216151025
   ```

   Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen. Die Reihenfolge, in der Sie diesen Schritt in den Datenbankpartitionen ausführen, variiert, je nachdem, ob Sie ein Online-Backup oder ein Offline-Backup durchführen. Weitere Informationen hierzu finden Sie in „Backup von Daten” auf Seite 328.

5. Stellen Sie mit dem folgenden Befehl die Verbindung zur Datenbank zample her:

   ```
   db2 connect to zample
   ```

6. Generieren Sie mit dem folgenden Befehl neue Transaktionsprotokolle für die Datenbank durch Erstellen einer Tabelle und Laden von Daten in den TSM-Server:

```
bar:/home/roecken> db2 load from mr of del modified by noheader replace
into employee copy yes use tsm
```

In diesem Beispiel hat die Tabelle den Namen employee, und die Daten werden aus einer Datei im ASCII-Format mit Begrenzern geladen, die den Namen mr hat. Die Option **COPY** YES wird angegeben, um eine Kopie der geladenen Daten zu erstellen, und durch die Option **USE TSM** wird angegeben, dass die Kopie der Daten auf dem TSM-Server zu speichern ist.

Anmerkung: Sie können die Option **COPY** YES nur angeben, wenn die Datenbank für die aktualisierende Recovery aktiviert ist. Das heißt, der Datenbankkonfigurationsparameter **logarchmeth1** muss auf den Wert USEREXIT, LOGRETAIN, DISK oder TSM gesetzt sein.

Das Dienstprogramm gibt eine Reihe von Nachrichten zurück, um den Verarbeitungsfortschritt anzuzeigen:

```
SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
"/home/roecken/mr".

SQL3500W  Die Phase "LOAD" wird gestartet (Zeit: "02/16/2009
15:12:13.392633").

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "0".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3110N  Die Verarbeitung des Dienstprogramms ist abgeschlossen. Es wurde(n) "1" Zeile(n)
aus der Eingabedatei gelesen.

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Eingabesatzzähler = "1".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3515W  Die Phase "LOAD" wurde beendet (Zeit: "02/16/2009
15:12:13.445718").


   Anzahl gelesener Zeilen         = 1
   Anzahl übersprungener Zeilen    = 0
   Anzahl geladener Zeilen         = 1
   Anzahl zurückgewiesener Zeilen  = 0
   Anzahl gelöschter Zeilen        = 0
   Anzahl festgeschriebener Zeilen = 1
```

7. Vergewissern Sie sich nach dem Laden der Daten in die Tabelle, dass auf dem TSM-Server ein Backup-Image, ein Ladekopieimage und eine Protokolldatei vorhanden ist, indem Sie die folgende Abfrage für die Datenbank zample ausführen:

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
```

Die folgenden Informationen werden zurückgegeben:

```
Retrieving FULL DATABASE BACKUP information.
 1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
        Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
    No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
    No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
    No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
```

```
    No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

  Retrieving DELTA TABLESPACE BACKUP information.
    No DELTA TABLESPACE BACKUP images found for ZAMPLE

  Retrieving LOAD COPY information.
   1 Time: 20090216151213

  Retrieving LOG ARCHIVE information.
  Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
  Taken at: 2009-02-16-15.10.38
```

8. Um die knotenübergreifende Recovery zu aktivieren, müssen Sie den Zugriff auf die zugeordneten Objekte des Computers bar für einen anderen Computer und ein anderes Benutzerkonto erteilen. Erteilen Sie in diesem Beispiel den Zugriff für den Computer dps und den Benutzer regress9 mit dem folgenden Befehl:

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl grant user regress9
            on nodename dps for db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Successfully added permissions for regress9 to access ZAMPLE on node dps.
   ```

   Anmerkung: Sie können die Ergebnisse der GRANT-Operation für **db2adutl** überprüfen, indem Sie mit dem folgenden Befehl die aktuelle Zugriffsliste für den aktuellen Knoten abrufen:

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl queryaccess
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Node                Username            Database Name   Type
   ------------------------------------------------------------
   DPS                 regress9            ZAMPLE          A
   ------------------------------------------------------------
    Access Types:  B - backup images   L - logs   A - both
   ```

9. In diesem Beispiel ist der zweite Computer (dps) noch nicht für die knotenübergreifende Recovery der Datenbank zample konfiguriert. Vergewissern Sie sich mit dem folgenden Befehl, dass diesem Benutzer und diesem Computer keine Daten auf dem TSM-Server zugeordnet sind:

   ```
   dps:/home/regress9/sqllib/adsm> db2adutl query db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
     --- Database directory is empty ---
   Warning: There are no file spaces created by DB2 on the ADSM server
   Warning: No DB2 backup images found in ADSM for any alias.
   ```

10. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte für die Datenbank zample ab, die dem Benutzer roecken und dem Computer bar zugeordnet sind:

    ```
    dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename
              bar owner roecken
    ```

    Die folgenden Informationen werden zurückgegeben:

    ```
    --- Database directory is empty ---

  Query for database ZAMPLE

    Retrieving FULL DATABASE BACKUP information.
     1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
            Sessions: 1

    Retrieving INCREMENTAL DATABASE BACKUP information.
        No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

    Retrieving DELTA DATABASE BACKUP information.
        No DELTA DATABASE BACKUP images found for ZAMPLE
    ```

```
Retrieving TABLESPACE BACKUP information.
    No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
    No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
    No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
 1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
Taken at: 2009-02-16-15.10.38
```

Diese Informationen stimmen mit den zuvor generierten TSM-Informationen überein und bestätigen damit, dass dieses Image auf dem Computer dps wiederhergestellt werden kann.

11. Stellen Sie mit dem folgenden Befehl die Datenbank zample vom TSM-Server auf dem Computer dps wieder her:

```
dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
```

Die folgenden Informationen werden zurückgegeben:

```
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Anmerkung: Wenn die Datenbank zample auf dps bereits vorhanden wäre, würde der Parameter **OPTIONS** weggelassen, und der Datenbankkonfigurationsparameter **vendoropt** würde verwendet. Dieser Konfigurationsparameter überschreibt den Parameter **OPTIONS** für eine BACKUP- oder RESTORE-Operation.

12. Führen Sie eine aktualisierende Recovery durch, um die Transaktionen anzuwenden, die beim Erstellen einer neuen Tabelle und beim Laden neuer Daten in der Protokolldatei für die Datenbank zample erfasst wurden. In diesem Beispiel schlägt die aktualisierende Recovery (ROLLFORWARD) fehl, weil das Dienstprogramm für die aktualisierende Recovery die Protokolldateien nicht finden kann, da keine Benutzer- und Computerinformationen angegeben sind:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Der Befehl gibt den folgenden Fehler zurück:

```
SQL4970N  Die aktualisierende Recovery der Datenbank "ZAMPLE"
kann wegen fehlender Protokolldatei(en) auf Knoten "0" nicht den angegebenen
Endpunkt (Protokollende oder angegebener Zeitpunkt) erreichen.
```

Veranlassen Sie das Dienstprogramm für aktualisierende Recovery, die Protokolldateien für einen anderen Computer zu suchen, indem Sie den entsprechenden Wert für **logarchopt** angeben. Verwenden Sie in diesem Beispiel den folgenden Befehl, um den Datenbankkonfigurationsparameter **logarchopt1** festzulegen und nach Protokolldateien für den Benutzer roecken und den Computer bar zu suchen:

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
```

13. Ermöglichen Sie mit dem folgenden Befehl dem Dienstprogramm für aktualisierende Recovery die Verwendung der Backup- und Ladekopieimages, indem Sie den Datenbankkonfigurationsparameter **vendoropt** festlegen:

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
```

14. Mit dem folgenden Befehl können Sie die knotenübergreifende Datenrecovery beenden, indem die in der Protokolldatei der Datenbank zample erfassten Transaktionen angewendet werden:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Die folgenden Informationen werden zurückgegeben:

```
                                 Status der aktualisierenden Recovery

    Aliasname der Eingabedatenbank             = zample
Anzahl der Member, die Status zurückgaben   = 1


Mitgldsnr.     Aktualis.    Nächstfolg.  Verarbeitete              Zuletzt festgeschriebene
               Wiederherst. Protokoll    Protokolldateien          Transaktion
               Status       zum Lesen
-------------  -----------  -----------  ------------------------  ------------------------------
        0      nicht anstehend           S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC


  DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Computer dps unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf dem Computer bar unter dem Benutzer roecken wiederhergestellt.

Beispiel 2: Kennwortverwaltung durch den Benutzer (die Option PASSWORDACCESS ist auf PROMPT gesetzt)

Dieses Beispiel für knotenübergreifende Recovery zeigt die Vorgehensweise zum Einrichten von zwei Computern für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter von den Benutzern verwaltet werden. In diesen Umgebungen sind zusätzliche Informationen erforderlich. Dies sind insbesondere der TSM-Knotenname und das Kennwort für den Computer, auf dem die Objekte erstellt wurden.

1. Fügen Sie in der Datei dsm.sys die folgende Zeile hinzu, weil der Quellencomputer den Namen bar trägt:

   ```
   NODENAME bar
   ```

   Anmerkung: Auf Windows-Betriebssystemen heißt diese Datei dsm.opt. Nach dem Aktualisieren dieser Datei müssen Sie für Ihr System einen Warmstart durchführen, damit die Änderungen wirksam werden.

2. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte ab, die dem Benutzer roecken und dem Computer bar zugeordnet sind:

   ```
   dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename bar
      owner roecken password *******
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Query for database ZAMPLE

      Retrieving FULL DATABASE BACKUP information.
       1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
              Sessions: 1

      Retrieving INCREMENTAL DATABASE BACKUP information.
          No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

      Retrieving DELTA DATABASE BACKUP information.
          No DELTA DATABASE BACKUP images found for ZAMPLE

      Retrieving TABLESPACE BACKUP information.
          No TABLESPACE BACKUP images found for ZAMPLE

      Retrieving INCREMENTAL TABLESPACE BACKUP information.
          No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

      Retrieving DELTA TABLESPACE BACKUP information.
          No DELTA TABLESPACE BACKUP images found for ZAMPLE

      Retrieving LOAD COPY information.
       1 Time: 20090216151213
   ```

```
Retrieving LOG ARCHIVE information.
Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
Taken at: 2009-02-16-15.10.38
```

3. Führen Sie die folgenden Schritte aus, wenn die Datenbank zample auf dem Computer dps noch nicht vorhanden ist:
   a. Erstellen Sie mit dem folgenden Befehl eine leere Datenbank zample:
      ```
      dps:/home/regress9> db2 create db zample
      ```
   b. Aktualisieren Sie den Datenbankkonfigurationsparameter **tsm_nodename** mit dem folgenden Befehl:
      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
      ```
   c. Aktualisieren Sie den Datenbankkonfigurationsparameter **tsm_password** mit dem folgenden Befehl:
      ```
      dps:/home/regress9> db2 update db cfg for zample using
         tsm_password ********
      ```
4. Versuchen Sie mit dem folgenden Befehl, die Datenbank zample wiederherzustellen:
   ```
   dps:/home/regress9> db2 restore db zample use tsm options
      "'-fromnode=bar -fromowner=roecken'" without prompting
   ```
   Die Restoreoperation wird erfolgreich ausgeführt, jedoch wird eine Warnung ausgeben:
   ```
      SQL2540W  RESTORE wurde erfolgreich ausgeführt, es wurde jedoch eine
   Warnung "2523" beim Restore der Datenbank im Modus
   'No Interrupt' festgestellt.
   ```
5. Führen Sie mit dem folgenden Befehl eine aktualisierende Recovery durch:
   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```
   Weil in diesem Beispiel die Datenbankkonfigurationsdatei durch die Restoreoperation ersetzt wurde, kann das Dienstprogramm zur aktualisierenden Recovery nicht die richtigen Protokolldateien finden und die folgende Fehlernachricht wird zurückgegeben:
   ```
   SQL1268N  Die aktualisierende Recovery wurde nach dem
   Fehler "-2112880618" beendet, während die Protokolldatei "S0000000.LOG"
   für die Datenbank "ZAMPLE" auf dem Knoten "0" abgerufen wurde.
   ```
   Setzen Sie die folgenden TSM-Datenbankkonfigurationsparameter auf die richtigen Werte zurück:
   a. Legen Sie mit dem folgenden Befehl den Konfigurationsparameter **tsm_nodename** fest:
      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
      ```
   b. Legen Sie mit dem folgenden Befehl den Datenbankkonfigurationsparameter **tsm_password** fest:
      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
      ```
   c. Legen Sie den Datenbankkonfigurationsparameter **logarchopt1** so fest, dass vom Dienstprogramm für aktualisierende Recovery die richtigen Protokolldateien gefunden werden; verwenden Sie hierfür den folgenden Befehl:
      ```
      dps:/home/regress9> db2 update db cfg for zample using logarchopt1
         "'-fromnode=bar -fromowner=roecken'"
      ```
   d. Legen Sie mit dem folgenden Befehl den Datenbankkonfigurationsparameter **vendoropt** so fest, dass die Recoverydatei auch während der aktualisierenden Recovery verwendet werden kann:
      ```
      dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
         "'-fromnode=bar -fromowner=roecken'"
      ```
6. Mit dem folgenden Befehl können Sie die knotenübergreifende Recovery durch das Ausführen der aktualisierenden Recovery beenden:
   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

Die folgenden Informationen werden zurückgegeben:

```
                                Status der aktualisierenden Recovery

    Aliasname der Eingabedatenbank             = zample
Anzahl der Member, die Status zurückgaben   = 1


Mitgldsnr.     Aktualis.    Nächstfolg.  Verarbeitete              Zuletzt festgeschriebene
               Wiederherst. Protokoll    Protokolldateien          Transaktion
               Status       zum Lesen
-------------  -----------  -----------  ------------------------  ------------------------------
        0      nicht anstehend           S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC


  DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Computer dps unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf dem Computer bar unter dem Benutzer roecken wiederhergestellt.

Beispiel 3: Konfigurieren des TSM-Servers für die Verwendung von Client-Proxy-Knoten

Dieses Beispiel für die knotenübergreifende Recovery zeigt die Vorgehensweise zum Einrichten von zwei Computern als Proxy-Knoten für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter mit der Option PASSWORDACCESS=GENERATE verwaltet werden.

Anmerkung: Nach dem Aktualisieren der Datenbankkonfiguration müssen Sie möglicherweise ein Offline-Backup der Datenbank erstellen.

In diesem Beispiel werden die Computer bar und dps unter dem Proxy-Namen clusternode registriert. Die Computer sind bereits als Proxy-Knoten konfiguriert.

1. Registrieren Sie mit den folgenden Befehlen die Computer bar und dps als Proxy-Knoten auf dem TSM-Server:

   ```
   REGISTER NODE clusternode mypassword
   GRANT PROXYNODE TARGET=clusternode AGENT=bar,dps
   ```

2. Um die Datenbank für die Protokollarchivierung für den TSM-Server zu aktivieren, aktualisieren Sie den Datenbankkonfigurationsparameter **logarchmeth1** für die Datenbank zample mit dem folgenden Befehl:

   ```
   bar:/home/roecken> db2 update db cfg for zample using
      LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
   ```

3. Trennen Sie mit dem folgenden Befehl alle Benutzer und Anwendungen von der Datenbank:

   ```
   db2 force applications all
   ```

4. Stellen Sie mit dem folgenden Befehl sicher, dass keine Anwendungen mit der Datenbank verbunden sind:

   ```
   db2 list applications
   ```

   Sie müssten eine Nachricht empfangen, die besagt, dass keine Daten zurückgegeben wurden.

   Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen.

5. Erstellen Sie mit dem folgenden Befehl ein Backup der Datenbank auf dem TSM-Server:

```
db2 backup db zample use tsm options "'-asnodename=clusternode'"
```

Informationen ähnlich den folgenden werden zurückgegeben:

```
Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: 20090216151025
```

Anstatt die Option **-asnodename** im Befehl **BACKUP DATABASE** anzugeben, können Sie den Datenbankkonfigurationsparameter **vendoropt** aktualisieren.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen. Die Reihenfolge, in der Sie diesen Schritt an den Datenbankpartitionen ausführen, variiert, je nachdem, ob Sie ein Online-Backup oder ein Offline-Backup durchführen. Weitere Informationen hierzu finden Sie in „Backup von Daten" auf Seite 328.

6. Stellen Sie mit dem folgenden Befehl die Verbindung zur Datenbank zample her:

```
db2 connect to zample
```

7. Generieren Sie mit dem folgenden Befehl neue Transaktionsprotokolle für die Datenbank durch Erstellen einer Tabelle und Laden von Daten in den TSM-Server:

```
bar:/home/roecken> db2 load from mr of del modified by noheader replace       into employee copy yes use ts
```

In diesem Beispiel hat die Tabelle den Namen employee, und die Daten werden aus einer Datei im ASCII-Format mit Begrenzern geladen, die den Namen mr hat. Die Option **COPY** YES wird angegeben, um eine Kopie der geladenen Daten zu erstellen, und durch die Option **USE TSM** wird angegeben, dass die Kopie der Daten auf dem TSM-Server zu speichern ist.

Anmerkung: Sie können die Option **COPY** YES nur angeben, wenn die Datenbank für die aktualisierende Recovery aktiviert ist. Das heißt, der Datenbankkonfigurationsparameter **logarchmeth1** muss auf den Wert USEREXIT, LOGRETAIN, DISK oder TSM gesetzt sein.

Das Dienstprogramm gibt eine Reihe von Nachrichten zurück, um den Verarbeitungsfortschritt anzuzeigen:

```
SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
"/home/roecken/mr".

SQL3500W  Die Phase "LOAD" wird gestartet (Zeit: "02/16/2009
15:12:13.392633").

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "0".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3110N  Die Verarbeitung des Dienstprogramms ist abgeschlossen. Es wurde(n) "1" Zeile(n)
aus der Eingabedatei gelesen.

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Eingabesatzzähler = "1".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3515W  Die Phase "LOAD" wurde beendet (Zeit: "02/16/2009
15:12:13.445718").


   Anzahl gelesener Zeilen         = 1
   Anzahl übersprungener Zeilen    = 0
   Anzahl geladener Zeilen         = 1
   Anzahl zurückgewiesener Zeilen  = 0
   Anzahl gelöschter Zeilen        = 0
   Anzahl festgeschriebener Zeilen = 1
```

8. Vergewissern Sie sich nach dem Laden der Daten in die Tabelle, dass auf dem TSM-Server ein Backup-Image, ein Ladekopieimage und eine Protokolldatei vorhanden ist, indem Sie die folgende Abfrage für die Datenbank zample ausführen:

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
           Sessions: 1

   Retrieving INCREMENTAL DATABASE BACKUP information.
       No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

   Retrieving DELTA DATABASE BACKUP information.
       No DELTA DATABASE BACKUP images found for ZAMPLE

   Retrieving TABLESPACE BACKUP information.
       No TABLESPACE BACKUP images found for ZAMPLE

   Retrieving INCREMENTAL TABLESPACE BACKUP information.
       No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

   Retrieving DELTA TABLESPACE BACKUP information.
       No DELTA TABLESPACE BACKUP images found for ZAMPLE

   Retrieving LOAD COPY information.
    1 Time: 20090216151213

   Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
   ```

9. In diesem Beispiel ist der zweite Computer (dps) noch nicht für die knotenübergreifende Recovery der Datenbank zample konfiguriert. Vergewissern Sie sich mit dem folgenden Befehl, dass diesem Benutzer und diesem Computer keine Daten zugeordnet sind:

   ```
   dps:/home/regress9/sqllib/adsm> db2adutl query db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
     --- Database directory is empty ---
   Warning: There are no file spaces created by DB2 on the ADSM server
   Warning: No DB2 backup images found in ADSM for any alias.
   ```

10. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte für die Datenbank zample ab, die dem Proxy-Knoten clusternode zugeordnet sind:

    ```
    dps:/home/regress9/sqllib/adsm> db2adutl query db zample
       options="-asnodename=clusternode"
    ```

    Die folgenden Informationen werden zurückgegeben:

    ```
       --- Database directory is empty ---

    Query for database ZAMPLE

       Retrieving FULL DATABASE BACKUP information.
        1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
               Sessions: 1

       Retrieving INCREMENTAL DATABASE BACKUP information.
           No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

       Retrieving DELTA DATABASE BACKUP information.
           No DELTA DATABASE BACKUP images found for ZAMPLE

       Retrieving TABLESPACE BACKUP information.
           No TABLESPACE BACKUP images found for ZAMPLE
    ```

```
Retrieving INCREMENTAL TABLESPACE BACKUP information.
    No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
    No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
 1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
Taken at: 2009-02-16-15.10.38
```

Diese Informationen stimmen mit den zuvor generierten TSM-Informationen überein und bestätigen damit, dass dieses Image auf dem Computer dps wiederhergestellt werden kann.

11. Stellen Sie mit dem folgenden Befehl die Datenbank zample vom TSM-Server auf dem Computer dps wieder her:

    ```
    dps:/home/regress9> db2 restore db zample use tsm options
       "'-asnodename=clusternode'" without prompting
    ```

    Die folgenden Informationen werden zurückgegeben:

    ```
    DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
    ```

    Anmerkung: Wenn die Datenbank zample auf dps bereits vorhanden wäre, würde der Parameter **OPTIONS** weggelassen, und der Datenbankkonfigurationsparameter **vendoropt** würde verwendet. Dieser Konfigurationsparameter überschreibt den Parameter **OPTIONS** für eine BACKUP- oder RESTORE-Operation.

12. Führen Sie eine aktualisierende Recovery durch, um die Transaktionen anzuwenden, die beim Erstellen einer neuen Tabelle und beim Laden neuer Daten in der Protokolldatei für die Datenbank zample erfasst wurden. In diesem Beispiel schlägt die aktualisierende Recovery (ROLLFORWARD) fehl, weil das Dienstprogramm für die aktualisierende Recovery die Protokolldateien nicht finden kann, da keine Benutzer- und Computerinformationen angegeben sind:

    ```
    dps:/home/regress9> db2 rollforward db zample to end of logs and stop
    ```

    Der Befehl gibt den folgenden Fehler zurück:

    ```
    SQL4970N  Die aktualisierende Recovery der Datenbank "ZAMPLE"
    kann wegen fehlender Protokolldatei(en) auf Knoten "0" nicht den angegebenen
    Endpunkt (Protokollende oder angegebener Zeitpunkt) erreichen.
    ```

    Veranlassen Sie das Dienstprogramm für aktualisierende Recovery, die Protokolldateien auf einem anderen Computer zu suchen, indem Sie den entsprechenden Wert für **logarchopt** angeben. Verwenden Sie in diesem Beispiel den folgenden Befehl, um den Datenbankkonfigurationsparameter **logarchopt1** festzulegen und nach Protokolldateien für den Benutzer roecken und den Computer bar zu suchen:

    ```
    dps:/home/regress9> db2 update db cfg for zample using logarchopt1
       "'-asnodename=clusternode'"
    ```

13. Ermöglichen Sie mit dem folgenden Befehl dem Dienstprogramm für aktualisierende Recovery die Verwendung der Backup- und Ladekopieimages, indem Sie den Datenbankkonfigurationsparameter **vendoropt** festlegen:

    ```
    dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
       "'-asnodename=clusternode'"
    ```

14. Mit dem folgenden Befehl können Sie die knotenübergreifende Datenrecovery beenden, indem die in der Protokolldatei der Datenbank zample erfassten Transaktionen angewendet werden:

    ```
    dps:/home/regress9> db2 rollforward db zample to end of logs and stop
    ```

    Die folgenden Informationen werden zurückgegeben:

```
                                Status der aktualisierenden Recovery

    Aliasname der Eingabedatenbank             = zample
Anzahl der Member, die Status zurückgaben   = 1


Mitgldsnr.     Aktualis.    Nächstfolg.  Verarbeitete              Zuletzt festgeschriebene
               Wiederherst. Protokoll    Protokolldateien          Transaktion
               Status       zum Lesen
-------------  -----------  -----------  ------------------------  ------------------------------
        0      nicht anstehend           S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC


  DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Computer dps unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf dem Computer bar unter dem Benutzer roecken wiederhergestellt.

Beispiel 4: Konfigurieren des TSM-Servers für die Verwendung von Client-Proxy-Knoten in einer DB2 pureScale-Umgebung

Dieses Beispiel zeigt die Vorgehensweise zum Einrichten von zwei Membern als Proxy-Knoten für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter mit der Option PASSWORDACCESS=GENERATE verwaltet werden.

Anmerkung: Nach dem Aktualisieren der Datenbankkonfiguration müssen Sie möglicherweise ein Offline-Backup der Datenbank erstellen.

In diesem Beispiel werden die Member member1 und member2 unter dem Proxy-Namen clusternode registriert. In DB2 pureScale-Umgebungen können Sie Backup- oder Datenrecoveryoperationen von jedem Member aus ausführen. In diesem Beispiel wird die Datenrevocery von dem Member member2 aus ausgeführt.

1. Registrieren Sie mit den folgenden Befehlen die Member member1 und member2 als Proxy-Knoten auf dem TSM-Server:

   ```
   REGISTER NODE clusternode mypassword
   GRANT PROXYNODE TARGET=clusternode AGENT=member1,member2
   ```

2. Um die Datenbank für die Protokollarchivierung für den TSM-Server zu aktivieren, aktualisieren Sie den Datenbankkonfigurationsparameter **logarchmeth1** für die Datenbank zample mit dem folgenden Befehl:

   ```
   member1:/home/roecken> db2 update db cfg for zample using
      LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
   ```

   Anmerkung: In DB2 pureScale-Umgebungen können Sie die globalen **logarchmeth1**-Datenbankkonfigurationsparameter einmal (1) von einem beliebigen Member aus festlegen.

   Die folgenden Informationen werden zurückgegeben:

   ```
   DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
   ```

3. Trennen Sie mit dem folgenden Befehl alle Benutzer und Anwendungen von der Datenbank:

   ```
   db2 force applications all
   ```

4. Stellen Sie mit dem folgenden Befehl sicher, dass keine Anwendungen mit der Datenbank verbunden sind:

   ```
   db2 list applications global
   ```

   Sie müssten eine Nachricht empfangen, die besagt, dass keine Daten zurückgegeben wurden.

5. Erstellen Sie mit dem folgenden Befehl ein Backup der Datenbank auf dem TSM-Server:

   ```
   db2 backup db zample use tsm options '-asnodename=clusternode'
   ```

   Informationen ähnlich den folgenden werden zurückgegeben:

   ```
   Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: 20090216151025
   ```

   Anstatt die Option **-asnodename** im Befehl **BACKUP DATABASE** anzugeben, können Sie den Datenbankkonfigurationsparameter **vendoropt** aktualisieren.

   Anmerkung: In DB2 pureScale-Umgebungen können Sie diesen Befehl zum Ausführen eines Backups aller Daten für die Datenbank von einem beliebigen Member aus ausführen.

6. Stellen Sie mit dem folgenden Befehl die Verbindung zur Datenbank zample her:

   ```
   db2 connect to zample
   ```

7. Generieren Sie mit dem folgenden Befehl neue Transaktionsprotokolle für die Datenbank durch Erstellen einer Tabelle und Laden von Daten in den TSM-Server:

   ```
   member1:/home/roecken> db2 load from mr of del modified by noheader replace
         into employee copy yes use tsmwhere
   ```

   In diesem Beispiel hat die Tabelle den Namen employee, und die Daten werden aus einer Datei im ASCII-Format mit Begrenzern geladen, die den Namen mr hat. Die Option **COPY** YES wird angegeben, um eine Kopie der geladenen Daten zu erstellen, und durch die Option **USE TSM** wird angegeben, dass die Kopie der Daten auf dem TSM-Server zu speichern ist.

   Anmerkung: Sie können die Option **COPY** YES nur angeben, wenn die Datenbank für die aktualisierende Recovery aktiviert ist. Das heißt, der Datenbankkonfigurationsparameter **logarchmeth1** muss auf den Wert USEREXIT, LOGRETAIN, DISK oder TSM gesetzt sein.

   Das Dienstprogramm gibt eine Reihe von Nachrichten zurück, um den Verarbeitungsfortschritt anzuzeigen:

   ```
   SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
   "/home/roecken/mr".

   SQL3500W  Die Phase "LOAD" wird gestartet (Zeit: "02/16/2009
   15:12:13.392633").

   SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "0".

   SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

   SQL3110N  Die Verarbeitung des Dienstprogramms ist abgeschlossen. Es wurde(n) "1" Zeile(n)
   aus der Eingabedatei gelesen.

   SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Eingabesatzzähler = "1".

   SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

   SQL3515W  Die Phase "LOAD" wurde beendet (Zeit: "02/16/2009
   15:12:13.445718").


      Anzahl gelesener Zeilen         = 1
      Anzahl übersprungener Zeilen    = 0
      Anzahl geladener Zeilen         = 1
      Anzahl zurückgewiesener Zeilen  = 0
      Anzahl gelöschter Zeilen        = 0
      Anzahl festgeschriebener Zeilen = 1
   ```

8. Vergewissern Sie sich nach dem Laden der Daten in die Tabelle, dass auf dem TSM-Server ein Backup-Image, ein Ladekopieimage und eine Protokolldatei vorhanden ist, indem Sie die folgende Abfrage für die Datenbank `zample` ausführen:

```
member1:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
```

   Die folgenden Informationen werden zurückgegeben:

```
      Retrieving FULL DATABASE BACKUP information.
       1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
              Sessions: 1

      Retrieving INCREMENTAL DATABASE BACKUP information.
          No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

      Retrieving DELTA DATABASE BACKUP information.
          No DELTA DATABASE BACKUP images found for ZAMPLE

      Retrieving TABLESPACE BACKUP information.
          No TABLESPACE BACKUP images found for ZAMPLE

      Retrieving INCREMENTAL TABLESPACE BACKUP information.
          No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

      Retrieving DELTA TABLESPACE BACKUP information.
          No DELTA TABLESPACE BACKUP images found for ZAMPLE

      Retrieving LOAD COPY information.
       1 Time: 20090216151213

   Retrieving LOG ARCHIVE information.

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.01.10

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.01.11

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.01.19

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.02.49

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.03.15

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.03.15

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.03.16
```

9. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte für die Datenbank `zample` ab, die dem Proxy-Knoten `clusternode` zugeordnet sind:

```
member2:/home/regress9/sqllib/adsm> db2adutl query db zample
   options="-asnodename=clusternode"
```

   Die folgenden Informationen werden zurückgegeben:

```
      --- Database directory is empty ---

   Query for database ZAMPLE

      Retrieving FULL DATABASE BACKUP information.
       1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
              Sessions: 1

      Retrieving INCREMENTAL DATABASE BACKUP information.
          No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

      Retrieving DELTA DATABASE BACKUP information.
          No DELTA DATABASE BACKUP images found for ZAMPLE

      Retrieving TABLESPACE BACKUP information.
```

```
        No TABLESPACE BACKUP images found for ZAMPLE

     Retrieving INCREMENTAL TABLESPACE BACKUP information.
        No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

     Retrieving DELTA TABLESPACE BACKUP information.
        No DELTA TABLESPACE BACKUP images found for ZAMPLE

     Retrieving LOAD COPY information.
      1 Time: 20090216151213

  Retrieving LOG ARCHIVE information.

  Log file: S0000000.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.01.10

  Log file: S0000000.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.01.11

  Log file: S0000000.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.01.19

  Log file: S0000001.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.02.49

  Log file: S0000001.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.02.49

  Log file: S0000001.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.02.49

  Log file: S0000002.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.03.15

  Log file: S0000002.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.03.15

  Log file: S0000002.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.03.16
```

Diese Informationen stimmen mit den zuvor generierten TSM-Informationen überein und bestätigen damit, dass dieses Image auf dem Member member2 wiederhergestellt werden kann.

10. Stellen Sie die Datenbank zample auf dem TSM-Server vom Member member2 aus wieder her, indem Sie folgenden Befehl verwenden:

```
member2:/home/regress9> db2 restore db zample use tsm options
   '-asnodename=clusternode' without prompting
```

Die folgenden Informationen werden zurückgegeben:

```
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Anmerkung: Wenn die Datenbank zample auf member2 bereits vorhanden wäre, würde der Parameter **OPTIONS** weggelassen, und der Datenbankkonfigurationsparameter **vendoropt** würde verwendet. Dieser Konfigurationsparameter überschreibt den Parameter **OPTIONS** für eine BACKUP- oder RESTORE-Operation.

11. Ermöglichen Sie mit dem folgenden Befehl dem Dienstprogramm für aktualisierende Recovery die Verwendung der Backup- und Ladekopieimages, indem Sie den Datenbankkonfigurationsparameter **vendoropt** festlegen:

```
member2:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-asnodename=clusternode'"
```

Anmerkung: In DB2 pureScale-Umgebungen können Sie die globalen **vendoropt**-Datenbankkonfigurationsparameter einmal (1) von einem beliebigen Member aus festlegen.

12. Mit dem folgenden Befehl können Sie die memberübergreifende Datenrecovery beenden, indem die in der Protokolldatei der Datenbank zample erfassten Transaktionen angewendet werden:

```
member2:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Die folgenden Informationen werden zurückgegeben:

```
                                Status der aktualisierenden Recovery

 Aliasname der Eingabedatenbank             = zample
```

```
Anzahl der Member, die Status zurückgaben = 3


Mitgldsnr.     Aktualis.    Nächstfolg.  Verarbeitete              Zuletzt festgeschriebene
               Wiederherst. Protokoll    Protokolldateien          Transaktion
               Status       zum Lesen
-------------  -----------  -----------  ------------------------  ------------------------------
            0  nicht anstehend           S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
            1  nicht anstehend           S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
            2  nicht anstehend           S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC

  DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank `zample` auf dem Member `member2` unter dem Benutzer `regress9` wurde auf denselben Stand wie die Datenbank auf Member `member1` unter dem Benutzer `roecken` wiederhergestellt.

Recovery einer gelöschten Tabelle

Es ist möglich, dass Sie versehentlich eine Tabelle mit Daten löschen, die noch benötigen. In solchen Fällen empfiehlt es sich, nach dem Löschen einer Tabelle die kritischen Tabellen wiederherstellbar zu machen.

Die Tabellendaten können durch eine Datenbankrecovery mit anschließender aktualisierender Recovery bis zu einem bestimmten Zeitpunkt, der vor dem Löschzeitpunkt der Tabelle liegen muss, wiederhergestellt werden. Die Operationen zum Restore und zur aktualisierenden Recovery können zeitaufwendig sein, wenn die Datenbank sehr groß ist. Zudem sind die Daten während der Recovery nicht verfügbar.

Die Funktion zur Recovery gelöschter Tabellen ermöglicht es Ihnen, Daten in gelöschten Tabellen über einen Restore und eine anschließende aktualisierende Recovery auf Tabellenbereichsebene wiederherzustellen.

Diese Recovery auf Tabellenbereichsebene nimmt weniger Zeit in Anspruch als eine Recovery auf Datenbankebene, und die Datenbank steht den Benutzern während der Recovery weiterhin zur Verfügung.

Vorbereitende Schritte

Damit eine gelöschte Tabelle wiederherstellbar ist, muss für den Tabellenbereich, in dem sich die Tabelle befindet, der Parameter DROPPED TABLE RECOVERY aktiviert sein. Diese Option kann während der Erstellung des Tabellenbereichs oder durch Aufrufen der Anweisung ALTER TABLESPACE aktiviert werden. Die Option DROPPED TABLE RECOVERY ist tabellenbereichsspezifisch und kann nur für einen regulären Tabellenbereich angegeben werden. Wenn Sie feststellen möchten, ob ein Tabellenbereich wiederherstellbar ist, können Sie die Spalte DROP_RECOVERY in der Katalogtabelle SYSCAT.TABLESPACES abfragen.

Die Option DROPPED TABLE RECOVERY ist bei der Erstellung eines Tabellenbereichs automatisch aktiviert. Wenn Sie keinen Tabellenbereich für die Recovery einer gelöschten Tabelle aktivieren möchten, können Sie entweder die Option DROPPED TABLE RECOVERY explizit auf OFF setzen, wenn Sie die Anweisung CREATE TABLESPACE absetzen, oder Sie können die Anweisung ALTER TABLESPACE verwenden, um die Recovery für gelöschte Tabellen für einen vorhandenen Tabellenbereich zu inaktivieren. Wenn für zahlreiche gelöschte Tabellen eine Recovery durchgeführt werden muss oder wenn die Protokolldatei sehr groß ist, dann kann die Funktion zur Recovery gelöschter Tabellen zu Auswirkungen auf die Leistung bei der aktualisierenden Recovery führen.

Wenn eine Anweisung DROP TABLE für eine Tabelle ausgeführt wird, deren Tabellenbereich für eine Recovery gelöschter Tabellen aktiviert ist, wird ein zusätzlicher Eintrag in den Protokolldateien vorgenommen, der die gelöschte Tabelle angibt. Außerdem wird in der Datei des Recoveryprotokolls ein Eintrag mit Informationen erstellt, die für die erneute Erstellung der Tabelle verwendet werden können.

Für partitionierte Tabellen ist die Option DROPPED TABLE RECOVERY immer aktiviert, selbst wenn DROPPED TABLE RECOVERY für nicht partitionierte Tabellen in einem oder mehreren Tabellenbereichen inaktiviert ist. Für eine partitionierte Tabelle wird nur ein Protokollsatz über die gelöschte Tabelle geschrieben. Dieser Protokollsatz reicht aus, um alle Datenpartitionen der Tabelle wiederherzustellen.

Informationen zu diesem Vorgang

Wenn sich die Tabelle im Status 'Reorganisation anstehend' befand, als sie gelöscht wurde, stimmt die DDL-Anweisung CREATE TABLE in der Protokolldatei nicht genau mit der entsprechenden Anweisung in der Importdatei überein. Die Importdatei liegt in dem Format der Tabelle vor, das sie aufwies, bevor die erste Anweisung ALTER mit empfohlenem **REORG** ausgeführt wurde. Die Anweisung CREATE TABLE in der Protokolldatei stimmt jedoch mit dem Status der Tabelle einschließlich der Ergebnisse aller Anweisungen ALTER TABLE überein.

Mit LOAD oder IMPORT zu verwendende Änderungswerte für Dateitypen
Geben Sie für eine Recovery der Tabelle durch Laden oder Importieren die folgenden Änderungswerte für Dateitypen an:

- Der Dateitypänderungswert **usegraphiccodepage** muss im Befehl **IMPORT** oder **LOAD** verwendet werden, wenn die wiederherzustellenden Daten den Datentyp GRAPHIC oder VARGRAPHIC aufweisen. Der Grund hierfür liegt darin, dass mehrere Codepages enthalten sein könnten.
- Der Dateitypänderungswert **delprioritychar** muss im Befehl **IMPORT** oder **LOAD** verwendet werden. Er ermöglicht es den Befehlen **LOAD** und **IMPORT**, Zeilen syntaktisch zu analysieren, die Zeilenumbruchzeichen innerhalb von Zeichen- oder Grafikspaltendaten enthalten.

Einschränkungen

Es kann nur jeweils eine gelöschte Tabelle wiederhergestellt werden.

Für die Datentypen, die von einer gelöschten Tabelle wiederhergestellt werden können, gibt es einige Einschränkungen. Folgende Daten können nicht wiederhergestellt werden:

- Die Option DROPPED TABLE RECOVERY kann nicht für temporäre Tabellen verwendet werden.
- Die den Zeilentypen zugeordneten Metadaten. (Die Daten selbst werden wiederhergestellt, nicht jedoch die Metadaten.) Die Daten in der Hierarchietabelle der typisierten Tabelle werden wiederhergestellt. Diese Daten sind möglicherweise größeren Umfangs, als sie in der gelöschten typisierten Tabelle den Anschein machten.
- XML-Daten. Wenn Sie versuchen, eine gelöschte Tabelle mit XML-Daten wiederherzustellen, sind die entsprechenden Spaltendaten leer.

Vorgehensweise

Gehen Sie wie folgt vor, um eine gelöschte Tabelle wiederherzustellen:

1. Geben Sie die gelöschte Datei an, indem Sie den Befehl **LIST HISTORY DROPPED TABLE** aufrufen. Die Kennung der gelöschten Tabelle wird in der Spalte für die Backup-ID ausgegeben.
2. Führen Sie den Restore auf Datenbank- oder Tabellenbereichsebene mit einem Backup-Image aus, das vor dem Löschen der Tabelle erstellt wurde.
3. Erstellen Sie ein Exportverzeichnis, in das die Dateien geschrieben werden können, die die Tabellendaten enthalten. Auf das Verzeichnis muss entweder von allen Datenbankpartitionen aus zugegriffen werden können, oder es muss in allen Datenbankpartitionen vorhanden sein. Jede Datenbankpartition erstellt automatisch Unterverzeichnisse in diesem Exportverzeichnis. Diese Unterverzeichnisse sind nach dem Muster `NODE`*`nnnn`* benannt, wobei *nnnn* für die Datenbankpartition oder Knotennummer steht. Die Datendateien, die die gelöschten Tabellendaten enthalten, so wie sie zuvor in jeder Datenbankpartition vorhanden waren, werden in ein untergeordnetes Unterverzeichnis mit dem Namen 'data' exportiert. Beispiel:

   ```
   \export_directory\NODE0000\data.
   ```
4. Führen Sie eine aktualisierende Recovery bis zu einem Zeitpunkt nach dem Löschen der Tabelle aus. Verwenden Sie dabei für den Befehl **ROLLFORWARD DATABASE** die Option **RECOVER DROPPED TABLE**. Sie können alternativ eine aktualisierende Recovery bis zum Ende der Protokolldateien durchführen, sodass Aktualisierungen an anderen Tabellen im Tabellenbereich oder der Datenbank nicht verloren gehen.
5. Erstellen Sie die Tabelle mithilfe der Anweisung CREATE TABLE anhand der Recoveryprotokolldatei erneut.
6. Importieren Sie die Tabellendaten, die während der aktualisierenden Recovery exportiert wurden, in die Tabelle. Wenn sich die Tabelle im Status *Reorganisation anstehend* befand, als sie gelöscht wurde, muss der Inhalt der DDL-Anweisung CREATE TABLE eventuell geändert werden, sodass er dem Inhalt der Datendatei entspricht.

Recovery nach Systemabsturz

Transaktionen (bzw. UOWs), die für eine Datenbank ausgeführt werden, können auf unerwartete Weise unterbrochen werden. Wenn eine Störung auftritt, bevor alle Änderungen, die Bestandteil der UOW (Unit of Work - Arbeitseinheit) sind, beendet, festgeschrieben und auf Platte geschrieben wurden, verbleibt die Datenbank in einem inkonsistenten und unbrauchbaren Status.

Die *Recovery nach einem Systemabsturz* versetzt die Datenbank wieder in einen konsistenten und verwendbaren Status. Dies geschieht durch Rollback der unvollständigen Transaktionen und Beenden der festgeschriebenen Aktionen, die sich zum Zeitpunkt des Systemabsturzes noch im Hauptspeicher befanden (Abb. 20 auf Seite 374). Wenn sich eine Datenbank in einem konsistenten und verwendbaren Status befindet, hat sie einen Punkt erreicht, der als *Konsistenzzustand* bezeichnet wird.

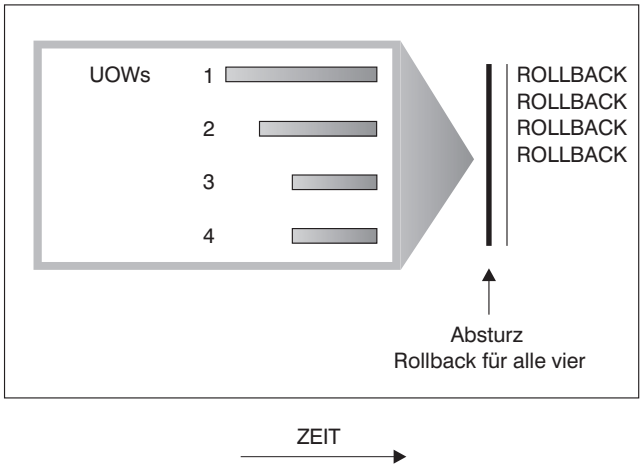


Abbildung 20. Rollback von UOWs (Recovery nach einem Systemabsturz)

Bei Verwendung von IBM DB2 pureScale Feature stehen zwei Typen von Recoverys nach Systemabsturz zur Verfügung, die Sie kennen sollten: *Recovery nach dem Absturz eines Members* und *Recovery nach dem Absturz einer Gruppe*. Die Recovery nach dem Absturz eines Members bezeichnet den Prozess für das Durchführen einer Recovery für einen Teil der Datenbank mithilfe eines einzigen Protokolldatenstroms des Members nach einem Fehlschlagen des Members. Die Recovery nach dem Absturz eines Members, die in der Regel automatisch als Teil des Neustarts des Members initialisiert wird, ist eine Onlineoperation. Dies bedeutet, dass andere Member weiterhin auf die Datenbank zugreifen können. Es kann für mehrere Member gleichzeitig eine Recovery nach Systemabsturz eines Members durchgeführt werden. Die Recovery nach dem Absturz einer Gruppe bezeichnet den Prozess für das Durchführen einer Recovery für eine Datenbank mithilfe von Protokolldatenströmen von mehreren Members nach einem Fehlschlagen, durch das im Cluster keine funktionsfähige Cluster-Caching-Funktion mehr vorhanden ist. Die Recovery nach dem Absturz einer Gruppe wird ebenfalls automatisch initialisiert (als Teil eines Gruppenneustarts). Wie bei DB2-Operationen für die Recovery nach dem Systemabsturz außerhalb einer DB2 pureScale-Umgebung ist während der Recovery nach dem Absturz einer Gruppe der Zugriff auf die Datenbank nicht möglich.

Wenn die Datenbank oder der Datenbankmanager fehlschlägt, verbleibt die Datenbank möglicherweise in einem inkonsistenten Status. Im Inhalt der Datenbank sind möglicherweise Änderungen vorhanden, die von Transaktionen vorgenommen wurden, die während des Fehlschlagens unvollständig waren. Möglicherweise fehlen der Datenbank auch Änderungen, die von Transaktionen vorgenommen wurden, die vor dem Fehlschlagen abgeschlossen, aber noch nicht auf Platte geschrieben wurden. Es muss eine Recoveryoperation nach einem Systemabsturz ausgeführt werden, um für die nur teilweise abgeschlossenen Transaktionen einen Rollback durchzuführen und um die Änderungen von abgeschlossenen Transaktionen, die bisher nur im Hauptspeicher vorgenommen wurden, auf Platte zu schreiben.

Zu den Bedingungen, die eine Recovery nach einem Systemabsturz erforderlich machen, gehören Folgende:

- Ein Stromausfall auf der Maschine, durch den der Datenbankmanager und die Datenbankpartitionen abnormal beendet werden.
- Ein Hardwarefehler, z. B. ein Hauptspeicher-, Platten, CPU- oder Netzfehler.

- Ein schwer wiegender Betriebssystemfehler, durch den die DB2-Instanz abnormal beendet wird.

Wenn Sie möchten, dass die Recovery nach einem Systemabsturz vom Datenbankmanager automatisch vorgenommen werden soll, aktivieren Sie den Datenbankkonfigurationsparameter für automatischen Neustart (**autorestart**), indem Sie ihn auf ON setzen. (Dies ist der Standardwert.) Wenn Sie den automatischen Neustart inaktivieren möchten, setzen Sie den Datenbankkonfigurationsparameter **autorestart** auf OFF. Ist der automatische Neustart inaktiviert, müssen Sie im Fall eines Datenbankfehlers den Befehl **RESTART DATABASE** absetzen. Wenn die Datenbank-E/A vor dem Auftreten des Absturzes ausgesetzt wurde, müssen Sie mit dem Befehl **RESTART DATABASE** die Option **WRITE RESUME** angeben, damit die Recovery nach dem Systemabsturz fortgesetzt wird. Der Neustart der Datenbank wird im Protokoll mit den Benachrichtigungen für die Systemverwaltung aufgezeichnet.

Wenn auf einer Datenbank, für die die aktualisierende Recovery aktiviert ist (d. h., der Konfigurationsparameter **logarchmeth1** ist auf einen anderen Wert als OFF gesetzt), eine Recovery nach einem Systemabsturz durchgeführt wird und während dieser Recovery ein Fehler auftritt, der auf einen einzelnen Tabellenbereich zurückzuführen ist, wird dieser Tabellenbereich in den Offlinestatus versetzt. Auf ihn kann erst wieder zugegriffen werden, nachdem er repariert wurde. Die Recovery nach dem Systemabsturz wird für andere Tabellenbereiche fortgesetzt. Nach der Beendigung der Recovery nach Systemabsturz kann auf die anderen Tabellenbereiche in der Datenbank zugegriffen werden, und es können Verbindungen zur Datenbank hergestellt werden. Wenn jedoch der in den Offlinestatus versetzte Tabellenbereich die Systemkataloge enthält, muss er repariert werden, bevor Verbindungen hergestellt werden können. Dieses Verhalten gilt nicht für DB2 pureScale-Umgebungen. Wenn während der Recovery nach dem Absturz eines Members oder einer Gruppe ein Fehler auftritt, schlägt die Operation für die Recovery nach dem Systemabsturz fehl.

Recovery beschädigter Tabellenbereiche

Ein beschädigter Tabellenbereich enthält einen oder mehrere Container, auf den/die nicht zugegriffen werden kann. Dies wird häufig durch Datenträgerprobleme verursacht, die entweder permanent (z. B. eine defekte Platte) oder temporär sind (z. B. eine Platte, die offline ist, oder ein abgehängtes Dateisystem).

Wenn der beschädigte Tabellenbereich der Bereich für die Systemkatalogtabellen ist, kann die Datenbank nicht erneut gestartet werden. Wenn die Containerprobleme nicht so behoben werden können, dass die ursprünglichen Daten erhalten bleiben, sind nur die folgenden Optionen verfügbar:

- Wiederherstellen der Datenbank
- Wiederherstellen des Katalogtabellenbereichs

Anmerkung:

1. Der Restore des Tabellenbereichs ist nur für wiederherstellbare Datenbanken zulässig, da die Datenbank aktualisierend wiederhergestellt werden muss.
2. Wenn Sie den Katalogtabellenbereich wiederherstellen, müssen Sie eine aktualisierende Recovery bis zum Ende der Protokolle durchführen.

Wenn es sich bei dem beschädigten Tabellenbereich *nicht* um den Tabellenbereich des Systemkatalogs handelt, versucht DB2 for Linux, UNIX and Windows, so viel von der Datenbank wie möglich verfügbar zu machen.

Wenn der beschädigte Tabellenbereich der einzige Tabellenbereich für temporäre Tabellen ist, müssen Sie einen neuen Tabellenbereich für temporäre Tabellen erstellen, sobald eine Verbindung zur Datenbank hergestellt werden kann. Nach der Erstellung kann der neue Tabellenbereich für temporäre Tabellen verwendet werden, und der normale Datenbankbetrieb, der einen Tabellenbereich für temporäre Tabellen erfordert, kann wieder aufgenommen werden. Den Offlinetabellenbereich für temporäre Tabellen können Sie löschen, wenn Sie es wünschen. Für die Reorganisation von Tabellen, die mit einem Tabellenbereich für temporäre Systemtabellen arbeitet, sind die folgenden speziellen Aspekte zu berücksichtigen:

- Wenn der Konfigurationsparameter **indexrec** der Datenbank oder des Datenbankmanagers auf RESTART gesetzt ist, müssen alle ungültigen Indizes während der Datenbankaktivierung erneut erstellt werden. Dies schließt Indizes aus einer Reorganisation ein, die während der Erstellungsphase abgestürzt ist.
- Wenn in einem beschädigten Tabellenbereich für temporäre Tabellen unvollständige Reorganisationsanforderungen vorhanden sind, müssen Sie möglicherweise den Konfigurationsparameter **indexrec** auf ACCESS setzen, um Fehler beim Neustart zu verhindern.

Recovery von Tabellenbereichen in wiederherstellbaren Datenbanken

Wenn eine Recovery nach Systemabsturz erforderlich ist, befindet sich der beschädigte Tabellenbereich offline und lässt keinen Zugriff mehr zu. Er wird in den Status 'Aktualisierende Recovery anstehend' versetzt. Wenn keine weiteren Probleme auftreten, kann ein Neustart die Datenbank auch mit dem beschädigten Tabellenbereich wieder online verfügbar machen.

Wieder online ist der beschädigte Tabellenbereich nicht verwendbar, jedoch kann der Rest der Datenbank genutzt werden. Zur Reparatur des beschädigten Tabellenbereichs und zur Wiederherstellung seiner Verwendbarkeit führen Sie die folgende Prozedur aus.

Vorgehensweise

Verwenden Sie eine der folgenden Prozeduren, um den beschädigten Tabellenbereich verwendbar zu machen:

- Methode 1
  1. Beheben Sie die Fehler an den beschädigten Containern ohne Verlust der ursprünglichen Daten.
  2. Führen Sie eine aktualisierende Recovery bis zum Ende der Protokolle für den Tabellenbereich aus.

     Anmerkung: Die Operation zur aktualisierenden Recovery versucht dabei zunächst, den Tabellenbereich wieder vom Offlinestatus in den normalen Status zurückzuversetzen.
- Methode 2
  1. Beheben Sie die Fehler an den beschädigten Containern mit oder ohne Verlust der ursprünglichen Daten.
  2. Führen Sie eine Restoreoperation für den Tabellenbereich aus.
  3. Führen Sie eine aktualisierende Recovery bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt für den Tabellenbereich aus.

Recovery von Tabellenbereichen in nicht wiederherstellbaren Datenbanken

Wenn eine Recovery nach Systemabsturz erforderlich ist, jedoch Tabellenbereiche beschädigt sind, könn Sie die Datenbank nur dann erfolgreich neu starten, wenn die beschädigten Tabellenbereiche aus der Datenbank gelöscht werden. In einer nicht wiederherstellbaren Datenbank werden die Protokolle, die zur Recovery der beschädigten Tabellenbereiche erforderlich sind, nicht behalten.

Daher besteht die einzig korrekte Aktion für solche Tabellenbereiche darin, sie zu löschen.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Datenbank mit beschädigten Tabellenbereichen erneut zu starten:

1. Rufen Sie eine Operation ohne Qualifikationsmerkmal zum erneuten Starten einer Datenbank auf. Wenn keine beschädigten Tabellenbereiche vorhanden sind, ist diese Operation erfolgreich. Wenn sie fehlschlägt (SQL0290N), finden Sie im Protokoll mit den Benachrichtigungen für die Systemverwaltung eine vollständige Liste der Tabellenbereiche, die momentan beschädigt sind.
2. Wenn Sie bereit sind, alle diese Tabellenbereiche zu löschen, leiten Sie eine weitere Neustartoperation für die Datenbank ein, wobei Sie alle beschädigten Tabellenbereiche unter der Option DROP PENDING TABLESPACES auflisten. Wenn ein beschädigter Tabellenbereich in der Liste DROP PENDING TABLESPACES aufgeführt ist, wird der Tabellenbereich in den Status 'Löschen anstehend' versetzt, und Sie müssen den Tabellenbereich nach Abschluss der Recoveryoperation löschen.

   Die Neustartoperation wird ohne Recovery der angegebenen Tabellenbereiche fortgesetzt. Wenn ein beschädigter Tabellenbereich *nicht* in der Liste DROP PENDING TABLESPACES aufgeführt ist, schlägt die Operation RESTART DATABASE mit der SQL-Nachricht SQL0290N fehl.

   Anmerkung: Wenn der Name eines Tabellenbereichs in der Liste DROP PENDING TABLESPACES aufgeführt wird, bedeutet dies nicht, dass der Tabellenbereich in den Status 'Löschen anstehend' versetzt wird. Ein Tabellenbereich wird nur dann in diesen Status versetzt, wenn während des Neustarts festgestellt wird, dass er beschädigt ist.
3. Ist die Operation für den Neustart der Datenbank erfolgreich, rufen Sie den Befehl **LIST TABLESPACES** auf, um zu ermitteln, welche Tabellenbereiche sich im Status 'Löschen anstehend' befinden.
4. Setzen Sie DROP TABLESPACE-Anweisungen ab, um die einzelnen Tabellenbereiche zu löschen, die sich im Status 'Löschen anstehend' befinden. Danach können Sie den Speicherbereich freigeben, den die beschädigten Tabellenbereiche verwendet haben, oder Sie können die Tabellenbereiche erneut erstellen.
5. Wenn Sie diese Tabellenbereiche nicht löschen (d. h. die darin enthaltenen Daten nicht verlieren) wollen, haben Sie folgende Möglichkeiten:
   - Beheben Sie die Fehler an den beschädigten Containern (ohne Verlust der ursprünglichen Daten).
   - Setzen Sie den Befehl **RESTART DATABASE** erneut ab.
   - Führen Sie eine Operation RESTORE DATABASE aus.

Begrenzen der Auswirkungen von Datenträgerausfällen

Ein *Datenträgerausfall* wird durch Fehler verursacht, die bei den Lese- und Schreibschnittstellen von Speichereinheiten wie z. B. Festplattenlaufwerken auftreten. Dieser Fehlertyp ist möglicherweise nur schwer zu erkennen und zu verhindern, daher müssen Maßnahmen ergriffen werden, die die potenziellen Auswirkungen des Fehlers verringern.

Zur Verringerung der Wahrscheinlichkeit eines Datenträgerfehlers und zur Vereinfachung der Recovery der Daten nach einem solchen Fehler sollten Sie folgende Maßnahmen ergreifen:

- Spiegeln oder duplizieren Sie die Platten, die die Daten und Protokolle für wichtige Datenbanken enthalten.
- Verwenden Sie eine RAID-Konfiguration (RAID - Redundant Array of Independent Disks), wie beispielsweise RAID-Stufe 5.
- Sehen Sie in einer Umgebung mit partitionierten Datenbanken für die Behandlung der Daten und der Protokolle in der Katalogpartition ein genau definiertes Verfahren vor. Da diese Datenbankpartition für die Verwaltung der Datenbank von entscheidender Bedeutung ist, sollten Sie Folgendes beachten:
  - Stellen Sie sicher, dass er sich auf einer zuverlässigen Platte befindet.
  - Duplizieren Sie ihn.
  - Erstellen Sie häufig Backups.
  - Speichern Sie keine Benutzerdaten auf ihm.

Schützen vor Plattenfehlern

Wenn Sie sich Gedanken über die Möglichkeit einer Beschädigung von Daten oder aktiven Protokollen aufgrund eines Plattenfehlers machen, sollten Sie den Einsatz von Systemen erwägen, die eine gewisse Toleranz gegenüber Plattenfehlern gewährleisten. Im Allgemeinen bietet sich hier die Verwendung einer *Platteneinheit* (Disk Array) an.

Platteneinheiten werden manchmal einfach als RAID (Redundant Array of Independent Disks) bezeichnet. Platteneinheiten können darüber hinaus durch Software auf Betriebssystem- oder Anwendungsebene implementiert werden. Das Unterscheidungsmerkmal zwischen Hardware- und Softwareplatteneinheiten ist die Art der CPU-Verarbeitung von E/A-Anforderungen. Bei Hardwareplatteneinheiten werden die E/A-Aktivitäten von Plattencontrollern verwaltet, während dies bei Softwareplatteneinheiten vom Betriebssystem bzw. von einer Anwendung übernommen wird.

Hardwareplatteneinheiten

Bei einer Hardwareplatteneinheit werden mehrere Platten von einem Plattencontroller mit eigener CPU verwendet und verwaltet. Da die gesamte Logik, die zur Verwaltung der zu dieser Einheit gehörenden Platten erforderlich ist, im Plattencontroller enthalten ist, ist diese Implementierung vom Betriebssystem unabhängig.

Es gibt verschiedene Typen der RAID-Architektur, die sich in Funktion und Leistung unterscheiden. Die derzeit gängigsten Typen sind jedoch RAID-Stufe 1 und Stufe 5.

RAID-Stufe 1 ist auch als Spiegelung (Mirroring) oder Duplizierung (Duplexing) von Platten bekannt. Bei der *Plattenspiegelung* werden Daten (eine vollständige Datei) von einer Platte auf eine zweite Platte kopiert, wobei nur ein einziger Platten-

controller verwendet wird. Die Vorgänge bei der *Plattenduplizierung* ähneln denen der Plattenspiegelung, jedoch sind hierbei die Platten an einen zweiten Plattencontroller angeschlossen (wie zwei SCSI-Adapter). Diese Verfahren bieten einen guten Datenschutz: es kann eine der beiden Platten ausfallen, und die Daten bleiben über die jeweils andere Platte verfügbar. Bei der Plattenduplizierung ist auch bei Ausfall eines Plattencontrollers der vollständige Schutz der Daten gewährleistet. Die Leistung ist gut, es wird jedoch die doppelte Anzahl an Platten benötigt.

RAID-Stufe 5 implementiert das plattenübergreifende Lesen und Schreiben von Daten (Striping) und die plattenübergreifende Parität nach Sektoren. Die Paritätsinformationen werden mit Daten verzahnt und nicht auf einem dedizierten Laufwerk gespeichert. Der Datenschutz ist gewährleistet: Wenn eine Platte ausfällt, stehen die Daten über die Informationen von den anderen Platten zusammen mit den verteilten Paritätsinformationen immer noch zur Verfügung. Die Leseleistung ist gut, die Schreibleistung jedoch nicht. Für eine RAID-5-Konfiguration sind mindestens drei identische Platten erforderlich. Die Menge des zusätzlich erforderlichen Plattenspeicherplatzes für den Systemaufwand variiert mit der Anzahl der Platten in der Platteneinheit. Im Fall einer RAID-5-Konfiguration mit fünf Platten beträgt der Speichermehraufwand 20 %.

Die RAID-Stufe 1+0 (10) umfasst das Spiegeln und Striping von Daten auf mindestens zwei Platten. Beim Spiegeln werden die Daten auf mindestens zwei Platten gleichzeitig geschrieben. Hierdurch erhalten Sie dieselbe Fehlertoleranz wie bei RAID-Stufe 1. Beim Striping werden die Daten in Blöcke unterteilt, und jeder Block wird in ein separates Plattenlaufwerk geschrieben. Hiermit wird eine hohe E/A-Leistung erzielt, indem das E/A-Aufkommen auf viele Kanäle und Laufwerke verteilt wird; die RAID-Stufe 1+0 reduziert jedoch den effektiven Plattenspeicherplatz auf die Hälfte, da alle Daten gespiegelt werden. Für die Implementierung der RAID-Stufe 10 sind mindestens 4 Laufwerke erforderlich.

Die RAID-Stufe 0+1 wird als gespiegeltes Array implementiert, dessen Segmente RAID-0-Arrays sind, und weist dieselbe Fehlertoleranz auf wie die RAID-Stufe 5. Dies liefert eine hohe E/A-Geschwindigkeit, indem das E/A-Aufkommen auf viele Kanäle und Laufwerke verteilt wird. Die RAID-Stufe 0+1 darf jedoch nicht mit der RAID-Stufe 1+0 verwechselt werden. Der Ausfall eines einzelnen Laufwerks führt dazu, dass das gesamte Array praktisch ein Array der RAID-Stufe 0 wird.

Bei Verwendung einer RAID-Platteneinheit (jedoch nicht RAID-Stufe 0) können Sie trotz einer ausgefallenen Platte auf die Daten der Platteneinheit zugreifen. Wenn Hot Plug- oder Hot Swap-fähige Platten in der Platteneinheit verwendet werden, kann die ausgefallene Platte während des Betriebs der Platteneinheit gegen eine Ersatzplatte ausgetauscht werden. Wenn bei einer RAID-5-Konfiguration zwei Platten gleichzeitig ausfallen, gehen alle Daten verloren (jedoch ist die Wahrscheinlichkeit eines gleichzeitigen Ausfalls zweier Platten sehr gering).

Für Ihre Protokolle können Sie eine Hardwareplatteneinheit der RAID-Stufe 1 oder eine Softwareplatteneinheit in Betracht ziehen, da diese Möglichkeiten eine Wiederherstellbarkeit bis zu dem Punkt des Ausfalls bieten und eine gute Schreibleistung zeigen, was für Protokolle wichtig ist. Verwenden Sie den Konfigurationsparameter *mirrorlogpath*, um auf einem Dateisystem der RAID-Stufe 1 einen Spiegelprotokollpfad anzugeben. In Fällen, in denen Zuverlässigkeit von essenzieller Bedeutung ist (d. h., dass keine Zeit für eine Recovery der Daten nach einem Plattenfehler verloren gehen darf) und die Schreibleistung nicht so wichtig ist, kann eine RAID-5-Konfiguration für die Hardwareplatteneinheit in Betracht kommen. Wenn hingegen die Schreibleistung eine erhebliche Rolle spielt und Sie diese trotz des Aufwands

für zusätzlichen Plattenspeicherplatz sicherstellen wollen, ziehen Sie eine RAID-1-Hardwareplatteneinheit sowohl für Ihre Daten als auch Ihre Protokolle in Betracht.

Weitere Informationen zu den verfügbaren RAID-Stufen finden Sie unter folgender Adresse (nur englisch): http://www.acnc.com/04_01_00.html.

Softwareplatteneinheiten

Eine Softwareplatteneinheit leistet im wesentlichen dasselbe wie eine Hardwareplatteneinheit, jedoch wird der Plattenverkehr entweder vom Betriebssystem oder von einem auf dem Server aktiven Anwendungsprogramm verwaltet. Wie alle anderen Programme auch steht die Softwareplatteneinheit bei der Nutzung der CPU- und Systemressourcen in einer Konkurrenzsituation. Dies ist keine gute Lösung für ein System mit knappen CPU-Ressourcen, und es ist zu bedenken, dass die Gesamtleistung der Platteneinheit von der Auslastung und Kapazität der CPU des Servers abhängig ist.

Eine typische Softwareplatteneinheit bietet Funktionen zur Spiegelung von Platten. Obwohl redundante Platten erforderlich sind, ist eine Softwareplatteneinheit relativ preiswert zu implementieren, da kostenintensive Plattencontroller nicht benötigt werden.

Vorsicht:
Wenn sich das Bootlaufwerk des Betriebssystems in der Platteneinheit befindet, kann Ihr System nicht starten, wenn dieses Laufwerk ausfällt. Wenn das Laufwerk ausfällt, bevor die Platteneinheit aktiv ist, kann die Platteneinheit keinen Zugriff auf das Laufwerk ermöglichen. Ein Bootlaufwerk sollte von der Platteneinheit getrennt betrieben werden.

Begrenzen der Auswirkungen von Transaktionsfehlern

Als *Transaktionsfehler* bezeichnet man den vorzeitigen Abbruch der Transaktionsverarbeitung, bevor die Transaktionen in der Datenbank festgeschrieben werden können, wodurch es zum Verlust oder zur Beschädigung von Daten kommen kann.

Zur Verringerung der Auswirkungen von Transaktionsfehlern versuchen Sie, Folgendes sicherzustellen:

- Eine ununterbrochene Stromversorgung für jeden DB2-Server
- Ausreichenden Plattenspeicherplatz für Datenbankprotokolle auf allen Datenbankpartitionen
- Zuverlässige Kommunikationsverbindungen zwischen den Datenbankpartitionsservern in einer Umgebung mit partitionierten Datenbanken
- Synchronisation der Systemuhren in einer Umgebung mit partitionierten Datenbanken

Recovery nach Transaktionsfehlern in einer Umgebung mit partitionierten Datenbanken

Tritt ein Transaktionsfehler in einer Umgebung mit partitionierten Datenbanken auf, ist in der Regel eine Datenbankrecovery sowohl auf dem ausgefallenen Datenbankpartitionsserver als auch auf allen anderen, an der Transaktion beteiligten Datenbankpartitionsservern erforderlich.

Es gibt zwei Arten der Datenbankrecovery:

- Eine Recovery nach Systemabsturz wird auf dem ausgefallenen Datenbankpartitionsserver durchgeführt, nachdem die Fehlerbedingung korrigiert wurde.
- Die *Recovery der Datenbankpartitionen nach einem Fehler* erfolgt auf den anderen (weiterhin aktiven) Datenbankpartitionsservern unmittelbar nach Feststellung des Fehlers.

In einer Umgebung mit partitionierten Datenbanken ist der Datenbankpartitionsserver, auf dem die Transaktion übergeben wird, die Koordinatorpartition und der erste Agent, der die Transaktion verarbeitet, ist der Koordinatoragent. Der Koordinatoragent ist für die Verteilung der Arbeit auf andere Datenbankpartitionsserver verantwortlich und protokolliert, welche Datenbankpartitionsserver an der Transaktion beteiligt sind. Wenn die Anwendung eine COMMIT-Anweisung für eine Transaktion ausführt, schreibt der Koordinatoragent die Transaktion mithilfe des Protokolls zum zweiphasigen Commit fest. Während der ersten Phase sendet die Koordinatorpartition eine PREPARE-Anforderung an alle anderen an der Transaktion beteiligten Datenbankpartitionsserver. Die Server antworten daraufhin wie folgt:

READ-ONLY
: Auf diesem Server gab es keine Datenänderung.

YES
: Auf diesem Server gab es eine Datenänderung.

NO
: Aufgrund eines Fehlers wurde der Server nicht für das Commit vorbereitet.

Wenn einer der Server mit `NO` antwortet, wird die Transaktion rückgängig gemacht. Andernfalls beginnt die Koordinatorpartition mit der zweiten Phase.

Während der zweiten Phase schreibt die Koordinatorpartition einen Commitprotokollsatz und sendet anschließend eine Commitanforderung an alle Server, die mit `YES` geantwortet haben. Wenn alle anderen Datenbankpartitionsserver das Commit durchgeführt haben, senden sie eine Bestätigung des Commits an die Koordinatorpartition. Die Transaktion ist abgeschlossen, wenn der Koordinatoragent von allen beteiligten Servern die Commitbestätigungen empfangen hat. Wenn dies der Fall ist, schreibt der Koordinatoragent einen FORGET-Protokollsatz.

Recovery auf einem aktiven Datenbankpartitionsserver nach Transaktionsfehler

Wenn ein Datenbankpartitionsserver feststellt, dass ein anderer Server inaktiv ist, werden alle Arbeiten, an denen der ausgefallene Datenbankpartitionsserver beteiligt ist, gestoppt:

- Wenn der noch aktive Datenbankpartitionsserver die Koordinatorpartition für eine Anwendung ist und die Anwendung auf dem ausgefallenen Datenbankpartitionsserver ausgeführt wird (und nicht zum Commit bereit ist), wird der Koordinatoragent unterbrochen, um Recoverymaßnahmen durchzuführen. Wenn der Koordinatoragent in der zweiten Phase der Commitverarbeitung ist, empfängt die Anwendung die Nachricht SQL0279N und verliert die Datenbankverbindung. Ansonsten sendet der Koordinatoragent eine Rollback-Anforderung an alle an der Transaktion beteiligten Server und die Anwendung empfängt die Nachricht SQL1229N.
- Wenn der ausgefallene Datenbankpartitionsserver die Koordinatorpartition für die Anwendung war, werden Agenten, die noch für die Anwendung auf den aktiven Servern arbeiten, unterbrochen, um Recoverymaßnahmen durchzuführen. Die Transaktion wird lokal auf jeder Datenbankpartition rückgängig gemacht, auf der sich die Transaktion nicht im Status vorbereiteter Transaktionen befindet. Auf den Datenbankpartitionen, auf denen sich die Transaktion im Sta-

tus vorbereiteter Transaktionen befindet, wird die Transaktion zu einer unbestätigten Transaktion. Die Koordinatordatenbankpartition ist nicht darüber informiert, dass die Transaktion auf einigen Datenbankpartitionen unbestätigt ist, weil sie nicht verfügbar ist.

- Wenn die Anwendung die Verbindung zu dem ausgefallenen Datenbankpartitionsserver (bevor er ausfiel) herstellte, aber weder der lokale Datenbankpartitionsserver noch der ausgefallene Datenbankpartitionsserver die Koordinatorpartition ist, werden Agenten, die für diese Anwendung aktiv sind, unterbrochen. Die Koordinatorpartition sendet eine Nachricht über eine Rollback-Operation (ROLLBACK) oder eine Trennoperation (DISCONNECT) an die anderen Datenbankpartitionsserver. Die Transaktion ist nur auf den Datenbankpartitionsservern unbestätigt, die weiterhin aktiv sind, wenn die Koordinatorpartition die Nachricht SQL0279 zurückgibt.

Jeder Prozess (wie z. B. ein Agent oder ein Deadlock-Detektor), der versucht, eine Anforderung an den ausgefallenen Server zu senden, wird informiert, dass er die Anforderung nicht senden kann.

Recovery nach Transaktionsfehler auf dem ausgefallenen Datenbankpartitionsserver

Wenn der Transaktionsfehler zu einer abnormalen Beendigung des Datenbankmanagers führt, können Sie den Befehl **db2start** mit der Option RESTART angeben, um den Datenbankmanager nach dem Neustart der Datenbankpartition erneut zu starten. Falls der Neustart der Datenbankpartition nicht möglich ist, können Sie den Befehl **db2start** auch in einer anderen Datenbankpartition zum Starten des Datenbankmanagers verwenden.

Die abnormale Beendigung des Datenbankmanagers kann zur Inkonsistenz einiger Datenbankpartitionen auf dem Server führen. Um diese Datenbankpartitionen wieder in einen konsistenten Status zu versetzen, kann auf einem Datenbankpartitionsserver wie folgt eine Recovery nach einem Systemabsturz ausgelöst werden:

- Explizit durch den Befehl **RESTART DATABASE**
- Implizit durch eine CONNECT-Anforderung, wenn der Datenbankkonfigurationsparameter *autorestart* auf ON gesetzt ist

Bei der Recovery nach einem Systemabsturz werden die Protokollsätze in den aktiven Protokolldateien erneut angewendet, um sicherzustellen, dass die Ergebnisse aller vollständigen Transaktionen in der Datenbank vorhanden sind. Nachdem alle Änderungen erneut angewendet wurden, werden alle nicht festgeschriebenen Transaktionen *außer* unbestätigten Transaktionen lokal rückgängig gemacht. In einer Umgebung mit partitionierten Datenbanken gibt es zwei Typen unbestätigter Transaktionen:

- Auf einem Datenbankpartitionsserver, der nicht die Koordinatorpartition ist, gilt eine Transaktion als unbestätigt, wenn sie zwar vorbereitet (PREPARE), aber noch nicht festgeschrieben (COMMIT) wurde.
- In der Koordinatorpartition ist eine Transaktion unbestätigt, wenn sie festgeschrieben (COMMIT), aber noch nicht als abgeschlossen protokolliert wurde (d. h., der Protokollsatz FORGET wurde noch nicht geschrieben). Diese Situation tritt ein, wenn der Koordinatoragent noch nicht alle Commitbestätigungen von allen Servern empfangen hat, die für die Anwendung aktiv waren.

Bei der Recovery nach einem Systemabsturz wird versucht, alle unbestätigten Transaktionen durch eine der folgenden Aktionen aufzulösen. Die durchgeführte Aktion ist davon abhängig, ob der Datenbankpartitionsserver die Koordinatorpartition für eine Anwendung war:

- Wenn der Server, der erneut gestartet wird, nicht die Koordinatorpartition für die Anwendung ist, sendet er eine Abfragenachricht an den Koordinatoragenten, um das Ergebnis der Transaktion festzustellen.
- Wenn der erneut gestartete Server die Koordinatorpartition für die Anwendung *ist*, sendet er eine Nachricht an alle anderen Agenten (untergeordneten Agenten), von denen der Koordinatoragent immer noch Commitbestätigungen erwartet.

Es ist möglich, dass durch eine Recovery nach einem Systemabsturz nicht alle unbestätigten Transaktionen aufgelöst werden können. Einige der Datenbankpartitionsserver sind z. B. möglicherweise nicht verfügbar. Wenn die Koordinatorpartition die Recovery nach einem Systemabsturz vor den anderen, an der Transaktion beteiligen Datenbankpartitionen abschließt, kann die Recovery nach einem Systemabsturz die unbestätigten Transaktionen nicht auflösen. Dies ist die erwartete Funktionsweise, da die Recovery nach einem Systemabsturz von jeder Datenbankpartition unabhängig durchgeführt wird. In diesem Fall wird die SQL-Warnung SQL1061W zurückgegeben. Unbestätigte Transaktionen belegen Ressourcen, z. B. Sperren und Speicherbereich für aktive Protokolle. Daher ist es möglich, dass ein Punkt erreicht wird, an dem keine Änderungen an der Datenbank mehr durchgeführt werden können, weil der Speicherbereich für die aktiven Protokolldateien durch unbestätigte Transaktionen belegt ist. Aus diesem Grund sollten Sie nach einer Recovery nach einem Systemabsturz feststellen, ob unbestätigte Transaktionen verblieben sind, und alle Datenbankpartitionsserver, die zur Auflösung der unbestätigten Transaktionen erforderlich sind, so schnell wie möglich wieder verfügbar machen.

Anmerkung: In einer Umgebung mit partitionierten Datenbankservern wird der Befehl RESTART DATABASE auf jedem Knoten einzeln ausgeführt. Um sicherzustellen, dass die Datenbank auf allen Knoten erneut gestartet wird, verwenden Sie den folgenden empfohlenen Befehl:

```
db2_all "db2 restart database <datenbankname>"
```

Wenn mindestens ein Server, der zur Auflösung einer unbestätigten Transaktion benötigt wird, nicht rechtzeitig wieder verfügbar gemacht werden kann und der Zugriff auf Datenbankpartitionen auf anderen Servern erforderlich ist, können Sie die unbestätigte Transaktion durch eine heuristische Entscheidung manuell auflösen. Mithilfe des Befehls **LIST INDOUBT TRANSACTIONS** können Sie die unbestätigte Transaktion auf dem Server abfragen, festschreiben oder rückgängig machen.

Anmerkung: Der Befehl **LIST INDOUBT TRANSACTIONS** wird auch in einer verteilten Transaktionsumgebung verwendet. Zur Unterscheidung zwischen den beiden Typen unbestätigter Transaktionen enthält das Feld für die Quelle (*Originator*) in der Ausgabe des Befehls **LIST INDOUBT TRANSACTIONS** eine der folgenden Angaben:

- DB2 Enterprise Server Edition. Dies zeigt an, dass die Transaktion aus einer Umgebung mit partitionierten Datenbanken stammt.
- XA. Dies zeigt an, dass die Transaktion aus einer verteilten Umgebung stammt.

Identifizieren des ausgefallenen Datenbankpartitionsservers

Wenn ein Datenbankpartitionsserver ausfällt, empfängt die Anwendung normalerweise einen der folgenden SQLCODE-Werte. Die Methode zum Identifizieren des jeweils ausgefallenen Datenbankmanagers hängt vom empfangenen SQLCODE-Wert ab:

SQL0279N
Dieser SQLCODE-Wert wird empfangen, wenn ein Datenbankpartitionsserver, der an einer Transaktion beteiligt ist, während der Commitverarbeitung beendet wird.

SQL1224N
Dieser SQLCODE-Wert wird empfangen, wenn der ausgefallene Datenbankpartitionsserver die Koordinatorpartition für die Transaktion ist.

SQL1229N
Dieser SQLCODE-Wert wird empfangen, wenn der ausgefallene Datenbankpartitionsserver nicht die Koordinatorpartition für die Transaktion ist.

Welcher Datenbankpartitionsserver ausgefallen ist, kann in zwei Schritten festgestellt werden.

1. Suchen Sie im SQL-Kommunikationsbereich den Partitionsserver, der den Fehler festgestellt hat. Der SQL-Kommunikationsbereich, der zum SQLCODE-Wert SQL1229N gehört, enthält in der sechsten Feldposition des Felds *sqlerrd* die Knotennummer des Servers, der den Fehler erkannte. (Die Knotennummer, die für den Server geschrieben wird, entspricht der Knotennummer in der Datei `db2nodes.cfg`.)
2. Ermitteln Sie für den in Schritt 1 gefundenen Server im Protokoll mit Benachrichtigungen für die Systemverwaltung die Knotennummer des ausgefallenen Servers.

Anmerkung: Wenn mehrere logische Knoten auf einem Prozessor verwendet werden, kann der Ausfall eines logischen Knotens den Ausfall anderer logischer Knoten auf demselben Prozessor verursachen.

Recovery nach dem Ausfall eines Datenbankpartitionsservers

Führen Sie eine Recovery nach dem Ausfall eines Datenbankpartitionsservers durch, indem Sie den zugrunde liegenden Fehler ermitteln und beheben.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Recovery nach dem Ausfall eines Datenbankpartitionsservers auszuführen.

1. Beheben Sie den Fehler, der den Ausfall verursachte.
2. Starten Sie den Datenbankmanager erneut, indem Sie auf einem beliebigen Datenbankpartitionsserver den Befehl **db2start** absetzen.
3. Starten Sie die Datenbank erneut, indem Sie auf dem bzw. den ausgefallenen Datenbankpartitionsserver(n) den Befehl **RESTART DATABASE** absetzen.

Wiederherstellen von unbestätigten Transaktionen auf Mainframe- oder Mid-Range-Servern

Recovery von unbestätigten Transaktionen auf dem Host, wenn bei DB2 Connect der DB2-Synchronisationspunktmanager konfiguriert ist

Wenn Ihre Anwendung auf einen Host- oder einen System i-Datenbankserver während einer Transaktion zugegriffen hat, gibt es einige Unterschiede, wie für unbestätigte Transaktionen eine Recovery durchgeführt wird. Für den Zugriff auf den Host- oder System i-Datenbankserver wird DB2 Connect verwendet. Die Schritte zur Recovery unterscheiden sich, wenn bei DB2 Connect der DB2-Synchronisationspunktmanager konfiguriert ist.

Informationen zu diesem Vorgang

Die Recovery von unbestätigten Transaktionen auf dem Host- oder System i-Server wird normalerweise automatisch vom Transaktionsmanager (TM) und dem DB2-Synchronisationspunktmanager (SPM) ausgeführt. Eine unbestätigte Transaktion auf einem Host- oder System i-Server belegt keine Ressourcen an der lokalen DB2-Position, sondern auf dem Host- oder System i-Server, so lange die Transaktion an dieser Position unbestätigt ist. Wenn der Administrator des Host- oder System i-Servers feststellt, dass eine heuristische Entscheidung getroffen werden muss, kann sich der Administrator an den lokalen DB2-Datenbankadministrator wenden (z. B . telefonisch), um festzustellen, ob für die Transaktion auf dem Host- oder System i-Server eine Commitoperation oder ein Rollback durchgeführt werden soll. Ist dies der Fall, kann mit dem Befehl **LIST DRDA INDOUBT TRANSACTIONS** der Status der Transaktion in der DB2 Connect-Instanz ermittelt werden.

Vorgehensweise

In den meisten Fällen können Sie in einer SNA-Kommunikationsumgebung folgendermaßen vorgehen:

1. Stellen Sie eine Verbindung zum SPM her. Beispiel:

```
db2 => connect to db2spm

   Datenbankverbindungsinformationen

 Datenbankserver                 = SPM0500
 SQL-Berechtigungs-ID            = CRUS
 Aliasname der lokalen Datenbank = DB2SPM
```

2. Führen Sie den Befehl **LIST DRDA INDOUBT TRANSACTIONS** aus, um die dem SPM bekannten unbestätigten Transaktionen anzuzeigen.

   Das folgende Beispiel zeigt eine dem SPM bekannte unbestätigte Transaktion. Der Datenbankname `db_name` ist der lokale Aliasname für den Host- oder System i-Server. Der Partner-LU-Name `partner_lu` ist der vollständig qualifizierte LU-Name des Host- oder System i-Servers. Dies erlaubt die bestmögliche Identifikation des Host- oder System i-Servers und sollte vom Anrufer des Host- oder System i-Servers bereitgestellt werden. Die Kennung der logischen Arbeitseinheit `luwid` stellt eine eindeutige Kennung für eine Transaktion bereit und ist auf allen Hosts und System i-Servern verfügbar. Wenn die fragliche Transaktion angezeigt wird, kann anhand des Feldes `uow_status` das Ergebnis der Transaktion festgestellt werden, wenn der Wert `C` (Commit) oder `R` (Rollback) ist. Wenn Sie den Befehl **LIST DRDA INDOUBT TRANSACTIONS** mit dem Parameter **WITH PROMPTING** absetzen, können Sie die Transaktion interaktiv festschreiben, rückgängig machen oder ignorieren.

```
db2 => list drda indoubt transactions
 Unbestätigte DRDA-Transaktionen:
1.db_name: DBAS3    db_alias: DBAS3    role: AR
  uow_status: C  partner_status: I  partner_lu: USIBMSY.SY12DQA
corr_tok: USIBMST.STB3327L
 luwid: USIBMST.STB3327.305DFDA5DC00.0001
 xid: 53514C2000000017 00000000544D4442 0000000000305DFD A63055E962000000
           00035F
```

3. Wenn eine unbestätigte Transaktion für `partner_lu` und für `luwid` nicht angezeigt wird bzw. wenn der Befehl **LIST DRDA INDOUBT TRANSACTIONS** folgende Ausgabe ergibt:

```
db2 => list drda indoubt transactions
SQL1251W  Keine Daten für manuelle Abfrage zurückgegeben.
```

Nächste Schritte

Es gibt jedoch noch eine weitere Situation, die zwar unwahrscheinlich ist, aber dennoch auftreten kann. Wenn eine unbestätigte Transaktion mit einer ordnungsgemäßen `luwid` für `partner_lu` angezeigt wird, aber der `uow_status` den Wert "I" aufweist, kann der SPM nicht entscheiden, ob die Transaktion festzuschreiben oder mit **ROLLBACK** rückgängig zu machen ist. In dieser Situation sollten Sie den Parameter **WITH PROMPTING** verwenden, um für die Transaktion entweder eine Commitoperation oder ein Rollback auf der DB2 Connect-Workstation durchzuführen. Lassen Sie DB2 Connect anschließend mit dem Host oder System i-Server auf der Basis der heuristischen Entscheidung resynchronisieren.

Recovery von unbestätigten Transaktionen auf dem Host, wenn DB2 Connect nicht den DB2-Synchronisationspunktmanager verwendet

Wenn Ihre Anwendung auf einen Host- oder einen System i-Datenbankserver während einer Transaktion zugegriffen hat, gibt es einige Unterschiede, wie für unbestätigte Transaktionen eine Recovery durchgeführt wird. Für den Zugriff auf Host- oder System i-Datenbankserver wird DB2 Connect verwendet. Die Recoveryschritte sind unterschiedlich, wenn bei DB2 Connect der DB2-Synchronisationspunktmanager konfiguriert ist.

Informationen zu diesem Vorgang

Verwenden Sie die Informationen in diesem Abschnitt, wenn die TCP/IP-Konnektivität verwendet wird, um DB2 for z/OS in einer Aktualisierung auf mehreren Systemen entweder über DB2 Connect Personal Edition oder DB2 Connect Enterprise Edition zu aktualisieren und der DB2-Synchronisationspunktmanager nicht verwendet wird. Die Recovery unbestätigter Transaktionen ist in diesem Fall anders als bei Verwendung des DB2-Synchronisationspunktmanagers.Wenn eine unbestätigte Transaktion in dieser Umgebung auftritt, wird auf dem Client, auf dem Datenbankserver und/oder in der Transaktionsmanagerdatenbank (TMD), je nachdem, wo der Fehler festgestellt wurde, ein Alert-Eintrag generiert. Der Alert-Eintrag wird in die Datei `db2alert.log` geschrieben.

Die Resynchronisation aller unbestätigten Transaktionen erfolgt automatisch, sobald der Transaktionsmanager und alle beteiligten Datenbanken sowie ihre Verbindungen wieder verfügbar sind. Es ist besser, eine automatische Resynchronisation zuzulassen, als heuristisch eine Entscheidung beim Datenbankserver herbeizuführen. Wenn dies jedoch erforderlich ist, gehen Sie wie im Folgenden beschrieben vor.

Anmerkung: Da der DB2-Synchronisationspunktmanager nicht verwendet wird, können Sie den Befehl **LIST DRDA INDOUBT TRANSACTIONS** nicht verwenden.

Vorgehensweise

1. Setzen Sie auf dem z/OS-Host den Befehl DISPLAY THREAD TYPE(INDOUBT) ab.

   Stellen Sie anhand dieser Liste die Transaktion fest, die Sie heuristisch beenden möchten. Weitere Informationen zum Befehl DISPLAY finden Sie im Handbuch *DB2 for z/OS Command Reference*. Die angezeigte LUWID kann derselben luwid in der Transaktionsmanagerdatenbank zugeordnet werden.
2. Setzen Sie den Befehl RECOVER THREAD(<LUWID>) ACTION(ABORT|COMMIT) in Abhängigkeit von der Aktion ab, die Sie ausführen möchten.

   Weitere Informationen zum Befehl RECOVER THREAD finden Sie im Handbuch *DB2 for z/OS Command Reference*.

Recovery nach einem Katastrophenfall

Unter dem Begriff *Recovery nach einem Katastrophenfall* werden die Aktivitäten zusammengefasst, die zum Wiederherstellen der Datenbank nach einem Brand, einem Erdbeben, nach Vandalismus oder anderen zerstörerischen Ereignissen erforderlich sind.

Ein Plan zur Recovery nach einem Katastrophenfall kann Folgendes vorsehen:

- Einen Zweitstandort, der im Notfall zur Verfügung steht
- Eine andere Maschine, auf der die Datenbank wiederhergestellt werden kann
- Die Aufbewahrung von Datenbankbackups, Tabellenbereichsbackups oder beiden sowie archivierten Protokollen an einem anderen Standort

Wenn Ihr Plan zur Recovery nach einem Katastrophenfall vorsieht, die gesamte Datenbank auf einer anderen Maschine wiederherzustellen, wird empfohlen, zumindest über ein Datenbankgesamtbackup und alle archivierten Protokolldateien für die Datenbank zu verfügen. Es ist zwar auch möglich, eine Datenbank erneut zu erstellen (Rebuild), wenn ein Gesamtbackup aller Tabellenbereiche in der Datenbank vorhanden ist, jedoch kann diese Methode eine große Anzahl von Backup-Images erfordern und mehr Zeit in Anspruch nehmen als die Recovery mithilfe eines Datenbankgesamtbackups.

Sie können auch eine Bereitschaftsdatenbank auf dem aktuellen Stand halten, indem Sie die Protokolle, die archiviert werden, auf sie anwenden. Oder Sie können die Datenbank- oder Tabellenbereichsbackups und die Protokollarchive am Bereitschaftsstandort lagern und einen Restore bzw. eine aktualisierende Recovery nur durchführen, wenn ein Katastrophenfall eingetreten ist. (Im letzteren Fall sind möglichst junge Backup-Images vorzuziehen.) Im Katastrophenfall ist es gewöhnlich jedoch nicht möglich, alle Transaktionen bis zum Zeitpunkt des Eintritts der Katastrophe wiederherzustellen.

Der Nutzen eines Tabellenbereichsbackups zur Recovery nach einem Katastrophenfall hängt vom Ausmaß der Beschädigung ab. In der Regel ist eine Recovery nach einem Katastrophenfall weniger kompliziert und zeitaufwendig, wenn Sie die gesamte Datenbank wiederherstellen. Daher sollte am Bereitschaftsstandort ein Datenbankgesamtbackup bereitgehalten werden. Im Fall einer beschädigten Platte kann die Recovery mithilfe eines Tabellenbereichsbackups jedes Tabellenbereichs auf dieser Platte durchgeführt werden. Wenn Sie aufgrund eines Plattenfehlers

(oder aus einem anderen Grund) keinen Zugriff auf einen Container mehr haben, können Sie den Container an einer anderen Position wiederherstellen.

Eine weitere Möglichkeit, Ihre Daten vor einem partiellen oder vollständigen Standortausfall zu schützen, ist die Implementierung der DB2-Funktion HADR (High Availability Disaster Recovery). Nach der Installation und Konfiguration von HADR bietet diese Funktion Schutz vor Datenverlust durch Replizieren von Datenänderungen aus einer Quellendatenbank, der so genannten Primärdatenbank, in eine Zieldatenbank, der so genannten Bereitschaftsdatenbank.

Sie können Ihre Daten gegen einen partiellen oder vollständigen Standortausfall auch durch Replikation schützen. Die Replikation ermöglicht ein regelmäßiges Kopieren von Daten in mehrere ferne Datenbanken. Die DB2-Datenbank stellt eine Reihe von Replikationstools zur Verfügung, bei denen Sie angeben können, welche Daten zu kopieren sind, in welche Datenbanktabellen die Daten zu kopieren sind und wie häufig Aktualisierungen zu kopieren sind.

Darüber hinaus kann auch eine Speicherspiegelung, z. B. mithilfe von Peer-to-Peer Remote Copy (PPRC), zum Schutz Ihrer Daten eingesetzt werden. PPRC ermöglicht ein synchrones Kopieren eines Datenträgers bzw. einer Platte zum Schutz gegen Katastrophen.

DB2-Datenbankprodukte stellen Ihnen zur Planung der Recovery nach einem Katastrophenfall mehrere Optionen zur Auswahl. Abhängig von Ihren Geschäftsanforderungen können Sie entweder Tabellenbereichsbackups oder Datenbankgesamtbackups als Schutz gegen Datenverlust verwenden oder überlegen, ob sich Ihre Umgebung für eine Lösung wie HADR eignet. Für welche Lösung Sie sich auch entscheiden, Sie sollten Ihre Recoveryprozeduren in jedem Fall in einer Testumgebung testen, bevor Sie sie in Ihrer Produktionsumgebung implementieren.

Versionsrecovery

Eine *Versionsrecovery* ist die Wiederherstellung einer früheren Version der Datenbank mithilfe eines Images der Datenbank, das im Rahmen einer Backup-Operation erstellt wurde.

Sie können diese Recoverymethode mit nicht wiederherstellbaren Datenbanken verwenden (d. h. Datenbanken, für die Sie über keine archivierten Protokolldateien verfügen). Sie können diese Methode auch bei wiederherstellbaren Datenbanken einsetzen, indem Sie mit dem Befehl **RESTORE DATABASE** die Option **WITHOUT ROLLING FORWARD** verwenden.

Durch eine RESTORE-Operation der Datenbank wird die gesamte Datenbank mithilfe eines zu einem früheren Zeitpunkt erstellten Backup-Images wiederhergestellt. Ein Datenbank-Backup ermöglicht es Ihnen, eine Datenbank in dem Status wiederherzustellen, in dem sie sich zum Zeitpunkt des Backups befand. Es gehen jedoch alle UOWs verloren, die seit dem Zeitpunkt des Backups bis zum Eintreten des Fehlers ausgeführt wurden (siehe Abb. 21 auf Seite 389).

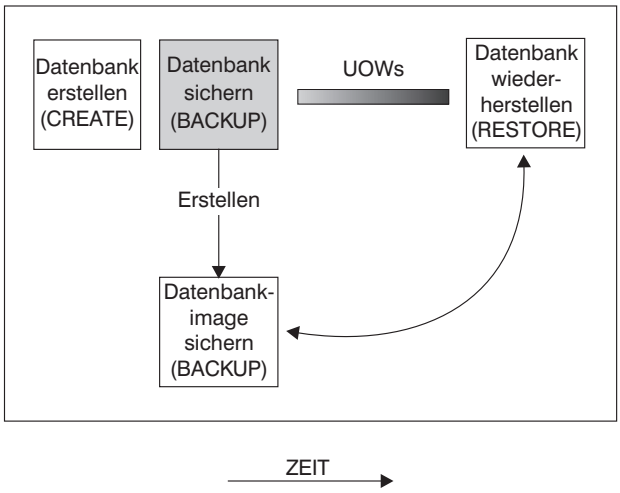


Abbildung 21. Versionsrecovery

Bei Einsatz der Methode der Versionsrecovery müssen Sie regelmäßige Gesamtbackups der Datenbank planen und durchführen.

In einer Umgebung mit partitionierten Datenbanken ist eine Datenbank auf viele Datenbankpartitionsserver (oder Knoten) verteilt. Sie müssen alle Datenbankpartitionen wiederherstellen, und die Backup-Images, die Sie zum Restore der Datenbank verwenden, müssen alle zum gleichen Zeitpunkt erstellt worden sein. (Jede Datenbankpartition wird separat gesichert und wiederhergestellt.) Ein Backup, bei dem alle Backupkopien der Datenbankpartitionen zur gleichen Zeit erstellt werden, wird als *Versionsbackup* bezeichnet.

Aktualisierende Recovery

Sie können eine aktualisierende Recovery für Datenbanken oder Tabellenbereiche ausführen.

Sie können die Methode der *aktualisierenden Recovery* nur dann einsetzen, wenn Sie ein Backup der Datenbank erstellt und die Protokolldateien archiviert haben (dazu muss einer der Datenbankkonfigurationsparameter **logarchmeth1** oder **logarchmeth2** auf einen anderen Wert als OFF gesetzt sein). Der Restore der Datenbank unter Angabe des Parameters **WITHOUT ROLLING FORWARD** (d. h. ohne aktualisierende Recovery) entspricht der Methode der Versionsrecovery. Die Datenbank wird in einem Status wiederhergestellt, der dem Zeitpunkt entspricht, zu dem das Image des Offline-Backups erstellt wurde. Wenn Sie für die Datenbank einen Restore durchführen und *nicht* den Parameter **WITHOUT ROLLING FORWARD** für die Restoreoperation der Datenbank angeben, befindet sich die Datenbank am Ende der Restoreoperation im Status "Aktualisierende Recovery anstehend". Dadurch kann die aktualisierende Recovery durchgeführt werden.

Anmerkung: Der Parameter **WITHOUT ROLLING FORWARD** kann in folgenden Fällen nicht verwendet werden:

- Sie führen die RESTORE-Operation mithilfe eines Online-Backup-Images aus.
- Sie setzen eine RESTORE-Operation auf Tabellenbereichsebene ab.

Während einer Recovery werden archivierte Protokolldateien aus dem Archiv abgerufen. Wenn die archivierten Protokolldateien komprimiert sind, werden die Dateien automatisch dekomprimiert und verwendet. Archivierte Protokolldateien, die

sich im Pfad für aktive Protokolldateien oder im Überlaufprotokollpfad befinden, werden ebenfalls automatisch dekomprimiert, wenn sie von Ihnen manuell in diesen Pfad kopiert wurden.

Es können zwei Typen der aktualisierenden Recovery in Erwägung gezogen werden:

- *Aktualisierende Recovery der Datenbank*. Bei diesem Typ der aktualisierenden Recovery werden im Anschluss an den Restore der Datenbank Transaktionen angewendet, die in den Datenbankprotokollen aufgezeichnet sind (siehe Abb. 22). In den Datenbankprotokollen werden alle Änderungen aufgezeichnet, die an der Datenbank vorgenommen werden. Diese Methode vervollständigt die Recovery der Datenbank bis zu ihrem Status an einem bestimmten Zeitpunkt oder bis zu ihrem Status unmittelbar vor dem Fehler, das heißt bis zum Ende der aktiven Protokolldateien.

  In einer Umgebung mit partitionierten Datenbanken ist eine Datenbank über viele Datenbankpartitionen verteilt, und der Befehl **ROLLFORWARD DATABASE** muss in der Datenbankpartition abgesetzt werden, in der sich die Katalogtabellen für die Datenbank befinden (Katalogpartition). Wenn Sie eine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt (punktuelle Recovery) durchführen, müssen alle Datenbankpartitionen aktualisierend wiederhergestellt werden, um sicherzustellen, dass alle Datenbankpartitionen den gleichen Stand haben. Wenn Sie eine einzelne Datenbankpartition wiederherstellen müssen, können Sie eine aktualisierende Recovery bis zum Ende der Protokolle durchführen, um sie auf den gleichen Stand wie die anderen Datenbankpartitionen in der Datenbank zu bringen. Bei der aktualisierenden Recovery einer einzelnen Datenbankpartition kann nur die Recovery bis zum Ende der Protokolle verwendet werden. Die Recovery bis zu einem bestimmten Zeitpunkt wird immer auf *alle* Datenbankpartitionen angewendet.

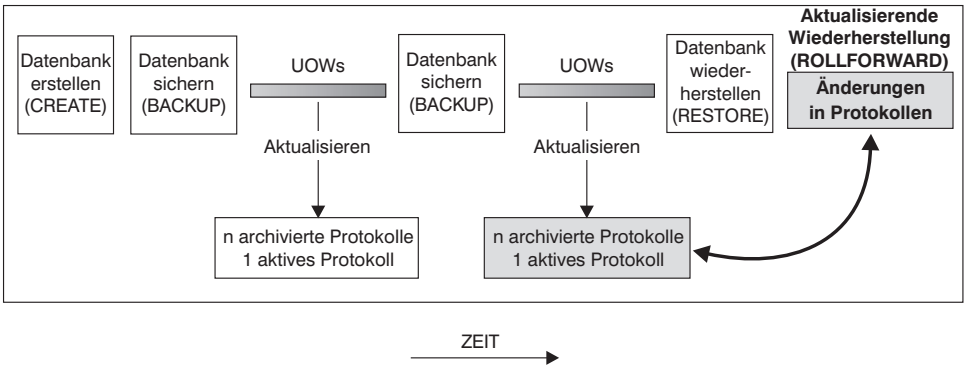


Abbildung 22. Aktualisierende Recovery einer Datenbank. Im Falle einer lange laufenden Transaktion kann es mehr als ein aktives Protokoll geben.

- *Aktualisierende Recovery eines Tabellenbereichs*. Wenn für eine Datenbank die aktualisierende Recovery aktiviert ist, können auch Tabellenbereiche gesichert, wiederhergestellt und aktualisierend wiederhergestellt werden (siehe Abb. 23 auf Seite 391). Zum Wiederherstellen und aktualisierenden Wiederherstellen eines Tabellenbereichs benötigen Sie ein Backup-Image entweder der gesamten Datenbank (d. h. aller Tabellenbereiche) oder mindestens eines einzelnen Tabellenbereichs. Außerdem benötigen Sie die Protokollsätze, die die wiederherzustellenden Tabellenbereiche betreffen. Sie können einen Tabellenbereich durch die Protokolle bis zu einem von zwei Punkten aktualisierend wiederherstellen:
  - Bis zum Ende der Protokolldateien - oder -
  - Bis zu einem bestimmten Zeitpunkt (bezeichnet als *Punktuelle Recovery*)

Eine aktualisierende Recovery von Tabellenbereichen kann in den beiden folgenden Situationen durchgeführt werden:

- Ein Tabellenbereich befindet sich nach seinem Restore immer im Status *Aktualisierende Recovery anstehend* und muss aktualisierend wiederhergestellt werden. Rufen Sie den Befehl **ROLLFORWARD DATABASE** auf, um die Protokolle auf die Tabellenbereiche entweder bis zu einem bestimmten Zeitpunkt oder bis zum Ende der Protokolldateien anzuwenden.
- Wenn sich mindestens ein Tabellenbereich nach einer Recovery infolge eines Systemabsturzes im Status *Aktualisierende Recovery anstehend* befindet, beheben Sie zuerst das Problem mit dem Tabellenbereich. In einigen Fällen kann ein Fehler am Tabellenbereich ohne Restore der Datenbank behoben werden. Beispielsweise kann ein Spannungsverlust den Tabellenbereich in den Status *Aktualisierende Recovery anstehend* versetzen. Ein Restore der Datenbank ist in diesem Fall nicht erforderlich. Nachdem das Problem mit dem Tabellenbereich behoben ist, können Sie den Befehl **ROLLFORWARD DATABASE** verwenden, um die Protokolle bis zum Ende der Protokolldateien auf die Tabellenbereiche anzuwenden. Wenn der Fehler vor der Recovery nach einem Systemabsturz behoben wird, reicht diese Recovery möglicherweise aus, um die Datenbank in einen konsistenten, verwendbaren Status zu versetzen.

  Anmerkung: Wenn der fehlerhafte Tabellenbereich die Systemkatalogtabellen enthält, können Sie die Datenbank nicht starten. Sie müssen den Tabellenbereich SYSCATSPACE wiederherstellen und anschließend eine aktualisierende Recovery bis zum Ende der Protokolldateien durchführen.

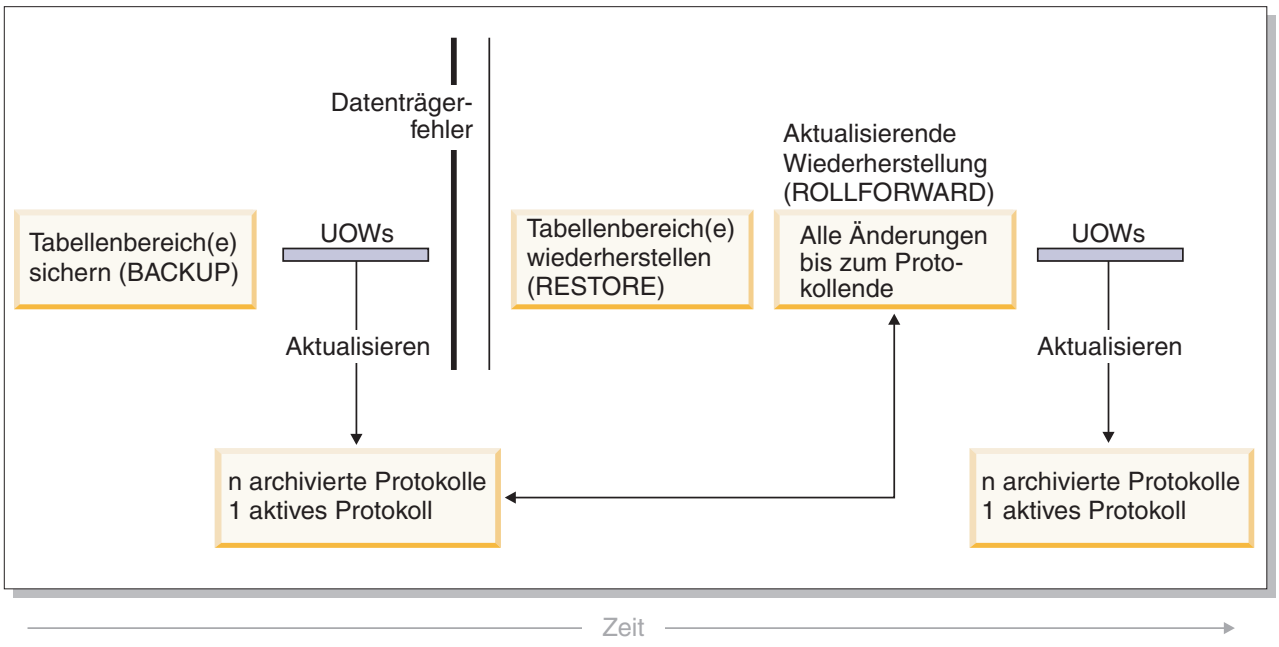


Abbildung 23. Aktualisierende Recovery von Tabellenbereichen. Im Falle einer lange laufenden Transaktion kann es mehr als ein aktives Protokoll geben.

Wenn Sie in einer Umgebung mit partitionierten Datenbanken einen Tabellenbereich bis *zu einem bestimmten Zeitpunkt* aktualisierend wiederherstellen, müssen Sie die Liste der Datenbankpartitionen nicht angeben, auf die sich der Tabellenbereich verteilt. Der DB2-Datenbankmanager übergibt die Anforderung zur aktualisieren-

den Recovery an alle Datenbankpartitionen. Dies bedeutet, dass der Tabellenbereich in allen Datenbankpartitionen, auf denen der Tabellenbereich sich befindet, wiederhergestellt werden muss.

Wenn Sie in einer Umgebung mit partitionierten Datenbanken einen Tabellenbereich bis *zum Ende der Protokolle* aktualisierend wiederherstellen, müssen Sie die Liste der Datenbankpartitionen angeben, wenn Sie den Tabellenbereich *nicht* in allen Datenbankpartitionen aktualisierend wiederherstellen wollen. Wenn Sie alle Tabellenbereiche (in allen Datenbankpartitionen), die sich im Status *Aktualisierende Recovery anstehend* befinden, bis zum Ende der Protokolle aktualisierend wiederherstellen wollen, müssen Sie die Liste der Datenbankpartitionen nicht angeben. Standardmäßig wird die Anforderung zur aktualisierenden Recovery der Datenbank an alle Datenbankpartitionen gesendet.

Aktualisierende Recoveryoperationen von Tabellenbereichen weisen in einer DB2 pureScale-Umgebung ein anderes Verhalten auf. Weitere Informationen finden Sie in „Zusammenführung von Protokolldatenströmen und Protokolldateiverwaltung in einer DB2 pureScale-Umgebung" auf Seite 297 und in „Protokollfolgenummern in DB2 pureScale-Umgebungen" auf Seite 302.

Wenn Sie einen Tabellenbereich aktualisierend wiederherstellen, der einen Teil einer partitionierten Tabelle enthält, und Sie ihn bis zu einem bestimmten Zeitpunkt aktualisierend wiederherstellen, müssen Sie auch alle anderen Tabellenbereiche, in denen sich die Tabelle befindet, bis zu demselben Zeitpunkt aktualisierend wiederherstellen. Sie können einen einzelnen Tabellenbereich, der einen Teil einer partitionierten Tabelle enthält, jedoch bis zum Ende der Protokolle aktualisierend wiederherstellen.

Wenn eine partitionierte Tabelle (mit ATTACH) zugeordnete, (mit DETACH) getrennte oder (mit DROP) gelöschte Datenpartitionen hat, muss die aktualisierende Recovery bis zu einem bestimmten Zeitpunkt auch alle Tabellenbereiche für diese Datenpartitionen berücksichtigen. Um zu bestimmen, ob eine partitionierte Tabelle zugeordnete, getrennte oder gelöschte Datenpartitionen hat, fragen Sie die Katalogtabelle SYSDATAPARTITIONS ab.

Änderung von Speichergruppen während der aktualisierenden Recovery

Ob Pfadänderungen für Speichergruppen während einer aktualisierenden Recovery wiederholt werden, hängt davon ab, ob Sie die Speichergruppe während des Restoreprozesses umgeleitet haben. Wenn Sie eine Speichergruppe während der Restoreoperation für die Datenbank nicht neu definiert haben, werden die Speichergruppe oder deren Pfade betreffende Protokollsätze während der aktualisierenden Recovery wiederholt. Aktualisierungen der Speicherpfade, Umbenennungsoperationen für Speichergruppen und Aktualisierungen der Zuordnung von Tabellenbereichen zu Speichergruppen, die in den Protokollsätzen beschrieben werden, werden während der aktualisierenden Recovery angewendet. Wenn eine aktualisierende Recovery versucht, einen das Hinzufügen von Speicherpfaden oder das Erstellen einer Speichergruppe betreffenden Protokollsatz zu wiederholen, und ein Speicherpfad nicht gefunden wird, wird Fehler SQL1051N zurückgegeben.

Wenn Sie Speicherpfade während der Restoreoperation neu definiert haben, wiederholt die aktualisierende Recovery keine Änderungen an den Speicherpfaden oder Datenträgerattributen von Speichergruppen, deren Pfade Sie umgeleitet haben. Änderungen am Datentag oder dem Namen von Speichergruppen werden jedoch wiederholt. Außerdem werden Protokollsätze für andere Operationen, ein-

schließlich DROP STOGROUP-Operationen, wiederholt. Es wird davon ausgegangen, dass für alle explizit angegebenen Speichergruppenpfade die gewünschten Endpfade festgelegt wurden.

Wenn eine Neuausgleichsoperation im Protokoll festgestellt wird, werden Operationen zum Neuausgleich des Tabellenbereichs während der aktualisierenden Recovery initialisiert. Die Neuausgleichsoperationen werden möglicherweise nicht während der Ausführung der aktualisierenden Recovery beendet. In diesem Fall wird die Verarbeitung des Neuausgleichs bei Beenden der aktualisierenden Recovery ausgesetzt und beim nächsten Aktivieren der Datenbank erneut gestartet.

Wird während einer aktualisierenden Recovery eine Anweisung CREATE STOGROUP im Protokoll festgestellt, wird die Speichergruppe in den Pfaden erstellt, die Sie beim Absetzen der Anweisung CREATE STOGROUP angegeben haben.

Inkrementelles Backup und inkrementelle Recovery

Da die Größe von Datenbanken, speziell die von Warehouses, in zunehmenden Maße Terabyte- und Petabyte-Bereiche erreicht, steigen die für das Backup und die Recovery solcher Datenbanken erforderlichen Zeitaufwände und Anforderungen an die Hardwareressourcen enorm.

Beim Umgang mit sehr umfangreichen Datenbanken ist es nicht immer die beste Vorgehensweise, jeweils die gesamte Datenbank mit allen Tabellenbereichen zu sichern, da der Speicherbedarf für mehrere Kopien enorm groß ist.

Bedenken Sie Folgendes:

- Wenn nur ein kleiner Prozentsatz der Daten in einem Warehouse geändert wird, dürfte es nicht notwendig sein, die gesamte Datenbank zu sichern.
- Das Hinzufügen von Tabellenbereichen zu vorhandenen Datenbanken und das ausschließliche Sichern von Tabellenbereichen ist riskant, da nicht gewährleistet ist, dass zwischen zwei Tabellenbereichsbackups keine weiteren Änderungen an anderen, beim Backup nicht berücksichtigten Tabellenbereichen vorgenommen wurden.

Um dieser Problematik gerecht zu werden, bietet DB2 die Möglichkeit eines inkrementellen Backups und einer inkrementellen Recovery.

Ein *inkrementelles Backup* ist ein Backup-Image, das nur die Seiten enthält, die seit dem letzten Backup aktualisiert wurden. Neben aktualisierten Daten und Indexseiten enthält jedes Image eines inkrementellen Backups auch die vollständigen ursprünglichen Metadaten der Datenbank (wie Datenbankkonfiguration, Tabellenbereichsdefinitionen, Datenbankprotokolle usw.), die normalerweise in Images von Gesamtbackups enthalten sind.

Anmerkung:

1. Enthält ein Tabellenbereich Langfeld- oder LOB-Daten (große Objekte) und wird ein inkrementelles Backup erstellt, werden alle Langfeld- oder LOB-Daten in das Backup-Image kopiert, wenn irgendwelche Seiten im betreffenden Tabellenbereich seit dem letzten Backup geändert wurden.
2. Wenn Sie ein inkrementelles Backup eines Tabellenbereichs erstellen, der eine benutzte Seite (d. h. eine Seite mit Daten, die geändert, aber noch nicht auf Platte geschrieben wurden) enthält, werden alle LOB-Daten im Backup gesichert. Normale Daten werden nur gesichert, wenn sie geändert wurden.

3. Bei der Datenumverteilung werden möglicherweise Tabellenbereiche für alle neuen Datenbankpartitionen erstellt, wenn der Parameter **ADD DBPARTITIONNUMS** im Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** angegeben wird. Dies kann sich auf die Ausführung von Operationen für das inkrementelle Backup auswirken.

Es werden zwei Typen inkrementeller Backups unterstützt:

- *Inkrementell*. Bei einem inkrementellen Backup enthält das Backup-Image eine Kopie aller Datenbankdaten, die seit dem zuletzt erfolgreich durchgeführten Gesamtbackup geändert wurden. Dies wird auch als ein kumulatives Backup-Image bezeichnet, da die über einen gewissen Zeitraum hinweg erstellten inkrementellen Backups jeweils den Inhalt des vorherigen Images eines inkrementellen Backups enthalten. Der Vorgänger eines inkrementellen Backup-Image ist jeweils das neueste erfolgreiche Gesamtbackup desselben Objekts.
- *Delta*. Ein Deltabackup-Image bzw. ein inkrementelles Deltabackup ist eine Kopie aller Datenbankdaten, die seit dem zuletzt erfolgreich durchgeführten Backup (Gesamtbackup, inkrementelles Backup oder Deltabackup) des betreffenden Tabellenbereichs geändert wurden. Dies wird auch als differenzielles oder nicht kumulatives Backup-Image bezeichnet. Der Vorgänger eines Deltabackup-Images ist das jüngste erfolgreiche Backup, das eine Kopie aller im Deltabackup-Image enthaltenen Tabellenbereiche enthält.

Der wesentliche Unterschied zwischen Images von inkrementellen Backups und Delta-Backup-Images wird deutlich beim Erstellen von aufeinander folgenden Backups eines Objekts, das kontinuierlichen Änderungen unterworfen ist. Jedes weitere erstellte inkrementelle Image enthält den vollständigen Inhalt des zuvor erstellten inkrementellen Images sowie alle seit dem letzten Gesamtbackup geänderten oder neu hinzugekommenen Daten. Deltabackup-Images enthalten hingegen nur die seit dem letzten Backup geänderten Seiten, wobei der Typ des Backups keine Rolle spielt.

Kombinationen von inkrementellen Backups von Datenbanken und Tabellenbereichen sind zulässig, sowohl im Online- als auch im Offlinemodus. Gehen Sie bei der Planung Ihrer Backup-Strategie sorgfältig vor, da bei der Kombination aus inkrementellen Backups von Datenbanken und Tabellenbereichen nicht ausgeschlossen werden kann, dass es sich bei dem Vorgänger eines Datenbank-Backups (oder eines Tabellenbereichs-Backups mehrerer Tabellenbereiche) anstatt um ein Einzelimage um eine eindeutige Gruppe von zuvor zu unterschiedlichen Zeitpunkten erstellten Backups von Datenbanken und Tabellenbereichen handelt.

Um die Datenbank oder den Tabellenbereich in einem konsistenten Status wiederherzustellen, müssen Sie zur Recovery ein konsistentes Image des gesamten wiederherzustellenden Objekts (Datenbank oder Tabellenbereich) verwenden und anschließend alle erforderlichen inkrementellen Backup-Images in der Reihenfolge anwenden, die in nachfolgender Liste beschrieben ist.

Zur Aktivierung der Überwachung von Datenbankaktualisierungen unterstützt DB2 den neuen Datenbankkonfigurationsparameter **trackmod**. Dieser Parameter kann einen der beiden folgenden Werte haben:

- `NO`. Inkrementelle Backups sind bei dieser Konfiguration nicht zulässig. Aktualisierungen von Datenbankseiten werden weder verfolgt noch aufgezeichnet. Dies ist der Standardwert.
- `YES`. Inkrementelle Backups sind bei dieser Konfiguration zulässig. Wird die Aktualisierungsüberwachung aktiviert, wird diese Änderung bei der nächsten erfolgreichen Verbindungsherstellung zur Datenbank wirksam. Bevor für einen

bestimmten Tabellenbereich ein inkrementelles Backups durchgeführt werden kann, muss ein Gesamtbackup erstellt werden.

Bei SMS- und DMS-Tabellenbereichen erfolgt die Unterteilung dieser Überwachung auf Tabellenbereichsebene. Bei der Überwachung auf Tabellenbereichsebene zeigt eine Markierung für jeden Tabellenbereich an, ob sich in dem Tabellenbereich Seiten befinden, die gesichert werden müssen. Falls in einem Tabellenbereich keine Seiten gesichert werden müssen, kann die Backup-Operation diesen Tabellenbereich komplett überspringen.

Die Überwachung von Datenbankaktualisierungen kann sich, wenn auch nur minimal, auf die Laufzeitleistung von Transaktionen auswirken, die Daten aktualisieren oder einfügen.

Restore von inkrementellen Backup-Images

Eine Restoreoperation, bei der inkrementelle Backup-Images verwendet werden, umfasst vier Schritte.

Informationen zu diesem Vorgang

1. Identifizieren des inkrementellen Zielimages.

   Ermitteln Sie das endgültige wiederherzustellende Image, und fordern Sie eine Operation zum inkrementellen Restore vom DB2-Restoredienstprogramm an. Dieses Image wird als Zielimage des inkrementellen Restores bezeichnet, da es das letzte Image ist, das wiederhergestellt wird. Das Zielimage des inkrementellen Restores wird mit dem Parameter **TAKEN AT** im Befehl **RESTORE DATABASE** angegeben.
2. Wiederherstellen des neuesten Gesamtbackups der Datenbank oder des Tabellenbereichs, um die Basis für die nachfolgend anzuwendenden inkrementellen Backup-Images zu erstellen.
3. Wiederherstellen aller erforderlichen Gesamtbackup-Images oder inkrementellen Tabellenbereichsbackup-Images in der Reihenfolge ihrer Erstellung, aufbauend auf dem in Schritt 2 wiederhergestellten Basisimage.
4. Wiederholen von Schritt 3, bis das Zielimage aus Schritt 1 zum zweiten Mal gelesen wird. Auf das Zielimage wird während einer Operation zum vollständigen inkrementellen Restore zweimal zugegriffen. Beim ersten Zugriff werden noch keine Benutzerdaten, sondern nur die Anfangsdaten aus dem Image gelesen. Erst beim zweiten Zugriff wird das Image vollständig gelesen und verarbeitet.

   Der zweifache Zugriff auf das Zielimage der Operation zum inkrementellen Restore soll sicherstellen, dass die Datenbank zu Beginn korrekt konfiguriert wird, d. h. mit dem richtigen Protokoll sowie den korrekten Datenbankkonfigurations- und Tabellenbereichsdefinitionen für die Datenbank, die während der Restoreoperation erstellt wird. In Fällen, in denen ein Tabellenbereich seit der Erstellung des ersten Gesamtbackup-Image der Datenbank gelöscht wurde, werden die Tabellenbereichsdaten für dieses Image zwar aus den Backup-Images gelesen, aber bei der Verarbeitung des inkrementellen Restores ignoriert.

Inkrementelle Backup-Images können auf zwei verschiedene Weisen wiederhergestellt werden, nämlich automatisch oder manuell.

- Bei einem automatischen inkrementellen Restore wird der Befehl **RESTORE DATABASE** nur einmal abgesetzt, um das gewünschte Zielimage anzugeben. DB2 for Linux, UNIX and Windows ermittelt die verbliebenen erforderlichen Backup-Images anhand des Datenbankprotokolls und stellt sie wieder her.

- Bei einem manuellen inkrementellen Restore muss der Befehl **RESTORE DATABASE** einmal für jedes einzelne Backup-Image abgesetzt werden, das wiederhergestellt werden muss (wie in den zuvor aufgelisteten Schritten erläutert).

Vorgehensweise

- Zum Ausführen eines automatischen Restores einer Gruppe inkrementeller Backup-Images setzen Sie den Befehl **RESTORE DATABASE** ab und geben Sie hierbei im Parameter **TAKEN AT** folgendermaßen die Zeitmarke für das letzte Image an, das Sie wiederherstellen wollen:

  ```
  db2 restore db sample incremental automatic taken at zeitmarke
  ```

  Dieser Befehl veranlasst das Restoredienstprogramm, die am Anfang dieses Abschnitts aufgeführten Schritte automatisch auszuführen. Während der Anfangsphase der Verarbeitung wird das Backup-Image mit der (im Format *jjjjmmtthhmmss*) angegebenen Zeitmarke gelesen, und das Restoredienstprogramm prüft, ob die Datenbank, ihr Protokoll und die Tabellenbereichsdefinitionen vorhanden und gültig sind.

  Während der zweiten Verarbeitungsphase wird das Datenbankprotokoll abgefragt, um eine Kette von Backup-Images zu bilden, die zum Ausführen der angeforderten Restoreoperation erforderlich sind. Wenn dies aus einem bestimmten Grund nicht möglich ist und DB2 for Linux, UNIX and Windows keine vollständige Kette der erforderlichen Images erstellen kann, wird die Restoreoperation beendet und eine Fehlernachricht zurückgegeben. In diesem Fall ist ein automatischer inkrementeller Restore nicht möglich, sodass Sie den Befehl **RESTORE DATABASE** mit dem Parameter **INCREMENTAL ABORT** absetzen müssen. Dadurch werden alle übrigen Ressourcen bereinigt, sodass Sie mit einem manuellen inkrementellen Restore fortfahren können.

  Anmerkung: Es wird dringend empfohlen, den Parameter **WITH FORCE OPTION** des Befehls **PRUNE HISTORY** nicht zu verwenden. Die Standardoperation dieses Befehls verhindert, dass Sie Protokolleinträge löschen, die möglicherweise für die Recovery unter Verwendung des zuletzt erstellten Gesamtbackup-Image der Datenbank erforderlich sind. Mit der Option **WITH FORCE OPTION** ist es jedoch möglich, Einträge zu löschen, die für eine automatische Restoreoperation benötigt werden.

  Während der dritten Verarbeitungsphase stellt DB2 for Linux, UNIX and Windows alle verbliebenen Backup-Images in der generierten Kette wieder her. Wenn in dieser Phase ein Fehler auftritt, müssen Sie den Befehl **RESTORE DATABASE** mit der Option **INCREMENTAL ABORT** absetzen, um die übrigen Ressourcen zu bereinigen. Anschließend müssen Sie feststellen, ob der Fehler behoben werden kann, bevor Sie den Befehl **RESTORE DATABASE** erneut absetzen bzw. erneut einen manuellen inkrementellen Restore versuchen.

- Zum Ausführen eines manuellen Restores einer Gruppe inkrementeller Backup-Images setzen Sie Befehle **RESTORE DATABASE** ab und geben Sie hierbei im Parameter **TAKEN AT** folgendermaßen die Zeitmarke für jedes Image an, das Sie wiederherstellen wollen:

  1.

     ```
     db2 restore database datenbankname incremental taken at zeitmarke
     ```

     Dabei verweist *zeitmarke* auf das letzte inkrementelle Backup-Image (*Ziel-Image*), für das ein Restore durchgeführt werden soll.

  2.

     ```
     db2 restore database datenbankname incremental taken at zeitmarke1
     ```

Dabei verweist *zeitmarke1* auf das anfängliche vollständige Datenbank-Image (oder Tabellenbereichs-Image).

3.
```
db2 restore database datenbank incremental taken at zeitmarkeX
```

   Dabei verweist *zeitmarkeX* auf alle inkrementellen Backup-Images in der Reihenfolge der Erstellung.

4. Wiederholen Sie Schritt 3, und führen Sie für jedes inkrementelle Backup-Image einen Restore bis einschließlich des Images *zeitmarke* aus.

Wenn Sie eine Restoreoperation für eine Datenbank ausführen und zuvor Backup-Images von Tabellenbereichen erstellt wurden, müssen die Tabellenbereichsimages in der chronologischen Reihenfolge der Zeitmarken ihres Backups wiederhergestellt werden.

Mit dem Dienstprogramm **db2ckrst** können Sie das Datenbankprotokoll abfragen und eine Liste der Zeitmarken der für einen inkrementellen Restore erforderlichen Backup-Images generieren. Außerdem wird eine vereinfachte Restoresyntax für einen manuellen inkrementellen Restore generiert. Es empfiehlt sich, über alle Backups vollständig Buch zu führen und dieses Dienstprogramm nur zur Orientierung zu verwenden.

Automatischer inkrementeller Restore - Einschränkungen

Der automatische inkrementelle Restore kann bei einem Datenbankrestore nützlich sein. Bei der Entscheidung darüber, welche Methode für die Datenbankrecovery verwendet werden soll, müssen jedoch die für den automatischen inkrementellen Restore geltenden Einschränkungen beachtet werden, um unnötige Probleme zu vermeiden.

Die folgenden Einschränkungen gelten für den automatischen inkrementellen Restore:

1. Wenn ein Tabellenbereichsname seit dem Backup, von dem Sie einen Restore durchführen wollen, geändert wurde und Sie den neuen Namen in der Restoreoperation auf Tabellenbereichsebene verwenden, wird die erforderliche Kette von Backup-Images nicht korrekt aus dem Datenbankprotokoll generiert, sodass ein Fehler auftritt (SQL2571N).

   Beispiel:

   ```
   db2 backup db sample -> <zm1>
   db2 backup db sample incremental -> <zm2>
   db2 rename tablespace from userspace1 to t1
   db2 restore db sample tablespace ('t1') incremental automatic taken
   at <zm2>


   SQL2571N Ein automatischer inkrementeller Restore kann nicht fortgesetzt werden.
   Ursachencode: "3".
   ```

   Mögliche Lösung: Nehmen Sie einen manuellen inkrementellen Restore vor.

2. Wenn Sie eine Datenbank löschen, wird das Datenbankprotokoll gelöscht. Wenn Sie die gelöschte Datenbank wiederherstellen, wird das Datenbankprotokoll in dem Status wiederhergestellt, den es zum Zeitpunkt der Erstellung des wiederhergestellten Backup-Images hatte. Alle nach diesem Zeitpunkt erstellten Protokolleinträge gehen verloren. Wenn Sie versuchen, einen automatischen inkrementellen Restore durchzuführen, für die einige dieser verloren gegangenen Protokolleinträge benötigt würden, versucht das Dienstprogramm RESTORE,

eine falsche Backup-Image-Kette zu erstellen, und gibt eine entsprechende Fehlermeldung ("falsche Reihenfolge") aus (SQL2572N).

Beispiel:

```
db2 backup db sample -> <zm1>
db2 backup db sample incremental -> <zm2>
db2 backup db sample incremental delta -> <zm3>
db2 backup db sample incremental delta -> <zm4>
db2 drop db sample
db2 restore db sample incremental automatic taken at <zm2>
db2 restore db sample incremental automatic taken at <zm4>
```

Mögliche Lösung:

- Nehmen Sie einen manuellen inkrementellen Restore vor.
- Stellen Sie zuerst die Protokolldatei unter Verwendung von Image <zm4> wieder her, bevor Sie einen automatischen inkrementellen Restore starten.

3. Wenn Sie ein Backup-Image aus einer Datenbank in eine andere Datenbank wiederherstellen und anschließend ein inkrementelles Backup oder Deltabackup durchführen, können Sie zum Restore dieses Backup-Images keinen automatischen inkrementellen Restore mehr verwenden.

   Beispiel:

   ```
   db2 create db a
   db2 create db b

   db2 update db cfg for a using trackmod on

   db2 backup db a -> zm1
   db2 restore db a taken at zm1 into b

   db2 backup db b incremental -> zm2

   db2 restore db b incremental automatic taken at zm2

   SQL2542N Es wurde keine Übereinstimmung für ein Backup-Image der Datenbankdatei
   anhand des Aliasnamens der Quellendatenbank "datenbankalias" und der angegebenen
   Zeitmarke "zeitmarke" gefunden.
   ```

   Empfohlene Fehlerumgehung:

   - Nehmen Sie wie folgt einen manuellen inkrementellen Restore vor:

     ```
     db2 restore db b incremental taken at zm2
     db2 restore db a incremental taken at zm1 into b
     db2 restore db b incremental taken at zm2
     ```

   - Nach der manuellen Restoreoperation in Datenbank B führen Sie ein Datenbankgesamtbackup aus, um eine neue Kette von inkrementellen Backups zu beginnen.

Optimieren der Recoveryleistung

Mithilfe bestimmter Strategien können Sie die Leistung von DB2 während der Datenbankrecovery verbessern und den Zeitaufwand für die Recovery nach einem DB2-Serviceausfall reduzieren.

Folgendes ist im Hinblick auf die Recoveryleistung zu beachten:

- Sie können die Leistung für Datenbanken, die häufig aktualisiert werden, verbessern, indem Sie die Protokolldateien auf einer separaten Einheit speichern. In einer OLTP-Umgebung (Online Transaction Processing - Onlinetransaktionsverarbeitung) ist oftmals mehr E/A-Aufwand für das Schreiben von Daten in die Protokolle als für das Speichern einer Zeile von Daten erforderlich. Durch das Speichern der Protokolldateien auf einer separaten Einheit wird die Bewegung des

Plattenzugriffsarms minimiert, die zum Wechseln zwischen einer Protokolldatei und den Datenbankdateien erforderlich ist.

Berücksichtigen Sie dabei außerdem, welche anderen Dateien sich auf der Platte befinden. Wenn die Protokolldateien z. B. auf eine Platte versetzt werden, die auch für das System-Paging in einem System verwendet wird, das nicht über genügend Realspeicher verfügt, werden dadurch Ihre Optimierungsversuche zunichte gemacht.

DB2-Datenbankprodukte versuchen automatisch, die für das Ausführen einer Backup- oder Restoreoperation benötigte Zeit zu minimieren, indem optimale Werte für die Anzahl Puffer, die Puffergröße und die Parallelitätseinstellungen ausgewählt werden. Die Werte basieren auf der Menge des für Dienstprogramme verfügbaren Zwischenspeichers, der Anzahl verfügbarer Prozessoren und der Datenbankkonfiguration.

- Verwenden Sie mehrere Quelleneinheiten, um die zur Durchführung einer Restoreoperation benötigte Zeit zu verringern.
- Wenn eine Datenbank große Mengen von Langfeld- und LOB-Daten enthält, kann das Wiederherstellen der Datenbank sehr viel Zeit in Anspruch nehmen. Wenn für die Datenbank die aktualisierende Recovery aktiviert ist, bietet der Befehl **RESTORE** die Möglichkeit, ausgewählte Tabellenbereiche wiederherzustellen. Handelt es sich bei Ihren Langfeld- und LOB-Daten um wichtige Unternehmensdaten, sollten Sie den Zeitaufwand für das Wiederherstellen dieser Tabellenbereiche gegen den für die Backup-Operation dieser Tabellenbereiche erforderlichen Zeitaufwand abwägen. Wenn die Langfeld- und LOB-Daten in separaten Tabellenbereichen gespeichert werden, lässt sich der für das Wiederherstellen von Daten erforderliche Zeitaufwand dadurch verringern, dass die Tabellenbereiche mit den Langfeld- und LOB-Daten vom Restore ausgeschlossen werden. Wenn die LOB-Daten von einer getrennten Quelle reproduziert werden können, sollten Sie beim Erstellen oder Ändern einer Tabelle zum Hinzufügen von LOB-Spalten die Option NOT LOGGED verwenden. Wenn Sie die Tabellenbereiche, die Langfeld- und LOB-Daten enthalten, nicht wiederherstellen wollen, aber die Tabellenbereiche wiederherstellen müssen, die die Tabelle enthalten, müssen Sie die aktualisierende Recovery bis zum Ende der Protokolle durchführen, sodass alle Tabellenbereiche, die die Tabellendaten enthalten, konsistent sind.

  Anmerkung: Wenn Sie einen Tabellenbereich sichern, der Tabellendaten ohne die zugehörigen LONG- oder LOB-Felder enthält, können Sie keine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt für diesen Tabellenbereich durchführen. Alle Tabellenbereiche für eine Tabelle müssen gleichzeitig bis zum selben Zeitpunkt aktualisierend wiederhergestellt werden.
- Folgendes gilt für Backup- und Restoreoperationen:
  - Es sollten mehrere Einheiten verwendet werden.
  - Überlasten Sie die Bandbreite der E/A-Einheitencontroller nicht.
- DB2-Datenbankprodukte verwenden zur Durchführung einer Recovery nach einem Systemabsturz und zur aktualisierenden Datenbankrecovery mehrere Agenten. Sie werden während dieser Operationen eine bessere Leistung feststellen, vor allem auf Maschinen mit symmetrischen Multiprozessoren (SMP). Die Verwendung mehrerer Agenten bei der Datenbankrecovery nutzt die Vorteile der zusätzlichen CPUs auf SMP-Maschinen.

  Der mit der parallelen Recovery eingeführte Agententyp lautet **db2agnsc**. DB2-Datenbankmanager wählen Sie die zu verwendende Anzahl Agenten auf der Basis der vorhandenen Anzahl CPUs auf der Maschine aus.

  DB2-Datenbankmanager verteilen Protokollsätze an diese Agenten, sodass sie nach Bedarf gleichzeitig erneut angewendet werden können. Auf diese Weise

kann z. B. die Verarbeitung von Protokollsätzen für Einfügungen, Löschungen, Aktualisierungen sowie das Hinzufügen und Löschen von Schlüsseln parallel ausgeführt werden. Da die Protokollsätze auf Seitenebene parallel ausgeführt werden (Protokollsätze derselben Datenseite werden von demselben Agent verarbeitet), wird die Leistung verbessert, auch wenn alle Prozesse für dieselbe Tabelle ausgeführt werden.

- Wenn Sie eine Recoveryoperation ausführen, wählen DB2-Datenbankmanager automatisch optimale Werte für die Anzahl Puffer, die Puffergröße und die Parallelitätseinstellungen aus. Die Werte basieren auf der Menge des für Dienstprogramme verfügbaren Zwischenspeichers, der Anzahl verfügbarer Prozessoren und der Datenbankkonfiguration. Daher sollten Sie je nach der Größe der auf Ihrem System verfügbaren Speicherkapazität in Betracht ziehen, mehr Speicher zuzuordnen, indem Sie den Wert des Konfigurationsparameters **util_heap_sz** erhöhen.

Erforderliche Zugriffsrechte und Berechtigungen zur Verwendung des Recoverydienstprogramms

Sie benötigen die Berechtigung SYSADM, SYSCTRL oder SYSMAINT, um das Recoverydienstprogramm verwenden zu können.

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf diese zuzugreifen. Berechtigungsstufen stellen eine Methode dar, um Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zusammenzufassen. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte.

Benutzer können nur auf die Objekte zugreifen, für die sie zugriffsberechtigt sind, d. h. für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Kapitel 13. Restore - Übersicht

Mithilfe der DB2-Restore-Tools können Sie die DB2-Datenbank auf einem früheren Stand wiederherstellen. Ein Backup-Image der Datenbank muss vorhanden sein, bevor diese Tools verwendet werden können.

In der einfachsten Form des DB2-Befehls **RESTORE DATABASE** müssen Sie lediglich den Aliasnamen der Datenbank angeben, die Sie wiederherstellen möchten. Beispiel:

```
db2 restore db sample
```

Da die Datenbank SAMPLE vorhanden ist und beim Absetzen des Befehls **RESTORE DATABASE** ersetzt wird, wird in diesem Beispiel die folgende Nachricht zurückgegeben:

```
SQL2539W  Achtung! Restore in eine bestehende Datenbank, die mit der
Datenbank des Backup-Images identisch ist. Die Datenbankdateien werden gelöscht.
Fortfahren ? (j/n)
```

Wenn Sie `j` angeben, sollte die Restoreoperation erfolgreich beendet werden können.

Für den Restore einer Datenbank ist eine exklusive Verbindung erforderlich, d. h., es können keine Anwendungen für die Datenbank ausgeführt werden, sobald die Operation gestartet ist. Das Restoredienstprogramm verhindert, dass andere Anwendungen vor der erfolgreichen Beendigung der Operation auf die Datenbank zugreifen. Der Restore eines Tabellenbereichs kann hingegen online erfolgen.

Ein Tabellenbereich kann erst nach der erfolgreichen Beendigung der Recoveryoperation (und evtl. anschließender aktualisierender Recovery) wieder verwendet werden.

Bei Tabellen, die sich über mehrere Tabellenbereiche erstrecken, ist es ratsam, die betreffende Gruppe von Tabellenbereichen zusammen zu sichern (und wiederherzustellen).

Bei der partiellen oder selektiven Restoreoperation können Sie ein Backup-Image auf Tabellenbereichsebene oder ein Image eines vollständigen Datenbank-Backups verwenden, aus dem Sie mindestens einen Tabellenbereich auswählen. Alle Protokolldateien, die diesen Tabellenbereichen ab dem Zeitpunkt der Erstellung des Backup-Images zugeordnet wurden, müssen vorhanden sein.

Sie können eine Datenbank aus einem Backup-Image, das auf einer 32-Bit-Version erstellt wurde, in einer 64–Bit-Version wiederherstellen, nicht jedoch umgekehrt.

Beim Restore von Backups aus 32-Bit-Umgebungen auf 64-Bit-Umgebungen müssen Sie die Datenbankkonfigurationsparameter überprüfen, um sicherzustellen, dass diese für die 64-Bit-Instanzumbgebung optimiert sind. So ist zum Beispiel der Standardwert für den Anweisungszwischenspeicher in 32-Bit-Umgebungen niedriger als in 64-Bit-Umgebungen.

Für das Backup und die Wiederherstellung Ihrer Datenbanken sollten Sie die DB2-Dienstprogramme Backup und Restore verwenden. Das Verschieben einer Datei-

gruppe von einer Maschine auf eine andere wird nicht empfohlen, da dies zu einer Beeinträchtigung der Datenbankintegrität führen kann.

Unter bestimmten Bedingungen können Sie transportierbare Gruppen im Befehl **RESTORE DATABASE** angeben, um Datenbanken zu versetzen. .

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Restore von Datenbankbackups. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Verwenden von Restore

Verwenden Sie den Befehl **RESTORE DATABASE**, um eine Datenbank oder einen Tabellenbereich nach einem Problem, zum Beispiel einem Datenträger-, Speicher- oder Anwendungsfehler, wiederherzustellen. Wenn Sie Ihre Datenbank oder einzelne Tabellenbereiche mit Backup gesichert haben, können Sie diese wiederherstellen, wenn sie beschädigt oder fehlerhaft sind.

Vorbereitende Schritte

Beim Restore in eine *vorhandene* Datenbank sollte noch keine Verbindung zu der wiederherzustellenden Datenbank bestehen. Das Restoredienstprogramm stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der Restoreoperation beendet wird. Beim Restore in eine *neue* Datenbank ist zum Erstellen der Datenbank die Herstellung einer Verbindung zu einer Instanz (mit ATTACH) erforderlich. Beim Restore in eine *neue ferne* Datenbank müssen Sie zuerst eine Verbindung zu der Instanz herstellen, in der sich die neue Datenbank befinden soll. Erstellen Sie anschließend die neue Datenbank, indem Sie die Codepage und das Gebiet des Servers angeben. Beim Restore wird die Codepage der Zieldatenbank mit der des Backup-Images überschrieben.

Informationen zu diesem Vorgang

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln.

Für das Restoredienstprogramm gelten die folgenden Einschränkungen:

- Das Restoredienstprogramm kann nur verwendet werden, wenn die betreffende Datenbank zuvor mit dem DB2-Backup-Dienstprogramm gesichert wurde.
- Wenn Benutzer, bei denen es sich nicht um den Instanzeigner (unter UNIX) oder um Mitglieder der DB2ADMNS- oder Administratorengruppe (unter Windows) handelt, versuchen, ein Backup-Image wiederherzustellen, wird eine Fehlernachricht (SQL2061N) ausgegeben. Wenn andere Benutzer Zugriff auf das Backup-Image benötigen, müssen die Dateizugriffsrechte nach dem Erstellen des Backups geändert werden.
- Es kann keine Operation zum Restore der Datenbank gestartet werden, wenn ein Prozess zur aktualisierenden Recovery aktiv ist.
- Wenn Sie die Option **TRANSPORT** nicht angeben, können Sie einen Tabellenbereich nur dann in einer vorhandenen Datenbank wiederherstellen, wenn der Tabellenbereich zurzeit vorhanden und mit dem wiederherzustellenden identisch ist. Hier bedeutet „identisch", dass der Tabellenbereich nicht zwischen der Backup- und der Restoreoperation gelöscht und erneut erstellt wurde. Bei der Datenbank auf der Platte und der Datenbank in dem Backup-Image muss es sich um die gleiche Datenbank handeln.

- Es ist nicht möglich, einen Restore auf Tabellenbereichsebene eines Tabellenbereichsbackups in einer neuen Datenbank auszuführen.
- Es es nicht möglich, eine Online-Restoreoperation auf Tabellenbereichsebene mit den Systemkatalogtabellen auszuführen.
- Sie können keinen Restore eines Backups, das in einer Umgebung mit einer Einzeldatenbankpartition erstellt wurde, in einer vorhandenen Umgebung mit partitionierten Datenbanken durchführen. Stattdessen müssen Sie das Backup in einer Umgebung mit einer Einzeldatenbankpartition wiederherstellen und anschließend die benötigten Datenbankpartitionen hinzufügen.
- Beim Restore eines Backup-Images mit einer Codepage in ein System mit einer anderen Codepage wird die Codepage des Systems von der Codepage des Backup-Images überschrieben.
- Mit dem Befehl **RESTORE DATABASE** können Sie keine durch nicht dynamischen Speicher aktivierten Tabellenbereiche in einen durch dynamischen Speicher aktivierten Tabellenbereich konvertieren.
- Für das Angeben der Option **TRANSPORT** gelten die folgenden Einschränkungen:
  - Wenn das Backup-Image durch eine Restoreoperation wiederhergestellt werden kann und für Upgrades unterstützt wird, kann es transportiert werden.
  - Bei Verwendung eines Online-Backups müssen die Quellen- und der Zieldatenserver unter der derselben DB2-Version ausgeführt werden.
  - Der Befehl **RESTORE DATABASE** muss in der Zieldatenbank ausgeführt werden. Wenn der ferne Client dieselbe Plattform wie der Server verwendet, kann der Schematransport lokal auf dem Server oder über eine ferne Instanzverbindung erfolgen. Wenn die Zieldatenbank eine katalogisierte ferne Datenbank in der Instanz ist, in der der Transport lokal ausgeführt wird, wird der Schematransport in die ferne Zieldatenbank nicht unterstützt.
  - Tabellenbereiche und Schemata können nur in eine vorhandene Datenbank transportiert werden. Die Transportoperation erstellt keine neue Datenbank. Zum Wiederherstellen einer Datenbank in einer neuen Datenbank können Sie den Befehl **RESTORE DATABASE** ohne die Option **TRANSPORT** verwenden.
  - Wenn die Schemata in der Quellendatenbank durch DB2-Sicherheitseinstellungen oder -Berechtigungen geschützt werden, behalten die transportierten Schemata in der Zieldatenbank diese Attribute bei.
  - Transport wird nicht unterstützt für Umgebungen mit partitionierten Datenbanken.
  - Wenn eine der Tabellen innerhalb des Schemas eine XML-Spalte enthält, schlägt der Transport fehl.
  - Die Option **TRANSPORT** ist nicht kompatibel mit der Option `REBUILD`.
  - Die Option **TRANSPORT** wird für die Wiederherstellung aus einem Momentaufnahmebackup-Image nicht unterstützt.
  - Die Zieldatenbank muss für die Datenbankwiederherstellung aktiviert sein.
  - Die Bereitstellungsdatenbank wird nur für die Transportoperation erstellt. Sie kann nicht für andere Operationen verwendet werden.
  - Die Datenbankkonfigurationsparameter für die Staging-Tabelle und die Zieltabelle müssen identisch sein, andernfalls schlägt die Transportoperation mit einem Inkompatibilitätsfehler fehl.
  - Der Konfigurationsparameter **auto_reval** auto_reval muss in der Zieldatenbank auf `deferred_force` gesetzt sein, damit als ungültig angegebene Objekte transportiert werden. Andernfalls schlägt der Transport fehl.
  - Wenn ein Online-Backup-Image verwendet wird und die aktiven Protokolle nicht enthalten sind, schlägt die Transportoperation fehl.

  - Bei Verwendung eines Online-Backups muss das Backup-Image mit der Option INCLUDE LOGS erstellt worden sein.
  - Wenn das Backup-Image aus einer früheren Version stammt, muss es ein vollständiges Offline-Backup-Image auf Datenbankebene sein.
  - Wenn in der Bereitstellungs- oder Zieldatenbank ein Fehler auftritt, muss die gesamte Wiederherstellungsoperation erneut ausgeführt werden. Alle auftretenden Fehler werden in der **db2diag**-Protokolldatei auf dem Zielserver erfasst und sollten vor dem erneuten Ausführen des Befehls **RESTORE** überprüft werden.
  - Wenn der Transportclient fehlschlägt, wird die Bereitstellungsdatenbank möglicherweise nicht ordnungsgemäß bereinigt. In diesem Fall muss die Bereitstellungsdatenbank gelöscht werden. Löschen Sie vor dem erneuten Ausführen des Befehls **RESTORE** alle Bereitstellungsdatenbanken, damit keine Container der Bereitstellungsdatenbanken die nachfolgende Transportoperation blockieren können.
  - Gleichzeitig ablaufende Transportoperationen für dieselbe Zieldatenbank werden nicht unterstützt.
  - Die Scripterstellung für umgeleitete Restoreoperationen wird beim Transportieren von Tabellenbereichen nicht unterstützt.
- Sie können einen Restore für einen Tabellenbereich durchführen, wenn die Speichergruppe aktualisiert worden ist. Die Zielspeichergruppe während des Restores des Tabellenbereichs ist die Speichergruppe, der der Tabellenbereich zur Zeit der Ausführung des Befehls **RESTORE** zugeordnet ist.
- Sie können keine punktuelle Recovery bis zu einer früheren Speichergruppenzuordnung ausführen.

Vorgehensweise

Gehen Sie wie folgt vor, um das Restoredienstprogramm aufzurufen:

- Setzen Sie den Befehl **RESTORE DATABASE** ab.
- Rufen Sie die Anwendungsprogrammierschnittstelle (API) db2Restore auf.
- Öffnen Sie den Taskassistenten in IBM Data Studio für den Befehl **RESTORE DATABASE**.

Beispiel

Das folgende Beispiel zeigt einen Befehl **RESTORE DATABASE**, der über den CLP abgesetzt wird:

```
db2 restore db sample from D:\DB2Backups taken at 20010320122644
```

Restore für ein Momentaufnahmebackup-Image

Bei einem Restore für ein Momentaufnahmebackup wird zum Kopieren von Daten die leistungsstarke Kopiertechnologie einer Speichereinheit genutzt.

Vorbereitende Schritte

Zur Ausführung von Backup- und Restoreoperationen für Momentaufnahmen benötigen Sie einen DB2 ACS-API-Treiber für Ihre Speichereinheit. Eine Liste der unterstützten Speicherhardware für den integrierten Treiber enthält die Tivoli-Dokumentation im Abschnitt Unterstützte Speichersubsysteme.

Sie müssen ein Momentaufnahmebackup durchführen, bevor ein Restore für ein Momentaufnahmebackup möglich ist. Siehe hierzu „Durchführen eines Momentaufnahmebackups” auf Seite 330.

Vorgehensweise

Sie können den Restore für ein Momentaufnahmebackup unter Verwendung des Befehls **RESTORE DATABASE** mit dem Parameter **USE SNAPSHOT** oder der API db2Restore mit dem Datenträgertyp SQLU_SNAPSHOT_MEDIA durchführen.

- Befehl **RESTORE DATABASE**:

```
db2 restore db sample use snapshot
```

- API db2Restore:

```
int sampleRestoreFunction( char dbAlias[],
                                                    char restoredDbAlias[],
                                                                  char user[],
                                                    char pswd[],
                           char workingPath[] )
{
  db2MediaListStruct mediaListStruct = { 0 };

  rmediaListStruct.locations = &workingPath;
    rmediaListStruct.numLocations = 1;
  rmediaListStruct.locationType = SQLU_SNAPSHOT_MEDIA;


  db2RestoreStruct restoreStruct = { 0 };

    restoreStruct.piSourceDBAlias = dbAlias;
    restoreStruct.piTargetDBAlias = restoredDbAlias;
  restoreStruct.piMediaList = &mediaListStruct;
    restoreStruct.piUsername = user;
    restoreStruct.piPassword = pswd;
  restoreStruct.iCallerAction = DB2RESTORE_STORDEF_NOINTERRUPT;


  struct sqlca sqlca = { 0 };


  db2Restore(db2Version900, &restoreStruct, &sqlca);

  return 0;
}
```

Restore in einer vorhandenen Datenbank

Bei einem Restore auf Datenbankebene kann sich das Backup-Image von der vorhandenen Datenbank im Hinblick auf den Aliasnamen, den Datenbanknamen oder die Datenbank*nummer* unterscheiden. Eine Datenbanknummer ist die eindeutige Kennung einer Datenbank, die sich während der Bestehens der Datenbank nicht ändert.

Der Datenbankmanager weist die Nummer zu, wenn Sie die Datenbank erstellen. DB2 verwendet immer die Datenbanknummer aus dem Backup-Image. Sie können einen Tabellenbereich nur dann in einer vorhandenen Datenbank wiederherstellen, wenn der Tabellenbereich vorhanden und mit dem wiederherzustellenden Tabellenbereich identisch ist, d. h., der Tabellenbereich wurde zwischen der Backup- und der Restoreoperation nicht gelöscht und anschließend erneut erstellt. Bei der Datenbank auf der Platte und der Datenbank in dem Backup-Image muss es sich

um die gleiche Datenbank handeln. Sie können beim Restore eines Tabellenbereichs die momentan definierten Speichergruppen nicht modifizieren oder explizit neue Speichergruppen erstellen.

Beim Restore in eine vorhandene Datenbank führt das Restoredienstprogramm die folgenden Aktionen aus:

- Löschen der Tabellen-, Index- und Langfelddaten der vorhandenen Datenbank und Ersetzen dieser Informationen durch die Daten des Backup-Images.
- Ersetzen der Tabelleneinträge jedes Tabellenbereichs, den Sie wiederherstellen.
- Beibehalten der Datei des Recoveryprotokolls, sofern sie nicht beschädigt oder leer ist. Wenn die Datei des Recoveryprotokolls beschädigt ist oder keine Einträge enthält, kopiert der Datenbankmanager die Datei aus dem Backup-Image. Wenn Sie die Datei des Recoveryprotokolls ersetzen möchten, können Sie den Befehl **RESTORE DATABASE** mit dem Parameter **REPLACE HISTORY FILE** ausführen.
- Beibehalten des Authentifizierungstyps für die vorhandene Datenbank.
- Beibehalten der Datenbankverzeichnisse der vorhandenen Datenbank. Die Verzeichnisse definieren die Speicherposition der Datenbank und die Art der Katalogisierung der Datenbank.
- Vergleichen der Datenbanknummern. Wenn sich die Nummern unterscheiden, führt das Dienstprogramm die folgenden Aktionen aus:
  - Löschen der Protokolle, die der vorhandenen Datenbank zugeordnet sind.
  - Kopieren der Datenbankkonfigurationsdatei aus dem Backup-Image.
  - Setzen des Parameters **NEWLOGPATH** für den Befehl **RESTORE DATABASE** auf den Wert des Datenbankkonfigurationsparameters **logpath**, falls der Parameter **NEWLOGPATH** angegeben wurde.

  Wenn die Datenbanknummern identisch sind, führt das Dienstprogramm die folgenden Aktionen aus:
  - Löschen aller Protokolldateien, wenn es sich um ein Image einer nicht wiederherstellbaren Datenbank handelt.
  - Löschen leerer Protokolldateien, wenn es sich um ein Image einer wiederherstellbaren Datenbank handelt. Protokolldateien, die nicht leer sind, sind hiervon nicht betroffen.
  - Beibehalten der aktuellen Datenbankkonfigurationsdatei.
  - Setzen des Parameters **NEWLOGPATH** für den Befehl **RESTORE DATABASE** auf den Wert des Datenbankkonfigurationsparameters **logpath**, falls der Parameter **NEWLOGPATH** angegeben wurde. Andernfalls kopiert das Dienstprogramm den aktuellen Protokollpfad in die Datenbankkonfigurationsdatei. Das Dienstprogramm überprüft den Protokollpfad. Falls die Datenbank den Pfad nicht verwenden kann, ändert das Dienstprogramm die Datenbankkonfiguration so, dass sie den Standardprotokollpfad verwendet.

Restore in einer neuen Datenbank

Sie können eine neue Datenbank erstellen und anschließend das Backup-Image einer vollständigen Datenbank in diese neue Datenbank wiederherstellen. Wenn Sie keine neue Datenbank erstellen, erstellt das Restoredienstprogramm eine neue Datenbank.

Beim Restore in eine neue Datenbank führt das Restoredienstprogramm Folgendes aus:

- Erstellen einer neuen Datenbank unter Verwendung des Datenbankaliasnamens, der im Parameter für den Aliasnamen der Zieldatenbank angegeben wurde.

(Wenn für die Zieldatenbank kein Aliasname angegeben wurde, erstellt das Restoredienstprogramm eine Datenbank mit einem Aliasnamen, der im Parameter für den Aliasnamen der Quellendatenbank angegeben wurde.)

- Wiederherstellen der Konfigurationsdatei der Datenbank aus dem Backup-Image.
- Setzen von **NEWLOGPATH** auf den Wert des Datenbankkonfigurationsparameters **logpath**, falls **NEWLOGPATH** im Befehl **RESTORE DATABASE** angegeben wurde. Das Dienstprogramm überprüft den Protokollpfad: Falls der Pfad von der Datenbank nicht verwendet werden kann, wird die Datenbankkonfiguration so geändert, dass sie den Standardprotokollpfad verwendet.
- Wiederherstellen des Authentifizierungstyps aus dem Backup-Image.
- Wiederherstellen des Inhalts aus den Datenbankverzeichnissen im Backup-Image.
- Wiederherstellen der Datei des Recoveryprotokolls für die Datenbank.
- Überschreiben der Codepage der Datenbank mit der Codepage des Backup-Image.

Verwenden des inkrementellen Restores in einer Test- und Produktionsumgebung

Wenn eine Produktionsdatenbank für inkrementelles Backup und inkrementellen Restore aktiviert wurde, können Sie ein Image eines inkrementellen Backups oder eines Delta-Backups verwenden, um eine Testdatenbank zu erstellen oder zu aktualisieren.

Sie können hierzu entweder den manuellen oder den automatischen inkrementellen Restore verwenden.

Zum Restore des Backup-Images aus der Produktionsdatenbank in die Testdatenbank geben Sie mit dem Befehl **RESTORE DATABASE** die Option INTO *aliasname-der-zieldatenbank* an. Beispiel: In einer Produktionsdatenbank mit den folgenden Backup-Images:

```
backup db prod
Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: ts1

   backup db prod incremental
Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: ts2
```

wäre das folgende ein Beispiel für einen manuellen inkrementellen Restore:

```
restore db prod incremental taken at zm2 into test without
   prompting
   DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.

restore db prod incremental taken at zm1 into test without
   prompting
   DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.

restore db prod incremental taken at zm2 into test without
   prompting
   DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Falls die Datenbank TEST bereits vorhanden ist, überschreibt die Restoreoperation alle bereits vorhandenen Daten. Wenn die Datenbank TEST noch nicht vorhanden ist, wird das Restoredienstprogramm die Datenbank erstellen und anschließend mit den Daten aus den Backup-Images füllen.

Da Operationen zum automatischen inkrementellen Restore vom Datenbankprotokoll abhängen, werden die Restoreschritte leicht abweichen, je nachdem, ob die

Testdatenbank vorhanden ist oder nicht. Wenn ein automatischer inkrementeller Restore in die Datenbank TEST durchgeführt werden soll, muss deren Protokoll das Backup-Image-Protokoll für die Datenbank PROD enthalten. Das Datenbankprotokoll für das Backup-Image ersetzt alle Datenbankprotokolle, die bereits für die Datenbank TEST vorhanden sind, wenn eine der folgenden Bedingungen zutrifft:

- Die Datenbank TEST ist beim Absetzen des Befehls **RESTORE DATABASE** nicht vorhanden.
- Die Datenbank TEST ist beim Absetzen des Befehls **RESTORE DATABASE** bereits vorhanden und das Protokoll der Datenbank TEST enthält keine Datensätze.

Das folgende Beispiel zeigt einen automatischen inkrementellen Restore in die Datenbank TEST, die noch nicht vorhanden ist:

```
restore db prod incremental automatic taken at zm2 into test without
   prompting
   DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Das Restoredienstprogramm erstellt die Datenbank TEST und füllt sie mit Daten.

Wenn die Datenbank TEST vorhanden ist und das Datenbankprotokoll Einträge enthält, müssen Sie die Datenbank vor dem automatischen inkrementellen Restore wie folgt löschen:

```
drop db test
   DB20000I  Der Befehl DROP DATABASE wurde erfolgreich ausgeführt.

restore db prod incremental automatic taken at zm2 into test without
   prompting
   DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Wenn Sie die Datenbank nicht löschen wollen, können Sie den Befehl **PRUNE HISTORY** mit einer weit in der Zukunft liegenden Zeitmarke und dem Parameter **WITH FORCE OPTION** verwenden, bevor Sie den Befehl **RESTORE DATABASE** absetzen:

```
connect to test
   Datenbankverbindungsinformationen

Datenbankserver            = server_id
SQL-Berechtigungs-ID             = id
   Aliasname der lokalen Datenbank = TEST

   prune history 9999 with force option
   DB20000I  Der Befehl PRUNE wurde erfolgreich ausgeführt.

   connect reset
   DB20000I  Der SQL-Befehl wurde erfolgreich ausgeführt.
restore db prod incremental automatic taken at zm2 into test without
   prompting
   SQL2540W  RESTORE wurde erfolgreich ausgeführt, es wurde jedoch eine Warnung
"2539" beim  Restore der Datenbank im Modus 'No Interrupt'
festgestellt.
```

In diesem Fall arbeitet der Befehl **RESTORE DATABASE** genau so, wie wenn die Datenbank TEST nicht vorhanden wäre.

Ist die Datenbank TEST vorhanden und enthält das Datenbankprotokoll keine Einträge, müssen Sie die Datenbank TEST nicht löschen, bevor der automatische inkrementelle Restore durchgeführt wird:

```
restore db prod incremental automatic taken at zm2 into test without
   prompting
   SQL2540W  RESTORE wurde erfolgreich ausgeführt, es wurde jedoch eine Warnung
"2539" beim  Restore der Datenbank im Modus 'No Interrupt'
festgestellt.
```

Sie können weiter inkrementelle Backups oder Deltabackups der Testdatenbank erstellen, ohne zuvor ein Gesamtbackup der Datenbank vorzunehmen. Wenn es später jedoch erforderlich wird, eines der Images eines inkrementellen Backups oder eines Delta-Backups wiederherzustellen, müssen Sie einen manuellen inkrementellen Restore durchführen. Dies liegt daran, dass Operationen zum automatischen inkrementellen Restore erfordern, dass alle während eines automatischen inkrementellen Restores wiederhergestellten Backup-Images aus demselben Datenbankaliasnamen erstellt wurden.

Wenn Sie ein Datenbankgesamtbackup der Testdatenbank erstellen, nachdem Sie die Restoreoperation unter Verwendung des Produktions-Backup-Images beendet haben, können Sie inkrementelle Backups oder Deltabackups erstellen und diese entweder manuell oder im automatischen Modus wiederherstellen.

Ausführen einer umgeleiteten Restoreoperation

Eine Restoreoperation für eine Datenbank verwendet ein Datenbank-Backup-Image zum erneuten Erstellen der Datenbank.

Sie können eine umgeleitete Restoreoperation in den folgenden Situationen verwenden:

- Wenn Sie für ein Backup-Image einen Restore auf einer Zielmaschine durchführen möchten, die sich von der Quellenmaschine unterscheidet.
- Wenn Sie für Ihre Tabellenbereichscontainer einen Restore an einer anderen physischen Position durchführen möchten.
- Wenn Ihre Restoreoperation fehlgeschlagen ist, da auf mindestens einen Container nicht zugegriffen werden konnte.
- Wenn Sie die Pfade einer definierten Speichergruppe neu definieren wollen.

Einschränkungen:

Sie können einen umgeleiteten Restore nicht zum Verschieben von Daten von einem Betriebssystem auf ein anderes verwenden.

Während des Restoreprozesses können Sie keine Speichergruppe erstellen oder löschen.

Während des Restoreprozesses für einen Tabellenbereich können Sie Speichergruppenpfade nicht ändern, auch nicht wenn Sie einen Restore für alle Tabellenbereiche durchführen, die der Speichergruppe zugeordnet sind.

Der Prozess zum Ausführen eines umgeleiteten Restores mithilfe eines inkrementellen Backup-Images ist dem Prozess zum Ausführen eines umgeleiteten Restores mithilfe eines nicht inkrementellen Backup-Images ähnlich. Verwenden Sie eine der folgenden Methoden:

- Führen Sie den Befehl **RESTORE DATABASE** mit dem Parameter **REDIRECT** aus und geben Sie das Backup-Image an, aus dem der inkrementelle Restore der Datenbank erfolgen soll.
- Generieren Sie ein Script zum umgeleiteten Restore aus einem Backup-Image und modifizieren Sie anschließend das Script nach Bedarf.

Bei der Methode mit dem Befehl **RESTORE DATABASE** handelt es sich um einen aus zwei Schritten bestehenden Restoreprozess für Datenbanken mit einem zwischengeschalteten Schritt zum Definieren eines Tabellenbereichscontainers oder eines Speichergruppenpfads. Gehen Sie zum Ausführen eines umgeleiteten Restores wie folgt vor:

1. Setzen Sie den Befehl **RESTORE DATABASE** mit dem Parameter **REDIRECT** ab.
2. Führen Sie einen der folgenden Schritte aus:
   - Definieren Sie Tabellenbereichscontainer durch Absetzen des Befehls **SET TABLESPACE CONTAINERS**.
   - Definieren Sie Speichergruppenpfade für die Datenbank, für die ein Restore ausgeführt werden soll, durch Absetzen des Befehls **SET STOGROUP PATHS**.
3. Setzen Sie den Befehl **RESTORE DATABASE** erneut ab und geben Sie dieses Mal den Parameter **CONTINUE** an.

Nachdem Sie den Befehl **RESTORE CONTINUE** abgesetzt haben, wird der neue Pfad als Pfad des Tabellenbereichscontainers für alle zugehörigen Tabellenbereiche festgelegt. Wenn Sie einen Befehl **LIST TABLESPACE CONTAINERS** oder **GET SNAPSHOT FOR TABLESPACES** nach dem Befehl **SET STOGROUP PATHS** und vor dem Befehl **RESTORE CONTINUE** absetzen, gibt die Ausgabe für die Pfade der Tabellenbereichscontainer nicht die neuen Pfade wieder, die Sie mit dem Befehl **SET STOGROUP PATHS** angegeben haben.

Während einer Operation zum umgeleiteten Restore werden Verzeichnis- und Dateicontainer automatisch erstellt, sofern sie nicht bereits vorhanden sind. Einheitencontainer werden nicht automatisch vom Datenbankmanager erstellt.

DB2-Datenbankprodukte stellen SQL-Anweisungen zum Hinzufügen, Ändern oder Entfernen von DMS-Tabellenbereichen mit nicht-dynamischem Speicher von Tabellenbereichscontainern und Speichergruppenpfaden von Tabellenbereichen mit dynamischem Speicher bereit. Ein umgeleiteter Restore ist die einzige Möglichkeit, eine Konfiguration für SMS-Tabellenbereichscontainer mit nicht-dynamischem Speicher zu modifizieren.

Sie können Tabellenbereichscontainer erneut definieren oder Speichergruppenpfade ändern, indem Sie den Befehl **RESTORE DATABASE** mit dem Parameter **REDIRECT** absetzen.

Die Umleitung von Tabellenbereichscontainern bietet große Flexibilität bei der Verwaltung von Tabellenbereichscontainern. Sie können die Speichergruppenkonfiguration einer Datenbank ändern, bevor Sie Datenseiten des Backup-Images wiederherstellen, ähnlich wie Sie die Pfade von Tabellenbereichscontainern umleiten können. Wenn Sie eine Speichergruppe nach dem Generieren des Backup-Images umbenannt haben, verweist der im Befehl **SET STOGROUP PATHS** angegebene Speichergruppenname auf den Speichergruppennamen im Backup-Image, nicht auf den aktuelleren Namen.

Ausführen einer umgeleiteten Restoreoperation in einer Umgebung mit partitionierten Datenbanken

In einer Umgebung mit partitionierten Datenbanken können Sie während eines umgeleiteten Restores für eine Datenbank nur die Speichergruppenpfade von der Katalogdatenbankpartition auf neue Speichergruppenpfade umleiten. Die Speichergruppenpfade aller anderen Datenbankpartitionen müssen mit der Katalogpartition synchronisiert sein.

Die Änderung eines beliebigen Speichergruppenpfads in der Katalogpartition versetzt alle Partitionen, die keine Katalogpartitionen sind, in den Status *Restore anstehend* (RESTORE_PENDING). Wenn Sie Speichergruppenpfade umleiten, müssen Sie die Katalogpartition vor allen anderen Datenbankpartition wiederherstellen. Nachdem Sie die Katalogdatenbankpartition wiederhergestellt haben, können Sie die Partitionen, die keine Katalogdatenbankpartitionen sind, ohne Umleitung von Speichergruppenpfaden parallel wiederherstellen. Die Nicht-Katalogdatenbankpartitionen übernehmen automatisch die neuen Speichergruppenpfade, die Sie für die Katalogdatenbankpartition angegeben haben. Neue Speichergruppenpfade werden auch automatisch übernommen, wenn die Speichergruppenpfade während eines Restores der Datenbank implizit geändert werden, wenn Sie eine andere Datenbank wiederherstellen (eine Datenbank mit einem anderen Namen, einer anderen Instanz oder einer anderen Nummer).

Wenn Sie die Speichergruppenpfade seit der Erstellung des letzten Backups geändert haben, können Sie dieses Backup-Image (mit anderen Speichergruppenpfaden) immer noch für einen Restore der Datenbankpartitionen verwenden. Dieser Restore wird nicht als umgeleiteter Restore betrachtet. Der Restore von diesem Backup-Image führt dazu, dass die Datenbankpartition vorübergehend die Speichergruppenpfade verwendet, die Sie zum Zeitpunkt der Backuperstellung definiert haben. Führen Sie eine aktualisierende Recovery aus, um die Änderungen an den Speichergruppenpfaden erneut anzuwenden und alle Datenbankpartitionen zu resynchronisieren.

Beispiele

Beispiel 1

Sie können einen umgeleiteten Restore der Tabellenbereichscontainer auf Datenbank SAMPLE mit dem Befehl **SET TABLESPACE CONTAINERS** zum Definieren von Tabellenbereichscontainern ausführen:

```
     db2 restore db sample redirect without prompting
     SQL1277W Zurzeit wird eine umgeleitete Restoreoperation ausgeführt.
Während eines Restores für einen Tabellenbereich können nur die Pfade
von den Tabellenbereichen, die wiederhergestellt werden, rekonfiguriert werden.
Während eines Restores für eine Datenbank, können die Speicherpfade
von Speichergruppen und DMS-Tabellenbereichscontainer rekonfiguriert werden.

DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.

     db2 set tablespace containers for 2 using (path 'userspace1.0', path
     'userspace1.1')
     DB20000I Der Befehl SET TABLESPACE CONTAINERS wurde erfolgreich ausgeführt.

     db2 restore db sample continue
     DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Beispiel 2

Sie können die Pfade der definierten Speichergruppe mithilfe des Befehls **SET STOGROUP PATHS** neu definieren:

```
RESTORE DB SAMPLE REDIRECT

SET STOGROUP PATHS FOR sg_hot ON '/ssd/fs1', '/ssd/fs2'
SET STOGROUP PATHS FOR sg_cold ON '/hdd/path1', '/hdd/path2'

RESTORE DB SAMPLE CONTINUE
```

Beispiel 3

Es folgt ein typisches Beispielszenario für den umgeleiteten, nicht inkrementellen Restore einer Datenbank mit dem Aliasnamen MEINEDB:

1. Setzen Sie einen Befehl RESTORE DATABASE mit der Option REDIRECT ab.

   ```
   db2 restore db meinedb replace existing redirect
   ```

2. Setzen Sie für jeden Tabellenbereich, dessen Container Sie erneut definieren möchten, einen Befehl SET TABLESPACE CONTAINERS ab. Beispiel für eine Windows-Umgebung:

   ```
   db2 set tablespace containers for 5 using
             (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
   ```

   Setzen Sie für jeden Tabellenbereich, dessen Containerpositionen erneut definiert werden, den Befehl LIST TABLESPACE CONTAINERS ab, um sicherzustellen, dass es sich bei den Containern der wiederhergestellten Datenbank um die in diesem Schritt angegebenen Container handelt.

3. Nach der erfolgreichen Durchführung der Schritte 1 und 2 setzen Sie den folgenden Befehl ab:

   ```
   db2 restore db mydb continue
   ```

   Dies ist der letzte Schritt des umgeleiteten Restores.

4. Falls Schritt 3 fehlschlägt oder die Restoreoperation abgebrochen wurde, kann der umgeleitete Restore erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen.

Anmerkung:

1. Nach der erfolgreichen Ausführung von Schritt 1 und vor der Vollendung von Schritt 3 kann die Restoreoperation durch Absetzen des folgenden Befehls abgebrochen werden:

   ```
   db2 restore db meinedb abort
   ```

2. Falls Schritt 3 fehlschlägt oder die Restoreoperation abgebrochen wurde, kann der umgeleitete Restore erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen.

Beispiel 4

Es folgt ein typisches Beispielszenario für den manuellen, umgeleiteten inkrementellen Restore einer Datenbank mit dem Aliasnamen MEINEDB und den folgenden Backup-Images:

```
   backup db meinedb
Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: <ts1>

   backup db meinedb incremental
Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: <ts2>
```

1. Setzen Sie einen Befehl RESTORE DATABASE mit den Optionen INCREMENTAL und REDIRECT ab.

   ```
   db2 restore db meinedb incremental taken at <zm2> replace existing redirect
   ```

2. Setzen Sie für jeden Tabellenbereich, dessen Container erneut definiert werden müssen, einen Befehl SET TABLESPACE CONTAINERS ab. Beispiel für eine Windows-Umgebung:

   ```
   db2 set tablespace containers for 5 using
             (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
   ```

   Setzen Sie den Befehl LIST TABLESPACE CONTAINERS ab, um sicherzustellen, dass es sich bei den Containern der wiederhergestellten Datenbank um die in diesem Schritt angegebenen Container handelt.

3. Nach der erfolgreichen Durchführung der Schritte 1 und 2 setzen Sie den folgenden Befehl ab:

```
db2 restore db mydb continue
```

4. Die übrigen Befehle zum inkrementellen Restore können nun wie folgt abgesetzt werden:

```
db2 restore db meinedb incremental taken at <zm1>
db2 restore db meinedb incremental taken at <ts2>
```

   Dies ist der letzte Schritt des umgeleiteten Restores.

Anmerkung:

1. Nach der erfolgreichen Ausführung von Schritt 1 und vor der Vollendung von Schritt 3 kann die Restoreoperation durch Absetzen des folgenden Befehls abgebrochen werden:

```
db2 restore db meinedb abort
```

2. Nach der erfolgreichen Ausführung von Schritt 3 und vor dem Absetzen aller erforderlichen Befehle in Schritt 4 kann die Restoreoperation durch Absetzen des folgenden Befehls abgebrochen werden:

```
db2 restore db meinedb incremental abort
```

3. Falls Schritt 3 fehlschlägt oder die Restoreoperation abgebrochen wurde, kann der umgeleitete Restore erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen.
4. Wenn der Befehl zum Restore in Schritt 4 fehlschlägt, kann der fehlgeschlagene Befehl erneut abgesetzt werden, um den Restoreprozess fortzusetzen.

Beispiel 5

Es folgt ein typisches Beispielszenario für den automatischen, umgeleiteten inkrementellen Restore derselben Datenbank:

1. Setzen Sie einen Befehl RESTORE DATABASE mit den Optionen INCREMENTAL AUTOMATIC und REDIRECT ab.

```
db2 restore db meinedb incremental automatic taken at <zm2>
              replace existing redirect
```

2. Setzen Sie für jeden Tabellenbereich, dessen Container erneut definiert werden müssen, einen Befehl SET TABLESPACE CONTAINERS ab. Beispiel für eine Windows-Umgebung:

```
db2 set tablespace containers for 5 using
          (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
```

   Setzen Sie den Befehl LIST TABLESPACE CONTAINERS ab, um sicherzustellen, dass es sich bei den Containern der wiederhergestellten Datenbank um die in diesem Schritt angegebenen Container handelt.

3. Nach der erfolgreichen Durchführung der Schritte 1 und 2 setzen Sie den folgenden Befehl ab:

```
db2 restore db mydb continue
```

   Dies ist der letzte Schritt des umgeleiteten Restores.

Anmerkung:

1. Nach der erfolgreichen Ausführung von Schritt 1 und vor der Vollendung von Schritt 3 kann die Restoreoperation durch Absetzen des folgenden Befehls abgebrochen werden:

```
db2 restore db meinedb abort
```

2. Falls Schritt 3 fehlschlägt oder die Restoreoperation abgebrochen wurde, kann der umgeleitete Restore erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen, nachdem Sie den folgenden Befehl abgesetzt haben:

   ```
   db2 restore db meinedb incremental abort
   ```

Erneutes Definieren von Tabellenbereichscontainern durch Restore einer Datenbank mithilfe eines automatisch generierten Scripts

Wenn Sie eine Datenbank wiederherstellen, nimmt das Restoredienstprogramm an, dass das physische Layout der Container mit dem der Datenbank zum Zeitpunkt des Backups identisch sein soll.

Wenn es erforderlich ist, die Position oder Größe eines der physischen Container zu ändern, müssen Sie den Befehl **RESTORE DATABASE** mit der Option **REDIRECT** absetzen. Diese Option verlangt, dass Sie die Positionen physischer Container, die im Backup-Image gespeichert sind, angeben und vollständige Informationen zur Gruppe von Containern für jeden Tabellenbereich mit nicht dynamischem Speicher bereitstellen, den Sie ändern wollen. Sie können die Containerinformationen zum Zeitpunkt des Backups erfassen. Dies kann jedoch etwas aufwendig sein.

Zur Vereinfachung eines umgeleiteten Restores ermöglicht das Restoredienstprogramm Ihnen die Generierung eines Scripts für den umgeleiteten Restore aus einem vorhandenen Backup-Image. Dazu setzen Sie den Befehl **RESTORE DATABASE** mit dem Parameter **REDIRECT** und dem Parameter **GENERATE SCRIPT** ab. Das Restoredienstprogramm untersucht das Backup-Image, extrahiert Containerinformationen aus dem Backup-Image und generiert ein CLP-Script, das sämtliche detaillierten Containerinformationen enthält. Anschließend können Sie sämtliche Pfade oder Containergrößen im Script modifizieren und das CLP-Script ausführen, um die Datenbank mit der neuen Gruppe von Containern erneut zu erstellen. Mithilfe des von Ihnen generierten Scripts können Sie eine Datenbank auch dann wiederherstellen, wenn Sie nur über ein Backup-Image verfügen und das Layout der Container nicht kennen. Das Script wird auf dem Client erstellt. Auf der Grundlage des Scripts können Sie festlegen, wo Speicherplatz für Protokolldateien und Container der wiederhergestellten Datenbank benötigt wird, und Sie können die Pfade für Protokolldateien und Container entsprechend ändern.

Das generierte Script besteht aus vier Abschnitten:

Initialisierung

Im ersten Abschnitt werden Befehlsoptionen festgelegt und die Datenbankpartitionen angegeben, in denen der Befehl auszuführen ist. Das folgende Beispiel zeigt, wie dieser erste Abschnitt aussehen kann:

```
UPDATE COMMAND OPTIONS USING S ON Z ON SAMPLE_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM  0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
```

Dabei gilt Folgendes:

- `S ON` gibt an, dass die Ausführung des Befehls gestoppt werden soll, wenn ein Befehlsfehler auftritt.
- `Z ON SAMPLE_NODE0000.out` gibt an, dass die Ausgabe in eine Datei mit dem Namen *aliasname-der-datenbank*`_NODE`*dbpartitionnum*`.out` erfolgen soll.
- `V ON` gibt an, dass der aktuelle Befehl über die Standardausgabe ausgegeben werden soll.

Wenn das Script in einer Umgebung mit partitionierten Datenbanken ausgeführt wird, ist es wichtig, die Datenbankpartition anzugeben, in der die Scriptbefehle auszuführen sind.

Befehl RESTORE DATABASE mit dem Parameter REDIRECT

Im zweiten Abschnitt wird der Befehl **RESTORE DATABASE** gestartet und der Parameter **REDIRECT** verwendet. In diesem Abschnitt können alle Parameter des Befehls **RESTORE DATABASE** angegeben werden mit Ausnahme derer, die in Kombination mit dem Parameter **REDIRECT** nicht verwendbar sind. Das folgende Beispiel zeigt, wie dieser zweite Abschnitt aussehen kann:

```
RESTORE DATABASE SAMPLE
-- USER 'benutzername'
-- USING 'kennwort'
   FROM '/home/jseifert/backups'
   TAKEN AT 20050906194027
-- DBPATH ON 'zielverzeichnis'
   INTO SAMPLE
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00001/LOGSTREAM0000/'
-- WITH pufferanzahl BUFFERS
-- BUFFER puffergröße
   -- REPLACE HISTORY FILE
   -- REPLACE EXISTING
REDIRECT
-- PARALLELISM n
   -- WITHOUT ROLLING FORWARD
   -- WITHOUT PROMPTING
;
```

Tabellenbereichsdefinitionen

Dieser Abschnitt enthält Tabellenbereichsdefinitionen für jeden Tabellenbereich, der im Backup-Image enthalten ist oder in der Befehlszeile angegeben wird. Für jeden Tabellenbereich ist ein Abschnitt vorhanden, der aus einem Kommentarblock mit Informationen über Name, Typ und Größe des Tabellenbereichs besteht. Die Informationen sind im gleichen Format wie eine Tabellenbereichsmomentaufnahme aufbereitet. Anhand der gegebenen Informationen können Sie die erforderliche Größe für die einzelnen Tabellenbereiche bestimmen. Wird die Ausgabe zu einem mit dynamischem Speicher erstellten Tabellenbereich angezeigt, ist die Klausel SET TABLESPACE CONTAINERS in der Anzeige nicht enthalten. Das folgende Beispiel zeigt einen Abschnitt einer Tabellenbereichsdefinition:

```
-- *****************************************************************
-- ** Tablespace name                         = SYSCATSPACE
-- **   Tablespace ID                         = 0
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Total number of pages                 = 5572
-- *****************************************************************
SET TABLESPACE CONTAINERS FOR 0
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0000.0'
);
-- *****************************************************************
-- ** Tablespace name                         = TEMPSPACE1
-- **   Tablespace ID                         = 1
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = System Temporary data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
```

```
-- **   Total number of pages                   = 0
-- ******************************************************************
SET TABLESPACE CONTAINERS FOR 1
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0001.0'
);
-- ******************************************************************
-- ** Tablespace name                         = DMS
-- **   Tablespace ID                         = 2
-- **   Tablespace Type                       = Database managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Auto-resize enabled                   = No
-- **   Total number of pages                 = 2000
-- **   Number of usable pages                = 1960
-- **   High water mark (pages)               = 96
-- ******************************************************************
SET TABLESPACE CONTAINERS FOR 2
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   '/tmp/dms1'                                          1000
, FILE   '/tmp/dms2'                                          1000
);
```

Befehl RESTORE DATABASE mit dem Parameter CONTINUE

Im letzten Abschnitt wird der Befehl **RESTORE DATABASE** mit dem Parameter **CONTINUE** abgesetzt, um den umgeleiteten Restore abzuschließen. Das folgende Beispiel zeigt, wie dieser letzte Abschnitt aussehen kann:

```
RESTORE DATABASE SAMPLE CONTINUE;
```

Ausführen eines umgeleiteten Restores mithilfe eines automatisch generierten Scripts

Wenn Sie eine umgeleitete Restoreoperation durchführen, müssen Sie die Positionen physischer Container angeben, die im Backup-Image gespeichert sind, und vollständige Informationen zur Gruppe von Containern für jeden Tabellenbereich bereitstellen, den Sie ändern wollen.

Vorbereitende Schritte

Sie können einen umgeleiteten Restore nur dann ausführen, wenn die Datenbank zuvor mit dem DB2-Backup-Dienstprogramm gesichert wurde.

Informationen zu diesem Vorgang

- Wenn die Datenbank vorhanden ist, müssen Sie eine Verbindung zu ihr herstellen können, um das Script zu generieren. Daher gilt: Wenn für die Datenbank ein Upgrade oder eine Recovery nach Systemabsturz erforderlich ist, muss dieses Upgrade bzw. diese Recovery ausgeführt werden, bevor Sie versuchen, ein Script für den umgeleiteten Restore zu generieren.
- Wenn Sie in einer Umgebung mit partitionierten Datenbanken arbeiten und die Zieldatenbank nicht vorhanden ist, können Sie den Befehl zur Generierung des Scripts für den umgeleiteten Restore nicht gleichzeitig in allen Datenbankpartitionen ausführen. Stattdessen muss der Befehl zur Generierung des Scripts für den umgeleiteten Restore jeweils nur in einer Datenbankpartition gleichzeitig, beginnend mit der Katalogpartition, ausgeführt werden.

Alternativ können Sie zuerst eine Pseudodatenbank mit demselben Namen wie Ihre Zieldatenbank erstellen. Nach der Erstellung der Pseudodatenbank können Sie das Script für den umgeleiteten Restore in allen Datenbankpartitionen gleichzeitig generieren.

- Auch wenn Sie den Parameter **REPLACE EXISTING** bei der Ausführung des Befehls **RESTORE DATABASE** zur Generierung des Scripts angeben, wird der Parameter **REPLACE EXISTING** in dem Script auf Kommentar gesetzt.
- Aus Sicherheitsgründen wird Ihr Kennwort nicht in das generierte Script eingetragen. Sie müssen das Kennwort manuell eingeben.
- Das Script für den Restore umfasst die Speichergruppenzuordnungen für jeden Tabellenbereich, den Sie wiederherstellen.

Vorgehensweise

Gehen Sie zur Durchführung eines umgeleiteten Restores mithilfe eines Scripts wie folgt vor:

1. Generieren Sie mithilfe des Restoredienstprogramms ein Script für den umgeleiteten Restore. Das Restoredienstprogramm kann über den Befehlszeilenprozessor (CLP) oder die Anwendungsprogrammierschnittstelle db2Restore aufgerufen werden. Das folgende Beispiel zeigt einen Befehl **RESTORE DATABASE** mit dem Parameter **REDIRECT** und dem Parameter **GENERATE SCRIPT**:

   ```
   db2 restore db test from /home/jseifert/backups taken at 20050304090733
                  redirect generate script test_node0000.clp
   ```

   Dieser Befehl erstellt ein Script für den umgeleiteten Restore auf dem Client mit dem Namen `test_node0000.clp`.
2. Öffnen Sie das Script für den umgeleiteten Restore in einem Texteditor, um alle erforderlichen Modifikationen vorzunehmen. Folgende Angaben können modifiziert werden:
   - RESTORE-Optionen
   - Pfade für dynamischen Speicher
   - Layout und Pfade von Containern
3. Führen Sie das modifizierte Script für den umgeleiteten Restore aus. Beispiel:

   ```
   db2 -tvf test_node0000.clp
   ```

Klonen einer Produktionsdatenbank mithilfe verschiedener Speichergruppenpfade

Eventuell müssen Sie eine Produktionsdatenbank in eine Testdatenbank klonen, die sich auf einer anderen Maschine befindet. Die Testmaschine und der Produktionsserver weisen wahrscheinlich unterschiedliche Speichergruppenpfade auf. Auf dem Testsystems gibt es möglicherweise keine Pfade, die durch die neuesten und schnellsten Speicherplatten gestützt werden.

Informationen zu diesem Vorgang

Stellen Sie sich vor, in einer Produktionsdatenbank `proddb` befinden sich einige Daten in der Speichergruppe `sg_hot` und einige Pfade dieser Speichergruppe auf einer SSD-Einheit. Sie wollen einen Restore der Daten in die kostengünstigere und langsamere Testdatenbank `testdb` durchführen. Das Testsystem verfügt nicht über die SSD-Einheit, aber die anderen Pfade entsprechen sich. Durch Ausführen eines umgeleiteten Restores können die Pfade für `sg_hot` auf dem Testsystem geändert werden, ohne die anderen Speichergruppen zu ändern.

Vorgehensweise

Gehen Sie wie folgt vor, um einen Restore von Daten einer Produktionsdatenbank in eine Testdatenbank durchzuführen:

1. Führen Sie ein Backup der Produktionsdatenbank aus. Setzen Sie den folgenden Befehl ab:

   ```
   BACKUP DATABASE produktions_db TO /backup
   ```

   Dabei ist *produktions_db* die Produktionsdatenbank.
2. Konfigurieren Sie einen umgeleiteten Restore in die Testdatenbank. Setzen Sie den folgenden Befehl ab:

   ```
   RESTORE DATABASE
   testdb REDIRECT
   ```

   Dabei ist *testdb* die Testdatenbank.
3. Ändern Sie die Speicherpfade für sg_hot, da keine Speichergruppe 'sg_hot' in der Testdatenbank zur Verfügung steht. Setzen Sie den folgenden Befehl ab:

   ```
   SET STOGROUP PATHS
   FOR sg_hot ON '/hdd/pfad1', '/hdd/pfad2'
   ```

   Dabei ist *sg_hot* die Speichergruppe sg_hot.
4. Setzen Sie den Restore der Testdatenbank fort. Setzen Sie den folgenden Befehl ab:

   ```
   RESTORE DATABASE testdb CONTINUE
   ```
5. Passen Sie den Namen der Speichergruppe an die neuen Pfade an. Setzen Sie die folgenden Befehle ab:

   ```
   CONNECT TO testdb
   RENAME STOGROUP sg_hot TO sg_cold
   ```

Erneutes Erstellen einer Datenbank

Unter der erneuten Erstellung (Rebuild) einer Datenbank ist der Prozess der Wiederherstellung einer Datenbank bzw. einer Teilgruppe ihrer Tabellenbereiche durch eine Reihe von Restoreoperationen zu verstehen. Die zur erneuten Datenbankerstellung bereitgestellte REBUILD-Funktionalität macht DB2-Datenbankprodukte zu leistungsfähigeren und vielseitigeren Produkten, die Ihnen eine komplettere Recoverylösung bieten.

Die Fähigkeit, eine Datenbank aus Backup-Images von Tabellenbereichen erneut erstellen zu können, bedeutet, dass Sie Datenbankgesamtbackups nicht mehr so häufig zu erstellen brauchen. Bei zunehmenden Datenbankgrößen finden sich immer seltener geeignete Gelegenheiten zur Erstellung von Datenbankgesamtbackups. Die alternative Verwendung von Tabellenbereichsbackups bietet die Möglichkeit, mit weniger häufigen Datenbankgesamtbackups zu arbeiten. Stattdessen können Sie häufiger Tabellenbereichsbackups erstellen und diese zusammen mit Protokolldateien in Ihrem Plan für den Einsatz bei einem Ausfall vorsehen.

Wenn Sie im Recoveryfall eine Teilgruppe von Tabellenbereichen schneller als andere wieder online verfügbar machen müssen, können Sie die REBUILD-Funktionalität zu diesem Zweck nutzen. Die Möglichkeit, nur eine Teilgruppe von Tabellenbereichen online verfügbar zu machen, ist insbesondere in Test- und Produktionsumgebungen nützlich.

Die erneute Erstellung einer Datenbank beinhaltet eine potenziell größere Anzahl von Restoreoperationen. Eine REBUILD-Operation kann ein Datenbankimage oder Tabellenbereichsimage (oder beides) verwenden. Sie kann Gesamtbackups oder inkrementelle Backups (oder beides) verwenden. Die einleitende Restoreoperation stellt das Zielimage wieder her, das die Struktur der Datenbank definiert, die wiederhergestellt werden kann (z. B. die Tabellenbereichsgruppen, die Speichergruppen und die Datenbankkonfiguration. Bei wiederherstellbaren Datenbanken ermöglicht die REBUILD-Operation die Erstellung einer Datenbank, zu der eine Verbindung herstellbar ist und die eine Teilgruppe der Tabellenbereiche enthält, die Sie online benötigen, während Tabellenbereiche, die später offline wiederhergestellt werden können, zunächst unverändert beibehalten werden.

Die Methode, die Sie zur erneuten Erstellung Ihrer Datenbank verwenden, hängt davon ab, ob sie wiederherstellbar oder nicht wiederherstellbar ist.

- Wenn die Datenbank wiederherstellbar ist, verwenden Sie eine der folgenden Methoden:
  - Führen Sie mithilfe eines Images eines Gesamtbackups oder eines inkrementellen Backup-Images einer Datenbank oder von Tabellenbereichen als Ziel die REBUILD-Operation für Ihre Datenbank aus, indem Sie den Tabellenbereich SYSCATSPACE und alle anderen Tabellenbereiche aus dem Zielimage nur mit der Option **REBUILD** im Befehl RESTORE wiederherstellen. Anschließend können Sie für Ihre Datenbank eine aktualisierende Recovery (ROLLFORWARD) bis zu einem bestimmten Zeitpunkt ausführen.
  - Führen Sie mithilfe eines Images eines Gesamtbackups oder eines inkrementellen Backup-Images einer Datenbank oder von Tabellenbereichen als Ziel die REBUILD-Operation für Ihre Datenbank aus, indem Sie in der Option **REBUILD** die Gruppe von wiederherzustellenden Tabellenbereichen angeben, die in der Datenbank zum Zeitpunkt des Zielimages definiert waren. Der Tabellenbereich SYSCATSPACE muss zu dieser Gruppe gehören. Durch diese Operation werden die Tabellenbereiche wiederhergestellt, die im Zielimage definiert sind. Anschließend werden mithilfe der Datei des Recoveryprotokolls alle anderen erforderlichen Backup-Images für die verbleibenden Tabellenbereiche, die sich nicht im Zielimage befinden, gesucht und wiederhergestellt. Wenn diese Restoreoperationen abgeschlossen sind, führen Sie eine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt aus.
- Wenn die Datenbank nicht wiederherstellbar ist:
  - Führen Sie mithilfe eines Images eines Gesamtbackups oder eines inkrementellen Backup-Images einer Datenbank als Ziel die REBUILD-Operation für Ihre Datenbank aus, indem Sie den Tabellenbereich SYSCATSPACE und alle anderen Tabellenbereiche aus dem Zielimage mit der entsprechenden **REBUILD**-Syntax im Befehl RESTORE wiederherstellen. Wenn die Restoreoperation abgeschlossen ist, können Sie eine Verbindung zur Datenbank herstellen.

Angeben des Zielimages

Die Ausführung einer erneuten Erstellung einer Datenbank beginnen Sie, indem Sie den Befehl **RESTORE** unter Angabe des aktuellsten Backup-Images absetzen, das Sie als Ziel der Restoreoperation verwenden. Dieses Image wird als Zielimage der REBUILD-Operation bezeichnet, weil es die Struktur der wiederherzustellenden Datenbank einschließlich der Tabellenbereiche, die wiederherstellbar sind, der Datenbankkonfiguration und der Protokollfolge definiert. Das Zielimage für die Rebuildoperation wird mit dem Parameter **TAKEN AT** im Befehl **RESTORE DATABASE** angegeben. Bei dem Zielimage kann es sich um jeden Typ von Backup (Gesamtbackup, Tabellenbereichsbackup, inkrementelles, Online- oder Offline-Back-

up) handeln. Zur erneuten Erstellung der Datenbank sind alle die Tabellenbereiche verfügbar, die in der Datenbank zum Zeitpunkt der Erstellung des Zielimages definiert waren.

Sie müssen die Tabellenbereiche, die wiederhergestellt werden sollen, mit einer der folgenden Methoden angeben:

- Geben Sie an, dass alle in der Datenbank definierten Tabellenbereiche wiederherzustellen sind, und geben Sie eine Ausnahmeliste an, falls Tabellenbereiche ausgeschlossen werden sollen.
- Geben Sie an, dass alle Tabellenbereiche wiederherzustellen sind, die Benutzerdaten im Zielimage enthalten, und geben Sie eine Ausnahmeliste an, falls Tabellenbereiche ausgeschlossen werden sollen.
- Geben Sie die Liste der in der Datenbank definierten Tabellenbereiche an, die wiederhergestellt werden sollen.

Wenn Sie die Tabellenbereiche kennen, welche die erneut erstellte Datenbank enthalten soll, führen Sie den Befehl **RESTORE** mit der entsprechenden **REBUILD**-Option aus, indem Sie das zu verwendende Zielimage angeben.

Rebuildphase

Wenn der Befehl **RESTORE** mit der entsprechenden **REBUILD**-Option abgesetzt und das Zielimage erfolgreich wiederhergestellt wurde, wird die Datenbank als in der Rebuildphase befindlich betrachtet. Nach der Restoreoperation des Zielimages stellen alle weiteren Restoreoperationen für Tabellenbereiche Daten in vorhandenen Tabellenbereichen, so wie sie in der erneut erstellten Datenbank definiert sind, wieder her. Anschließend werden diese Tabellenbereiche bei Abschluss der Rebuildoperation zusammen mit der Datenbank aktualisierend wiederhergestellt (ROLLFORWARD).

Wenn Sie den Befehl **RESTORE** mit der entsprechenden **REBUILD**-Option absetzen und die Datenbank nicht vorhanden ist, wird eine neue Datenbank auf der Basis der Attribute im Zielimage erstellt. Ist die Datenbank vorhanden, empfangen Sie eine Warnung, die besagt, dass die Phase der Neuerstellung (Rebuildphase) gestartet wird. Sie werden zur Angabe aufgefordert, ob die Rebuildoperation fortgesetzt werden soll oder nicht.

Die Rebuildoperation stellt alle Anfangsmetadaten aus dem Zielimage wieder her. Dies schließt auch alle Daten mit ein, die zur Datenbank gehören, jedoch nicht zu den Tabellenbereichsdaten oder Protokolldateien. Solche Anfangsmetadaten sind zum Beispiel:

- Tabellenbereichsdefinitionen
- Die Verlaufsdatei, bei der es sich um eine Datenbankdatei handelt, in der Verwaltungsoperationen aufgezeichnet werden

Die Rebuildoperation stellt außerdem die Datenbankkonfiguration wieder her. Das Zielimage legt die Protokollkette fest, die bestimmt, welche Images für die verbleibenden Restores in der Rebuildphase verwendet werden können. Nur Images in derselben Protokollkette sind verwendbar.

Wenn eine Datenbank bereits auf Platte vorhanden ist und die Verlaufsdatei aus dem Zielimage genommen werden soll, müssen Sie die Option **REPLACE HISTORY FILE** angeben. Die zu diesem Zeitpunkt auf Platte befindliche Verlaufsdatei wird durch die automatische Logik verwendet, um die verbleibenden Images zu suchen, die zur erneuten Erstellung der Datenbank erforderlich sind.

Nachdem das Zielimage wiederhergestellt wurde, geschieht Folgendes:

- Wenn die Datenbank wiederherstellbar ist, wird sie in den Status *aktualisierende Recovery anstehend* versetzt und alle Tabellenbereiche, die Sie mit RESTORE wiederherstellen, werden ebenfalls in den Status *aktualisierende Recovery anstehend* versetzt. Alle Tabellenbereiche, die in der Datenbank definiert sind, jedoch nicht wiederhergestellt wurden, werden in den Status *Restore anstehend* versetzt.
- Wenn die Datenbank nicht wiederherstellbar ist, werden die Datenbank und die Tabellenbereiche, die wiederhergestellt wurden, in den Normalstatus versetzt. Alle Tabellenbereiche, die nicht wiederhergestellt wurden, werden in den Status *Löschen anstehend* versetzt, da sie nicht mehr zurückgeschrieben werden können. Für diesen Typ von Datenbank ist damit die Rebuildphase abgeschlossen.

Für wiederherstellbare Datenbanken endet die Rebuildphase, wenn der erste Befehl **ROLLFORWARD DATABASE** abgesetzt wird und das Dienstprogramm zur aktualisierenden Recovery mit der Verarbeitung von Protokollsätzen beginnt. Wenn eine aktualisierende Recoveryoperation nach dem Start der Verarbeitung von Protokollsätzen fehlschlägt und als Nächstes eine Restoreoperation abgesetzt wird, wird die Restoreoperation nicht als Teil der Rebuildphase interpretiert. Solche Restoreoperationen sollten als normale Tabellenbereichswiederherstellungen betrachtet werden, die nicht zur Rebuildphase gehören.

Automatische Verarbeitung

Nach dem Restore des Zielimages stellt das Restoredienstprogramm fest, ob verbleibende Tabellenbereiche vorhanden sind, die wiederherzustellen sind. Wenn dies der Fall ist, werden sie über dieselbe Verbindung wiederhergestellt, die zur Ausführung des Befehls **RESTORE DATABASE** mit der Option **REBUILD** verwendet wurde. Das Dienstprogramm verwendet die Verlaufsdatei auf Platte, um die aktuellsten Backup-Images zu finden, die vor dem Zielimage erstellt wurden, das alle verbleibenden wiederherzustellenden Tabellenbereiche enthält. Das Restoredienstprogramm verwendet die Positionsdaten der Backup-Images, die in der Verlaufsdatei gespeichert sind, um jedes einzelne dieser Images automatisch wiederherzustellen. Diese nachfolgenden Restoreoperationen, die auf Tabellenbereichsebene erfolgen, können nur offline ausgeführt werden. Wenn das ausgewählte Image nicht zur aktuellen Protokollkette gehört, wird ein Fehler zurückgegeben. Jeder Tabellenbereich, der aus diesem Image wiederhergestellt wird, wird in den Status *aktualisierende Recovery anstehend* versetzt.

Das Restoredienstprogramm versucht, alle erforderlichen Tabellenbereiche automatisch wiederherzustellen. In einigen Fällen ist es jedoch aufgrund von Problemen mit der Verlaufsdatei nicht in der Lage, einige Tabellenbereiche wiederherzustellen, oder es tritt ein Fehler beim Restore eines der erforderlichen Images auf. In einem solchen Fall können Sie die Rebuildoperation entweder manuell zu Ende führen oder das Problem beheben und die Rebuildoperation erneut ausführen.

Wenn die automatische erneute Erstellung nicht erfolgreich abgeschlossen werden kann, schreibt das Restoredienstprogramm alle Informationen in das Diagnoseprotokoll (Protokolldatei **db2diag**), die es für die verbleibenden Restoreschritte gesammelt hat. Mithilfe dieser Informationen können Sie die erneute Erstellung manuell zu Ende führen.

Wenn eine Datenbank erneut erstellt wird, werden nur Container, die zu den Tabellenbereichen gehören, die Teil des Rebuildprozesses sind, angefordert.

Wenn Container durch einen umgeleiteten Restore umdefiniert werden müssen, müssen Sie den neuen Pfad und die Größe des neuen Containers für die verbleibenden Restoreoperationen und die nachfolgende aktualisierende Recoveryoperation festlegen.

Wenn die Daten für einen Tabellenbereich, der aus einem dieser verbleibenden Images wiederhergestellt wird, nicht in die Definitionen des neuen Containers passt, wird der Tabellenbereich in den Status *Restore anstehend* versetzt und am Ende der Restoreoperation eine Warnung ausgegeben. Weitere Informationen zu dem aufgetretenen Problem finden Sie im Diagnoseprotokoll.

Abschließen der Rebuildphase

Wenn alle vorgesehenen Tabellenbereiche wiederhergestellt wurden, haben je nach Konfiguration der Datenbank zwei Optionen. Wenn die Datenbank nicht wiederherstellbar ist, kann eine Verbindung zu dieser Datenbank hergestellt werden und alle wiederhergestellten Tabellenbereiche werden online verfügbar. Alle Tabellenbereiche, die sich im Status Löschen anstehend befinden, können nicht mehr wiederhergestellt werden und sollten gelöscht werden, wenn geplant ist, in Zukunft Backups für die Datenbank auszuführen.

Wenn die Datenbank wiederherstellbar ist, können Sie den Befehl ROLLFORWARD ausführen, um die Tabellenbereiche, die wiederhergestellt wurden, online verfügbar zu machen. Wenn der Tabellenbereich SYSCATSPACE nicht wiederhergestellt wurde, schlägt die aktualisierende Recovery fehl, sodass dieser Tabellenbereich wiederhergestellt werden muss, bevor die aktualisierende Recoveryoperation gestartet werden kann. Dies bedeutet, dass der Tabellenbereich SYSCATSPACE während der Rebuildphase wiederhergestellt werden muss.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken ist der Tabellenbereich SYSCATSPACE in Nichtkatalogpartitionen nicht vorhanden, sodass er dort nicht erneut erstellt werden kann. Allerdings muss der Tabellenbereich SYSCATSPACE in der Katalogpartition einer der Tabellenbereiche sein, die erneut erstellt werden, da ansonsten die aktualisierende Recoveryoperation nicht erfolgreich ausgeführt werden kann.

Die aktualisierende Recovery der Datenbank nimmt die Datenbank aus dem Status *aktualisierende Recovery anstehend* heraus und stellt alle Tabellenbereiche im Status *aktualisierende Recovery anstehend* aktualisierend wieder her. Das Dienstprogramm für aktualisierende Recovery (Rollforward) arbeitet nicht mit Tabellenbereichen im Status *Restore anstehend*.

Die Stoppzeit der aktualisierenden Recoveryoperation muss ein Zeitpunkt sein, der nach dem Ende des aktuellsten Backup-Images liegt, das in der Rebuildphase wiederhergestellt wird. Wenn ein anderer Zeitpunkt angegeben wird, tritt ein Fehler auf. Wenn die aktualisierende Recoveryoperation die Backupzeit des ältesten Images, das wiederhergestellt wurde, nicht erreichen kann, ist das Dienstprogramm für aktualisierende Recovery nicht in der Lage, die Datenbank bis zu einem konsistenten Punkt wiederherzustellen, sodass die Operation fehlschlägt.

Sie müssen sämtliche Protokolldateien für den Zeitraum zwischen dem frühesten und dem aktuellsten Backup-Image für die Verwendung durch das Dienstprogramm für aktualisierende Recovery zur Verfügung haben. Die erforderlichen Protokolle sind die Protokolle, die auf die Protokollkette aus dem frühesten Backup-Image bis zu dem Ziel-Backup-Image folgen, wie sie durch den Abschneidebereich im Zielimage definiert sind. Ansonsten schlägt die aktualisierende Recoveryoperati-

on fehl. Wenn während der Rebuildphase irgendwelche Backup-Images wiederhergestellt wurden, die aktueller waren als das Zielimage, sind die zusätzlichen Protokolle vom Zielimage bis zum aktuellsten wiederhergestellten Backup-Image erforderlich. Wenn die Protokolle nicht verfügbar gemacht werden, werden die durch die Protokolle nicht erreichten Tabellenbereiche von der aktualisierenden Recoveryoperation in den Status *Restore anstehend* versetzt. Sie können den Befehl **LIST HISTORY** ausführen, um den RESTORE-REBUILD-Eintrag mit dem Protokollbereich anzuzeigen, der für die aktualisierende Recovery erforderlich ist.

Die korrekten Protokolldateien müssen verfügbar sein. Wenn das Dienstprogramm für aktualisierende Recovery die Protokolle abrufen soll, müssen Sie sicherstellen, dass der DB2-Protokollmanager so konfiguriert ist, dass er die Position angibt, von der die Protokolldateien abgerufen werden können. Wenn der Protokollpfad oder der Archivierungspfad geändert wurden, müssen Sie die Option **OVERFLOW LOG PATH** des Befehls **ROLLFORWARD DATABASE** verwenden.

Verwenden Sie die Option **AND STOP** des Befehls **ROLLFORWARD DATABASE**, um die Datenbank verfügbar zu machen, wenn der Befehl ROLLFORWARD erfolgreich beendet wird. An diesem Punkt befindet sich die Datenbank nicht mehr im Status *aktualisierende Recovery anstehend*. Wenn die aktualisierende Recoveryoperation beginnt, jedoch vor dem erfolgreichen Abschluss ein Fehler auftritt, stoppt die aktualisierende Recoveryoperation am Fehlerpunkt und gibt eine Fehlernachricht zurück. Die Datenbank verbleibt im Status *aktualisierende Recovery anstehend*. Sie müssen Schritte zur Behebung des Problems (z. B. Korrektur der Protokolldatei) ausführen und einen weiteren ROLLFORWARD-Befehl absetzen, um die Verarbeitung fortzusetzen.

Wenn sich der Fehler nicht beheben lässt, können Sie die Datenbank am Fehlerpunkt verfügbar machen, indem Sie den Befehl **ROLLFORWARD STOP** ausführen. Nach Verwendung der Option **STOP** sind alle eventuell über diesen Punkt hinaus vorhandenen Protokolldaten nicht mehr verfügbar. Die Datenbank wird an diesem Punkt wieder verfügbar, und alle Tabellenbereiche, die wiederhergestellt wurden, werden online zugänglich. Tabellenbereiche, die noch nicht wiederhergestellt wurden, befinden sich im Status *Restore anstehend*. Die Datenbank selbst befindet sich im Normalstatus.

Sie müssen entscheiden, welches die beste Methode zur Wiederherstellung der im Status *Restore anstehend* verbleibenden Tabellenbereiche ist. Dies könnte eine neue Restore- und aktualisierende Recoveryoperation für einen Tabellenbereich oder auch eine Wiederholung der gesamten Operation zur erneuten Erstellung sein. Diese Entscheidung hängt vom Typ der aufgetretenen Probleme ab. In Fällen, in denen der Tabellenbereich SYSCATSPACE zu den Tabellenbereichen im Status *Restore anstehend* gehört, kann keine Verbindung zur Datenbank hergestellt werden.

Rebuild von Datenbanken und Tabellenbereichscontainer

Beim Rebuild von Datenbanken werden nur die Container angefordert, deren Tabellenbereiche Teil des Rebuildprozesses sind. Die Container, die zu den einzelnen Tabellenbereichen gehören, werden angefordert, wenn die Benutzerdaten der Tabellenbereiche aus einem Image wiederhergestellt werden.

Nach dem Restore des Zielimages sind nur die Definitionen für jeden Tabellenbereich, der der Datenbank zum Zeitpunkt des Backups bekannt war, wiederhergestellt. Dies bedeutet, dass der Datenbank, die durch den Rebuildprozess erstellt wird, die gleichen Tabellenbereiche bekannt sind, die sie zum Zeitpunkt des Back-

ups hatte. Für diejenigen Tabellenbereiche, für die auch die Benutzerdaten aus dem Zielimage wiederhergestellt werden sollten, wurden zu diesem Zeitpunkt auch die zugehörigen Container angefordert.

Für alle verbleibenden Tabellenbereiche, die durch direkte Restoreoperationen für diese Tabellenbereiche wiederhergestellt werden, werden die zugehörigen Container angefordert, wenn das Image wiederhergestellt wird, das die Daten der Tabellenbereiche enthält.

Erneutes Erstellen mit umgeleitetem Restore

Im Fall eines umgeleiteten Restores müssen alle Tabellenbereichscontainer beim Restore des Zielimages definiert werden. Wenn Sie die Option **REDIRECT** angeben, wird die Steuerung an Sie zurückgegeben, damit Sie Ihre Tabellenbereichscontainer neu definieren können. Wenn Sie Tabellenbereichscontainer mithilfe des Befehls **SET TABLESPACE CONTAINERS** erneut definiert haben, werden diese neuen Container zu diesem Zeitpunkt angefordert. Jeder Tabellenbereichscontainer, der von Ihnen nicht neu definiert wird, wird normal angefordert, wenn die Benutzerdaten des Tabellenbereichs aus einem Image wiederhergestellt werden.

Wenn die Daten für einen Tabellenbereich, der wiederhergestellt wird, nicht in die neuen Containerdefinitionen passen, wird der Tabellenbereich in den Status *Restore anstehend* versetzt und am Ende der Restoreoperation eine Warnung an Sie zurückgegeben. Im DB2-Diagnoseprotokoll finden Sie in diesem Fall eine Nachricht mit Zusatzinformationen zu dem Problem.

Rebuild von Datenbanken - Tabellenbereiche für temporäre Tabellen

Das Speichern von Tabellenbereichen für temporäre Tabellen unterscheidet sich vom Speichern der anderen Datenbankkomponenten in einem Backup-Image. Da Tabellenbereiche für temporäre Tabellen auf andere Weise gespeichert werden, werden sie beim Restore einer Datenbank auch auf andere Weise wiederhergestellt.

Im Allgemeinen besteht ein DB2-Backup-Image aus den folgenden Komponenten:

- Anfangsmetadaten für die Datenbank, wie zum Beispiel Tabellenbereichsdefinitionen, die Datenbankkonfigurationsdatei und die Verlaufsdatei.
- Daten für nicht temporäre Tabellenbereiche, die dem Dienstprogramm **BACKUP** angegeben wurden
- Endmetadaten für die Datenbank, wie zum Beispiel der Protokolldateiheader
- Protokolldateien (wenn die Option **INCLUDE LOGS** angegeben wurde)

In jedem Backup-Image, unabhängig davon, ob es sich um ein Datenbank- oder Tabellenbereichsbackup, ein Gesamtbackup oder ein inkrementelles (Delta-)Backup handelt, sind in jedem Fall diese Kernkomponenten enthalten.

Ein Backup-Image einer Datenbank enthält alle der zuvor aufgeführten Komponenten sowie Daten für alle in der Datenbank zum Zeitpunkt des Backups definierten Tabellenbereiche.

Ein Backup-Image von Tabellenbereichen enthält stets die zuvor aufgeführten Datenbankmetadaten, darüber hinaus jedoch nur die Daten der Tabellenbereiche, die bei der Ausführung des Backup-Dienstprogramms angegeben wurden.

Tabellenbereiche für temporäre Tabellen werden anders als Tabellenbereiche für nicht temporäre Tabellen behandelt. Die Daten in Tabellenbereichen für temporäre Tabellen werden in keinem Fall durch ein Backup gesichert, jedoch spielt ihr Vorhandensein für die Rahmendefinition der Datenbank eine wichtige Rolle. Obwohl die Daten solcher Tabellenbereiche selbst nie gesichert werden, werden die Tabellenbereiche für temporäre Tabellen als Teil der Datenbank betrachtet. Daher werden sie in den Metadaten, die zusammen mit einem Backup-Image gespeichert werden, speziell markiert. Dadurch sieht es so aus, als ob sie sich im Backup-Image befänden. Darüber hinaus enthalten die Tabellenbereichsdefinitionen Informationen zum Vorhandensein aller Tabellenbereiche für temporäre Tabellen.

Obwohl kein Backup-Image Daten eines Tabellenbereichs für temporäre Tabellen enthält, werden bei der erneuten Erstellung (Rebuild), wenn das Zielimage wieder hergestellt wird, unabhängig vom Typ des Backup-Image Tabellenbereiche für temporäre Tabellen nur insofern wiederhergestellt, als dass ihre Container angefordert und zugeordnet werden. Die Anforderung und Zuordnung von Containern erfolgt automatisch im Rahmen der Rebuildverarbeitung. Infolgedessen können beim erneuten Erstellen einer Datenbank Tabellenbereiche für temporäre Tabellen nicht ausgeschlossen werden.

Auswählen eines Zielimages für die erneute Erstellung einer Datenbank

Das Zielimage für die erneute Erstellung sollte das aktuellste Backup-Image sein, das Sie als Ausgangspunkt für Ihre Restoreoperation verwenden möchten.

Dieses Image wird als Zielimage der Rebuild-Operation bezeichnet, weil es die Struktur der wiederherzustellenden Datenbank einschließlich der Tabellenbereiche, die wiederherstellbar sind, der Datenbankkonfiguration und der Protokollfolge definiert. Es kann sich hierbei um jeden Typ von Backup (Gesamtbackup, Tabellenbereichsbackup, inkrementelles, Online- oder Offline-Backup) handeln.

Das Zielimage legt die Protokollfolge (oder Protokollkette) fest, die bestimmt, welche Images für die verbleibenden Restores in der Rebuildphase verwendet werden können. Nur Images in derselben Protokollkette sind verwendbar.

Die folgenden Beispiele veranschaulichen, wie das Image ausgewählt wird, das als Zielimage für die Rebuildoperation zu verwenden ist.

Nehmen Sie an, eine Datenbank mit dem Namen SAMPLE enthält die folgenden Tabellenbereiche:

- SYSCATSPACE (Systemkataloge)
- USERSP1 (Tabellenbereich für Benutzerdaten)
- USERSP2 (Tabellenbereich für Benutzerdaten)
- USERSP3 (Tabellenbereich für Benutzerdaten)

Abb. 24 auf Seite 426 zeigt die folgenden Backups auf Datenbankebene und Backups auf Tabellenbereichsebene, die erstellt wurden, in chronologischer Reihenfolge:

1. Datenbankgesamtbackup DB1
2. Tabellenbereichsgesamtbackup TS1
3. Tabellenbereichsgesamtbackup TS2
4. Tabellenbereichsgesamtbackup TS3
5. Restore der Datenbank und aktualisierende Recovery bis zu einem Zeitpunkt zwischen TS1 und TS2

6. Tabellenbereichsgesamtbackup TS4
7. Tabellenbereichsgesamtbackup TS5

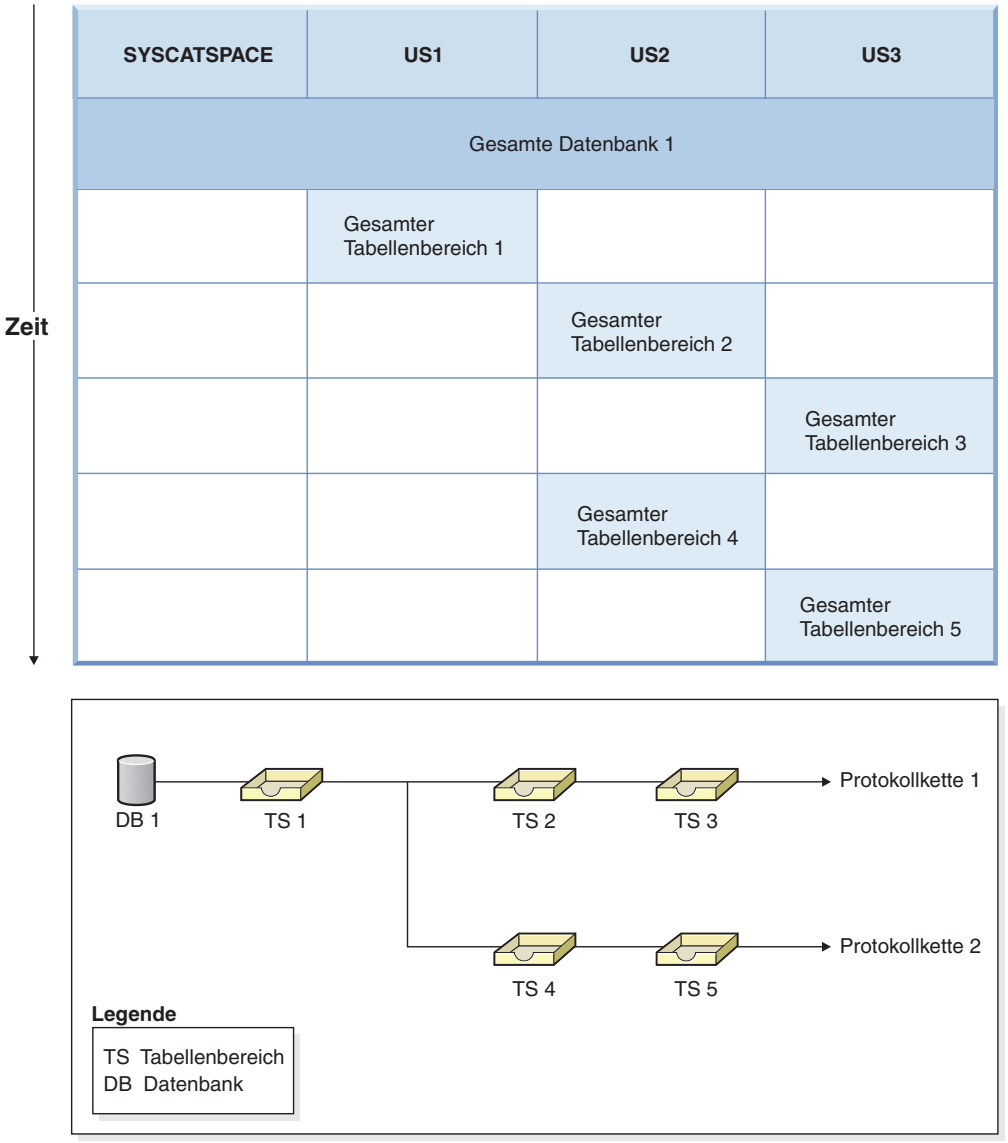


Abbildung 24. Backups auf Datenbank- und Tabellenbereichsebene der Datenbank SAMPLE

Beispiel 1

Das folgende Beispiel veranschaulicht die Befehle des Befehlszeilenprozessors (CLP), die Sie ausführen müssen, um die Datenbank SAMPLE bis zum aktuellen Zeitpunkt erneut zu erstellen. Als Erstes müssen Sie die Tabellenbereiche auswählen, die Sie erneut erstellen wollen. Da Sie die Datenbank bis zum aktuellen Zeitpunkt erneut erstellen wollen, müssen Sie das aktuellste Backup-Image als Zielimage auswählen. Das aktuellste Backup-Image ist Image TS5, das in Protokollkette 2 enthalten ist:

```
db2 restore db sample rebuild with all tablespaces in database taken at
      TS5 without prompting
db2 rollforward db sample to end of logs
db2 rollforward db sample stop
```

Diese Befehle stellen die Backup-Images TS5, TS4, TS1 und DB1 automatisch wieder her und führen eine aktualisierende Recovery der Datenbank bis zum Ende von Protokollkette 2 aus.

Anmerkung: Alle Protokolle, die zu Protokollkette 2 gehören, müssen für die vollständige Ausführung der aktualisierenden Recoveryoperationen zugänglich sein.

Beispiel 2

Dieses zweite Beispiel veranschaulicht die CLP-Befehle, die Sie ausführen müssen, um die Datenbank SAMPLE bis zum Ende von Protokollkette 1 erneut zu erstellen. Das Zielimage, das Sie dazu auswählen, sollte das aktuellste Backup-Image in Protokollkette 1 sein. Dies ist TS3:

```
db2 restore db sample rebuild with all tablespaces in database
      taken at TS3 without prompting
db2 rollforward db sample to end of logs
db2 rollforward db sample stop
```

Diese Befehle stellen die Backup-Images TS3, TS2, TS1 und DB1 automatisch wieder her und führen eine aktualisierende Recovery der Datenbank bis zum Ende von Protokollkette 1 aus.

Anmerkung: Alle Protokolle, die zu Protokollkette 1 gehören, müssen für die vollständige Ausführung der aktualisierenden Recoveryoperationen zugänglich sein.

Auswählen des falschen Zielimages zur erneuten Erstellung

Nehmen Sie an, eine Datenbank mit dem Namen SAMPLE2 enthält die folgenden Tabellenbereiche:

- SYSCATSPACE (Systemkataloge)
- USERSP1 (Tabellenbereich für Benutzerdaten)
- USERSP2 (Tabellenbereich für Benutzerdaten)

Abb. 25 auf Seite 428 zeigt die Backup-Protokollkette für SAMPLE2, die aus den folgenden Backups besteht:

1. BK1 ist ein Datenbankgesamtbackup mit allen Tabellenbereichen.
2. BK2 ist ein Tabellenbereichsgesamtbackup von USERSP1.
3. BK3 ist ein Tabellenbereichsgesamtbackup von USERSP2.

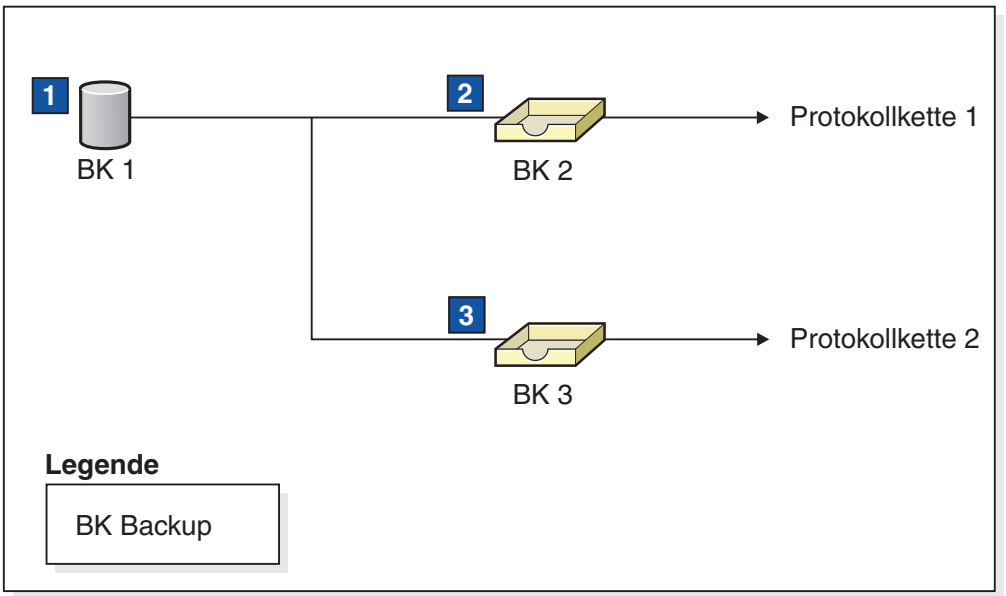


Abbildung 25. Backup-Protokollkette für die Datenbank SAMPLE2

Das folgende Beispiel zeigt den CLP-Befehl, den Sie ausführen müssen, um die Datenbank aus BK3 mit den Tabellenbereichen SYSCATSPACE und USERSP2 erneut zu erstellen:

```
db2 restore db sample2 rebuild with tablespace (SYSCATSPACE,
      USERSP2) taken at BK3 without prompting
```

Nehmen Sie nun an, dass nach Abschluss dieser Restoreoperation auch der Tabellenbereich USERSP1 wiederhergestellt werden soll und daher der folgende Befehl ausgeführt wird:

```
db2 restore db sample2 tablespace (USERSP1) taken at BK2
```

Diese Restoreoperation schlägt fehl und gibt eine Nachricht aus, die besagt, dass BK2 aus der falschen Protokollkette stammt (SQL2154N). Wie Abb. 25 zu entnehmen ist, kann nur das Image BK1 für den Restore von USERSP1 verwendet werden. Daher muss der folgende Befehl eingegeben werden:

```
db2 restore db sample2 tablespace (USERSP1) taken at BK1
```

Dieser Befehl wird erfolgreich ausgeführt, sodass für die Datenbank eine entsprechende aktualisierende Recovery erfolgen kann.

Erneutes Erstellen ausgewählter Tabellenbereiche

Die erneute Erstellung einer Datenbank bietet Ihnen die Möglichkeit, eine Datenbank zu erstellen, die eine Untergruppe der Tabellenbereiche enthält, die Bestandteil der ursprünglichen Datenbank sind.

Informationen zu diesem Vorgang

Die erneute Erstellung nur einer Untergruppe von Tabellenbereichen in einer Datenbank kann in folgenden Situationen nützlich sein:

- In einer Test- und Entwicklungsumgebung, in der Sie nur mit einem Teil der Tabellenbereiche arbeiten möchten.
- In einem Recoveryfall, in dem Sie Tabellenbereiche, die kritischer sind, schneller wieder online verfügbar machen müssen als andere, können Sie zunächst eine

Teilgruppe von Tabellenbereichen wiederherstellen und die Wiederherstellung anderer Tabellenbereiche zu einem späteren Zeitpunkt nachholen.

Betrachten Sie das folgende Beispiel zur erneuten Erstellung einer Datenbank, die eine Teilgruppe der Tabellenbereiche enthält, die Bestandteil der ursprünglichen Datenbank sind.

Für dieses Beispiel wird angenommen, dass eine Datenbank mit dem Namen MYDB vorhanden ist und die folgenden Tabellenbereiche enthält:

- SYSCATSPACE (Systemkataloge)
- USERSP1 (Tabellenbereich für Benutzerdaten)
- USERSP2 (Tabellenbereich für Benutzerdaten)
- USERSP3 (Tabellenbereich für Benutzerdaten)

Abb. 26 zeigt, dass die folgenden Backups erstellt wurden:

- BK1 ist ein Backup von SYSCATSPACE und USERSP1.
- BK2 ist ein Backup von USERSP2 und USERSP3.
- BK3 ist ein Backup von USERSP3.

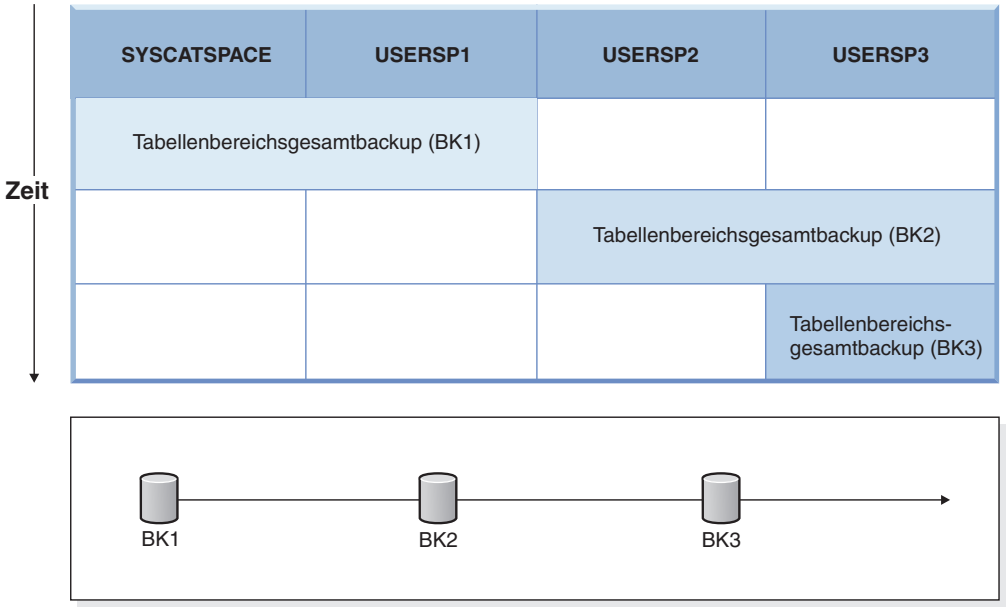


Abbildung 26. Für die Datenbank SAMPLE verfügbare Backup-Images

Die folgende Prozedur veranschaulicht die Verwendung der Befehle **RESTORE DATABASE** und **ROLLFORWARD DATABASE** über den Befehlszeilenprozessor (CLP) zur erneuten Erstellung nur der Tabellenbereiche SYSCATSPACE und USERSP1 bis zum Ende der Protokolle:

```
db2 restore db mydb rebuild with all tablespaces in image
             taken at BK1 without prompting
   db2 rollforward db mydb to end of logs
   db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und nur die Tabellenbereiche SYSCATSPACE und USERSP1 befinden sich im Status NORMAL. Die Tabellenbereiche USERSP2 und USERSP3 befinden sich im Status

Restore anstehend. Zu einem späteren Zeitpunkt können Sie weiterhin auch die Tabellenbereiche USERSP2 und USERSP3 wiederherstellen.

Erneutes Erstellen mit Images inkrementeller Backups

Sie können eine Datenbank mithilfe inkrementeller Backup-Images erneut erstellen.

Standardmäßig versucht das Restoredienstprogramm, einen automatischen inkrementellen Restore für alle inkrementellen Images auszuführen. Wenn Sie die Option **INCREMENTAL** des Befehls **RESTORE DATABASE** nicht verwenden, das Zielimage jedoch ein inkrementelles Backup-Image ist, bedeutet dies, dass das Restoredienstprogramm eine Rebuildoperation unter Verwendung des automatischen inkrementellen Restores ausführt. Wenn das Zielimage kein inkrementelles Image ist, jedoch ein anderes erforderliches Image ein inkrementelles Image ist, stellt das Restoredienstprogramm sicher, dass solche inkrementellen Images durch den automatischen inkrementellen Restore wiederhergestellt werden. Das Restoredienstprogramm verhält sich immer gleich, unabhängig davon, ob Sie die Option **INCREMENTAL** mit der Option **AUTOMATIC** angeben oder nicht.

Wenn Sie die Option **INCREMENTAL**, jedoch nicht die Option **AUTOMATIC** angeben, müssen Sie den gesamten Prozess zur erneuten Erstellung (Rebuild) manuell ausführen. Das Restoredienstprogramm stellt in diesem Fall nur die Anfangsmetadaten aus dem Zielimage wieder her, wie es dies auch bei einem regulären manuellen inkrementellen Restore tun würde. Anschließend müssen Sie dann den Restore des Zielimages unter Verwendung der erforderlichen inkrementellen Restorekette zu Ende führen. Und schließlich müssen Sie die verbleibenden Images wiederherstellen, um die Datenbank erneut zu erstellen.

Es wird empfohlen, den automatischen inkrementellen Restore zur erneuten Erstellung einer Datenbank zu verwenden. Nur bei einem Restorefehler sollten Sie versuchen, eine Datenbank mithilfe manueller Methoden erneut zu stellen.

Erneutes Erstellen von partitionierten Datenbanken

Zur erneuten Erstellung (Rebuild) einer partitionierten Datenbank erstellen Sie jede Datenbankpartition separat erneut. Für jede Datenbankpartition, angefangen bei der Katalogpartition, stellen Sie zunächst alle Tabellenbereiche, die Sie benötigen, mit RESTORE wieder her. Alle Tabellenbereiche, die nicht wiederhergestellt werden, werden in den Status *Restore anstehend* versetzt.

Wenn alle Datenbankpartitionen wiederhergestellt sind, setzen Sie den Befehl **ROLLFORWARD DATABASE** in der Katalogpartition ab, um eine aktualisierende Recovery für alle Datenbankpartitionen auszuführen.

Informationen zu diesem Vorgang

Anmerkung: Wenn Sie zu einem späteren Zeitpunkt Tabellenbereiche wiederherstellen müssen, die ursprünglich nicht in die Rebuildphase mit eingeschlossen waren, müssen Sie sicherstellen, dass bei der nachfolgenden aktualisierenden Recovery des Tabellenbereichs das Dienstprogramm für aktualisierende Recovery (Rollforward) alle Daten über die Datenbankpartitionen hinweg synchron hält. Wenn ein Tabellenbereich bei der ursprünglichen Restore- und aktualisierenden Recoveryoperation versehentlich unberücksichtigt bleibt, wird sein Fehlen unter Umständen erst erkannt, wenn bei einem Versuch, auf die Daten zuzugreifen, ein Datenzugriffsfehler auftritt. Sie müssen den fehlenden Tabellenbereich mit Restore wiederherstellen und eine aktualisierende Recovery für ihn ausführen, um ihn mit den übrigen Partitionen zu resynchronisieren.

Betrachten Sie das folgende Beispiel im Hinblick auf die erneute Erstellung einer partitionierten Datenbank mithilfe von Backup-Images, die auf Tabellenbereichsebene erstellt wurden.

Für dieses Beispiel wird angenommen, dass eine wiederherstellbare Datenbank mit dem Namen SAMPLE vorhanden ist und drei Datenbankpartitionen enthält:

- Datenbankpartition 1 enthält die Tabellenbereiche SYSCATSPACE, USERSP1 und USERSP2 und ist die Katalogpartition.
- Datenbankpartition 2 enthält die Tabellenbereiche USERSP1 und USERSP3.
- Datenbankpartition 3 enthält die Tabellenbereiche USERSP1, USERSP2 und USERSP3.

Die folgenden Backups wurden erstellt, wobei BK*xy* die Backupnummer *x* in Partition *y* bezeichnet:

- BK11 ist ein Backup der Tabellenbereiche SYSCATSPACE, USERSP1 und USERSP2.
- BK12 ist ein Backup der Tabellenbereiche USERSP2 und USERSP3.
- BK13 ist ein Backup der Tabellenbereiche USERSP1, USERSP2 und USERSP3.
- BK21 ist ein Backup des Tabellenbereichs USERSP1.
- BK22 ist ein Backup des Tabellenbereichs USERSP1.
- BK23 ist ein Backup des Tabellenbereichs USERSP1.
- BK31 ist ein Backup des Tabellenbereichs USERSP2.
- BK33 ist ein Backup des Tabellenbereichs USERSP2.
- BK42 ist ein Backup des Tabellenbereichs USERSP3.
- BK43 ist ein Backup des Tabellenbereichs USERSP3.

Die folgende Prozedur veranschaulicht die Verwendung der Befehle **RESTORE DATABASE** und **ROLLFORWARD DATABASE** über den Befehlszeilenprozessor (CLP) zur erneuten Erstellung der gesamten Datenbank bis zum Ende der Protokolle.

Vorgehensweise

1. Führen Sie in Datenbankpartition 1 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db sample rebuild with all tablespaces in database
         taken at BK31 without prompting
   ```

2. Führen Sie in Datenbankpartition 2 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db sample rebuild with tablespaces in database
         taken at BK42 without prompting
   ```

3. Führen Sie in Datenbankpartition 3 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db sample rebuild with all tablespaces in database
         taken at BK43 without prompting
   ```

4. Führen Sie in der Katalogpartition einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus:

   ```
   db2 rollforward db sample to end of logs
   ```

5. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

   ```
   db2 rollforward db sample stop
   ```

Nächste Schritte

An diesem Punkt kann eine Verbindung zur Datenbank in allen Datenbankpartitionen hergestellt werden, und alle Tabellenbereiche befinden sich im Status NORMAL.

Einschränkungen für das erneute Erstellen von Datenbanken

Mit dem Befehl **REBUILD** kann eine Gruppe von Restorebefehlen ausgeführt werden. Dieser Befehl ist nützlich, es müssen jedoch Einschränkungen beachtet werden.

Die folgende Liste enthält eine Zusammenfassung der Einschränkungen für die erneute Erstellung (Option REBUILD):

- Einer der Tabellenbereiche, die erneut erstellt werden, muss der Tabellenbereich SYSCATSPACE in der Katalogpartition sein.
- Sie müssen entweder Befehle über den Befehlszeilenprozessor (CLP) absetzen oder die entsprechenden Anwendungsprogrammierschnittstellen (APIs) verwenden, um eine Rebuildoperation auszuführen.
- Die Option **REBUILD** kann nicht für ein Zielimage aus einer Version vor Version 9.1 verwendet werden, es sei denn, es handelt sich um ein Image eines Offline-Backups der Datenbank. Wenn das Zielimage ein Offline-Backup der Datenbank ist, können nur die Tabellenbereiche in diesem Image für die erneute Erstellung verwendet werden. Die Datenbank muss nach einem erfolgreichen Abschluss der Rebuildoperation migriert werden. Versuche, eine erneute Erstellung mit irgendeinem anderen Typ von Zielimage aus einer Version vor Version 9.1 durchzuführen, schlagen fehl.
- Die Option **REBUILD** kann nicht für ein Zielimage angegeben werden, das unter einem anderen Betriebssystem erstellt wurde als dem, unter dem es wiederhergestellt wird, sofern es sich nicht um ein Datenbankgesamtbackup handelt. Wenn das Zielimage ein Gesamtbackup der Datenbank ist, können nur die Tabellenbereiche in diesem Image für die erneute Erstellung verwendet werden. Versuche, eine erneute Erstellung mit irgendeinem anderen Typ von Zielimage durchzuführen, das unter einem anderen Betriebssystem erstellt wurde als dem, unter dem es wiederhergestellt wird, schlagen fehl.
- Die Option `TRANSPORT` ist nicht kompatibel mit der Option `REBUILD`.

Sitzungen zur erneuten Erstellung - Beispiele für CLP

Dieser Abschnitt enthält eine Reihe von Beispielen für Rebuildoperationen.

Szenario 1

Für die folgenden Beispiele wird angenommen, dass eine wiederherstellbare Datenbank des Namens MYDB vorhanden ist, die folgende Tabellenbereiche enthält:

- SYSCATSPACE (Systemkataloge)
- USERSP1 (Tabellenbereich für Benutzerdaten)
- USERSP2 (Tabellenbereich für Benutzerdaten)
- USERSP3 (Tabellenbereich für Benutzerdaten)

Die folgenden Backups wurden erstellt:

- BK1 ist ein Backup von SYSCATSPACE und USERSP1.
- BK2 ist ein Backup von USERSP2 und USERSP3.
- BK3 ist ein Backup von USERSP3.

Beispiel 1

Mit der folgenden Prozedur wird die gesamte Datenbank bis zum aktuellsten Zeitpunkt erneut erstellt:

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

```
db2 restore db mydb rebuild with all tablespaces in database
         taken at BK3 without prompting
```

2. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to end of logs
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und alle Tabellenbereiche befinden sich im Status NORMAL.

Beispiel 2

Mit der folgenden Prozedur werden nur die Tabellenbereiche SYSCATSPACE und USERSP2 bis zu einem bestimmten Zeitpunkt erneut erstellt (wobei das Ende von BK3 weiter zurückliegt als der bestimmte Zeitpunkt, der wiederum weiter zurückliegt als das Ende der Protokolle):

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus und geben Sie die Tabellenbereiche an, die eingeschlossen werden sollen.

```
db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP2)
      taken at BK2 without prompting
```

2. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO PIT** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to PIT
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und nur die Tabellenbereiche SYSCATSPACE und USERSP2 befinden sich im Status NORMAL. Die Tabellenbereiche USERSP1 und USERSP3 befinden sich im Status *Restore anstehend* (RESTORE_PENDING).

Gehen Sie zur Wiederherstellung der Tabellenbereiche USERSP1 und USERSP3 zu einem späteren Zeitpunkt durch normale Restoreoperationen für die Tabellenbereiche (ohne Option **REBUILD**) wie folgt vor:

1. Führen Sie den Befehl **RESTORE DATABASE** *ohne* die Option **REBUILD** aus und geben Sie den wiederherzustellenden Tabellenbereich an. Stellen Sie zunächst USERSP1 wieder her:

```
db2 restore db mydb tablespace (USERSP1) taken at BK1 without prompting
```

2. Stellen Sie anschließend USERSP3 wieder her:

```
db2 restore db mydb tablespace taken at BK3 without prompting
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **END OF LOGS** aus und geben Sie die wiederherzustellenden Tabellenbereiche an (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to end of logs tablespace (USERSP1, USERSP3)
```

Die aktualisierende Recovery (Rollforward) wendet alle Protokolle bis zum Zeitpunkt (PIT) an und stoppt dann für diese beiden Tabellenbereiche, da an ihnen seit der ersten aktualisierenden Recovery keine Arbeiten ausgeführt wurden.

4. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

Beispiel 3

Mit der folgenden Prozedur werden nur die Tabellenbereiche SYSCATSPACE und USERSP1 bis zum Ende der Protokolle erneut erstellt:

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

```
db2 restore db mydb rebuild with all tablespaces in image
   taken at BK1 without prompting
```

2. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to end of logs
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und nur die Tabellenbereiche SYSCATSPACE und USERSP1 befinden sich im Status NORMAL. Die Tabellenbereiche USERSP2 und USERSP3 befinden sich im Status *Restore anstehend* (RESTORE_PENDING).

Beispiel 4

Im folgenden Beispiel befinden sich die Backups BK1 und BK2 nicht mehr an derselben Position wie in der Datei des Recoveryprotokolls angeben. Dies ist jedoch bei der Ausführung der Rebuildoperation nicht bekannt.

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus, indem Sie angeben, dass die gesamte Datenbank bis zum aktuellsten Zeitpunkt erneut erstellt werden soll:

```
db2 restore db mydb rebuild with all tablespaces in database
         taken at BK3 without prompting
```

   An diesem Punkt ist das Zielimage erfolgreich wiederhergestellt. Jedoch wird ein Fehler aus dem Restoredienstprogramm zurückgegeben, der besagt, dass ein erforderliches Image nicht gefunden werden konnte.

2. In diesem Fall müssen Sie die Rebuildoperation manuell ausführen. Da sich die Datenbank in der Rebuildphase befindet, kann dies wie folgt geschehen:
   a. Führen Sie einen Befehl **RESTORE DATABASE** aus, indem Sie die Position des Backup-Image BK1 angeben:

```
db2 restore db mydb tablespace taken at BK1 from position
         without prompting
```

   b. Führen Sie einen Befehl **RESTORE DATABASE** aus, indem Sie die Position des Backup-Image BK2 angeben:

```
db2 restore db mydb tablespace (USERSP2) taken at BK2 from
   position without prompting
```

   c. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to end of logs
```

d. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und alle Tabellenbereiche befinden sich im Status NORMAL.

Beispiel 5

In diesem Beispiel enthält der Tabellenbereich USERSP3 unabhängige Daten, die zur Generierung eines speziellen Berichts benötigt werden, jedoch wünschen Sie nicht, dass die Berichtsgenerierung die ursprüngliche Datenbank stört. Um Zugriff auf die Daten zu erhalten, ohne die ursprüngliche Datenbank zu beeinträchtigen, können Sie mit **REBUILD** eine neue Datenbank mit nur diesem Tabellenbereich und dem Tabellenbereich SYSCATSPACE generieren. Der Tabellenbereich SYSCATSPACE ist ebenfalls erforderlich, um nach der Restore- und der aktualisierenden Recoveryoperation die Herstellung einer Verbindung zu der Datenbank zu ermöglichen.

Gehen Sie zum Erstellen einer neuen Datenbank mit den aktuellsten Daten in den Tabellenbereichen SYSCATSPACE und USERSP3 wie folgt vor:

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus und geben Sie an, dass die Tabellenbereiche SYSCATSPACE und USERSP3 in einer neuen Datenbank namens NEWDB wiederhergestellt werden sollen:

```
db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP3)
     taken at BK3 into newdb without prompting
```

2. Führen Sie in NEWDB einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db newdb to end of logs
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db newdb stop
```

An diesem Punkt kann eine Verbindung zur neuen Datenbank hergestellt werden, und nur die Tabellenbereiche SYSCATSPACE und USERSP3 befinden sich im Status NORMAL. Die Tabellenbereiche USERSP1 und USERSP2 befinden sich im Status *Restore anstehend* (RESTORE_PENDING).

Anmerkung: Wenn die Containerpfade zwischen der aktuellen Datenbank und der neuen Datenbank problematisch sind (zum Beispiel wenn die Container für die ursprüngliche Datenbank geändert werden müssen, weil das Dateisystem nicht vorhanden ist, oder wenn die Container bereits von der ursprünglichen Datenbank verwendet werden), müssen Sie einen umgeleiteten Restore ausführen. Dieses Beispiel geht von der Annahme aus, dass die Standarddatenbankpfade für dynamischen Speicher für die Tabellenbereiche verwendet werden.

Szenario 2

Für das folgende Beispiel wird angenommen, dass eine wiederherstellbare Datenbank des Namens MYDB vorhanden ist, die den Tabellenbereich SYSCATSPACE und eintausend Benutzertabellenbereiche mit Namen der Form T*xxxx* besitzt. Dabei stellt *xxxx* die Tabellenbereichsnummer (z. B. T0001) dar. Es ist ein Datenbankgesamtbackup (BK1) vorhanden.

Beispiel 6

Mit der folgenden Prozedur werden alle Tabellenbereiche außer T0999 und T1000 wiederhergestellt:

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

```
db2 restore db mydb rebuild with all tablespaces in image except
     tablespace (T0999, T1000) taken at BK1 without prompting
```

2. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to end of logs
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und alle Tabellenbereiche außer T0999 und T1000 befinden sich im Status NORMAL. Die Tabellenbereiche T0999 und T1000 befinden sich im Status *Restore anstehend* (RESTORE_PENDING).

Szenario 3

Die Beispiele in diesem Szenario veranschaulichen, wie eine wiederherstellbare Datenbank mithilfe inkrementeller Backups erneut erstellt wird. Für die folgenden Beispiele wird angenommen, dass eine Datenbank des Namens MYDB vorhanden ist, die folgende Tabellenbereiche enthält:

- SYSCATSPACE (Systemkataloge)
- USERSP1 (Datentabellenbereich)
- USERSP2 (Tabellenbereich für Benutzerdaten)
- USERSP3 (Tabellenbereich für Benutzerdaten)

Die folgenden Backups wurden erstellt:

- FULL1 ist ein Gesamtbackup der Tabellenbereiche SYSCATSPACE, USERSP1, USERSP2 und USERSP3.
- DELTA1 ist ein Deltabackup der Tabellenbereiche SYSCATSPACE und USERSP1.
- INCR1 ist ein inkrementelles Backup der Tabellenbereiche USERSP2 und USERSP3.
- DELTA2 ist ein Deltabackup der Tabellenbereiche SYSCATSPACE, USERSP1, USERSP2 und USERSP3.
- DELTA3 ist Deltabackup des Tabellenbereichs USERSP2.
- FULL2 ist ein Gesamtbackup des Tabellenbereichs USERSP1.

Beispiel 7

Mit der folgenden Prozedur werden nur die Tabellenbereiche SYSCATSPACE und USERSP2 bis zum aktuellsten Zeitpunkt durch eine inkrementelle automatische Restoreoperation erneut erstellt.

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus. Die Option **INCREMENTAL AUTO** ist optional. Das Restoredienstprogramm erkennt die Granularität des Images und verwendet einen automatischen inkrementellen Restore, wenn er erforderlich ist.

```
db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP2)
     incremental auto taken at DELTA3 without prompting
```

2. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to end of logs
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und nur die Tabellenbereiche SYSCATSPACE und USERSP2 befinden sich im Status NORMAL. Die Tabellenbereiche USERSP1 und USERSP3 befinden sich im Status *Restore anstehend* (RESTORE_PENDING).

Beispiel 8

Mit der folgenden Prozedur wird die gesamte Datenbank bis zum aktuellsten Zeitpunkt durch einen inkrementellen automatischen Restore erneut erstellt.

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus. Die Option **INCREMENTAL AUTO** ist optional. Das Restoredienstprogramm erkennt die Granularität des Images und verwendet einen automatischen inkrementellen Restore, wenn er erforderlich ist.

```
db2 restore db mydb rebuild with all tablespaces in database
        incremental auto taken at DELTA3 without prompting
```

2. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to end of logs
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und alle Tabellenbereiche befinden sich im Status NORMAL.

Beispiel 9

Mit der folgenden Prozedur wird die gesamte Datenbank außer Tabellenbereich USERSP3 bis zum aktuellsten Zeitpunkt erneut erstellt.

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus. Obwohl es sich bei dem Zielimage um ein nicht inkrementelles Image handelt, erkennt das Restoredienstprogramm, dass die erforderliche Rebuildkette inkrementelle Images enthält, und stellt diese Images automatisch inkrementell wieder her.

```
db2 restore db mydb rebuild with all tablespaces in database except
        tablespace (USERSP3) taken at FULL2 without prompting
```

2. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to end of logs
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

Szenario 4

Die Beispiele in diesem Szenario veranschaulichen, wie eine wiederherstellbare Datenbank mithilfe von Backup-Images erneut erstellt wird, die Protokolldateien enthalten. Für die folgenden Beispiele wird angenommen, dass eine Datenbank des Namens MYDB vorhanden ist, die folgende Tabellenbereiche enthält:

- SYSCATSPACE (Systemkataloge)
- USERSP1 (Tabellenbereich für Benutzerdaten)
- USERSP2 (Tabellenbereich für Benutzerdaten)

Beispiel 10

Mit der folgenden Prozedur wird die Datenbank nur mit den Tabellenbereichen SYSCATSPACE und USERSP2 bis zum aktuellsten Zeitpunkt erneut erstellt. Es ist ein online erstelltes Image eines Datenbankgesamtbackups (BK1) vorhanden, das Protokolldateien enthält.

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

```
db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP2)
     taken at BK1 logtarget /logs without prompting
```

2. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle nach dem Ende von BK1 gesichert wurden und zugänglich sind):

```
db2 rollforward db mydb to end of logs overflow log path (/logs)
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und nur die Tabellenbereiche SYSCATSPACE und USERSP2 befinden sich im Status NORMAL. Der Tabellenbereich USERSP1 befindet sich im Status *Restore anstehend* (RESTORE_PENDING).

Beispiel 11

Mit der folgenden Prozedur wird die Datenbank bis zum aktuellsten Zeitpunkt erneut erstellt. Es sind zwei online erstellte Images von Tabellenbereichsgesamtbackups vorhanden, die Protokolldateien enthalten:

- BK1 ist ein Backup von SYSCATSPACE mit den Protokolldateien 10-45.
- BK2 ist ein Backup von USERSP1 und USERSP2 mit den Protokolldateien 64-80.

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

```
db2 restore db mydb rebuild with all tablespaces in database
         taken at BK2 logtarget /logs without prompting
```

   Die aktualisierende Recoveryoperation beginnt mit Protokolldatei 10, die sie immer im Überlaufprotokollpfad findet, wenn sie sich nicht im primären Protokolldateipfad befindet. Der Protokollbereich 46-63 muss für die aktualisierende Recovery verfügbar gemacht werden, da diese Protokolldateien in keinem Backup-Image enthalten sind.

2. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** unter Verwendung des Überlaufprotokollpfads für die Protokolldateien 64-80 aus:

```
db2 rollforward db mydb to end of logs overflow log path (/logs)
```

3. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und alle Tabellenbereiche befinden sich im Status NORMAL.

Szenario 5

Für die folgenden Beispiele wird angenommen, dass eine wiederherstellbare Datenbank des Namens MYDB vorhanden ist, die folgende Tabellenbereiche enthält:

- SYSCATSPACE (0), SMS-Systemkatalog (relativ definierter Container)
- USERSP1 (1) DMS-Tabellenbereich für Benutzerdaten (absolut definierter Container /usersp2)

- USERSP2 (2) DMS-Tabellenbereich für Benutzerdaten (absolut definierter Container /usersp3)

Die folgenden Backups wurden erstellt:

- BK1 ist ein Backup von SYSCATSPACE
- BK2 ist ein Backup von USERSP1 und USERSP2
- BK3 ist ein Backup von USERSP2

Beispiel 12

Mit der folgenden Prozedur wird die gesamte Datenbank bis zum aktuellsten Zeitpunkt durch einen umgeleiteten Restore erneut erstellt.

1. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db mydb rebuild with all tablespaces in database
            taken at BK3 redirect without prompting
   ```

2. Setzen Sie für jeden Tabellenbereich, dessen Container Sie erneut definieren möchten, einen Befehl **SET TABLESPACE CONTAINERS** ab. Beispiel:

   ```
   db2 set tablespace containers for 3 using (file '/newusersp1' 10000)
   ```

3.

   ```
   db2 set tablespace containers for 4 using (file '/newusersp2' 15000)
   ```

4. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **CONTINUE** aus:

   ```
   db2 restore db mydb continue
   ```

5. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und zugänglich sind):

   ```
   db2 rollforward db mydb to end of logs
   ```

6. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

   ```
   db2 rollforward db mydb stop
   ```

An diesem Punkt kann eine Verbindung zur Datenbank hergestellt werden, und alle Tabellenbereiche befinden sich im Status NORMAL.

Szenario 6

Für die folgenden Beispiele wird angenommen, dass eine Datenbank des Namens MYDB mit drei Datenbankpartitionen vorhanden ist:

- Datenbankpartition 1 enthält die Tabellenbereiche SYSCATSPACE, USERSP1 und USERSP2 und ist die Katalogpartition.
- Datenbankpartition 2 enthält die Tabellenbereiche USERSP1 und USERSP3.
- Datenbankpartition 3 enthält die Tabellenbereiche USERSP1, USERSP2 und USERSP3.

Die folgenden Backups wurden erstellt, wobei BK*xy* die Backupnummer *x* in Partition *y* bezeichnet:

- BK11 ist ein Backup der Tabellenbereiche SYSCATSPACE, USERSP1 und USERSP2.
- BK12 ist ein Backup der Tabellenbereiche USERSP2 und USERSP3.
- BK13 ist ein Backup der Tabellenbereiche USERSP1, USERSP2 und USERSP3.
- BK21 ist ein Backup des Tabellenbereichs USERSP1.
- BK22 ist ein Backup des Tabellenbereichs USERSP1.
- BK23 ist ein Backup des Tabellenbereichs USERSP1.
- BK31 ist ein Backup des Tabellenbereichs USERSP2.

- BK33 ist ein Backup des Tabellenbereichs USERSP2.
- BK42 ist ein Backup des Tabellenbereichs USERSP3.
- BK43 ist ein Backup des Tabellenbereichs USERSP3.

Beispiel 13

Mit der folgenden Prozedur wird die gesamte Datenbank bis zum Ende der Protokolle erneut erstellt.

1. Führen Sie in Datenbankpartition 1 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db mydb rebuild with all tablespaces in database
           taken at BK31 without prompting
   ```

2. Führen Sie in Datenbankpartition 2 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db mydb rebuild with tablespaces in database taken at
           BK42 without prompting
   ```

3. Führen Sie in Datenbankpartition 3 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db mydb rebuild with all tablespaces in database
           taken at BK43 without prompting
   ```

4. Führen Sie in der Katalogpartition einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus (unter der Voraussetzung, dass alle Protokolle gesichert wurden und in allen Datenbankpartitionen zugänglich sind):

   ```
   db2 rollforward db mydb to end of logs
   ```

5. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

   ```
   db2 rollforward db mydb stop
   ```

An diesem Punkt kann eine Verbindung zur Datenbank in allen Datenbankpartitionen hergestellt werden, und alle Tabellenbereiche befinden sich im Status NORMAL.

Beispiel 14

Mit der folgenden Prozedur werden die Tabellenbereiche SYSCATSPACE, USERSP1 und USERSP2 bis zum aktuellsten Zeitpunkt erneut erstellt.

1. Führen Sie in Datenbankpartition 1 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db mydb rebuild with all tablespaces in database
           taken at BK31 without prompting
   ```

2. Führen Sie in Datenbankpartition 2 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db mydb rebuild with all tablespaces in image taken at
           BK22 without prompting
   ```

3. Führen Sie in Datenbankpartition 3 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db mydb rebuild with all tablespaces in image taken at
           BK33 without prompting
   ```

   Hinweis: Dieser Befehl lässt den Tabellenbereich USERSP1 aus, der zum Abschließen der Rebuildoperation erforderlich ist.

4. Führen Sie in der Katalogpartition einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus:

   ```
   db2 rollforward db mydb to end of logs
   ```

5. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

```
db2 rollforward db mydb stop
```

Die aktualisierende Recovery wird erfolgreich beendet. Zur Datenbank kann eine Verbindung in allen Datenbankpartitionen hergestellt werden. Alle Tabellenbereiche mit folgenden Ausnahmen befinden sich im Status NORMAL: Der Tabellenbereich USERSP3 befindet sich in allen Datenbankpartitionen, in denen er enthalten ist, im Status *Restore anstehend* (RESTORE PENDING), und der Tabellenbereich USERSP1 hat in Datenbankpartition 3 den Status *Restore anstehend* (RESTORE PENDING).

Wenn ein Versuch unternommen wird, auf Daten in USERSP1 in Datenbankpartition 3 zuzugreifen, tritt ein Datenzugriffsfehler auf. Um diesen zu beheben, muss der Tabellenbereich USERSP1 wiederhergestellt werden:

a. Führen Sie in Datenbankpartition 3 einen Befehl **RESTORE DATABASE** aus, indem Sie ein Backup-Image angeben, das den Tabellenbereich USERSP1 enthält:

   ```
   db2 restore db mydb tablespace taken at BK23 without prompting
   ```

b. Führen Sie in der Katalogpartition einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** und der Option **AND STOP** aus:

   ```
   db2 rollforward db mydb to end of logs on dbpartitionnum (3) and stop
   ```

An diesem Punkt kann auf die Daten des Tabellenbereichs USERSP1 in Datenbankpartition 3 zugegriffen werden, da sich der Tabellenbereich im Status NORMAL befindet.

Szenario 7

Für die folgenden Beispiele wird angenommen, dass eine *nicht wiederherstellbare* Datenbank des Namens MYDB mit den folgenden Tabellenbereichen vorhanden ist:

- SYSCATSPACE (0), SMS-Systemkatalog
- USERSP1 (1), DMS-Tabellenbereich für Benutzerdaten
- USERSP2 (2), DMS-Tabellenbereich für Benutzerdaten

Es ist nur ein Backup (BK1) der Datenbank vorhanden.

Beispiel 15

Die folgende Prozedur veranschaulicht die Verwendung der Rebuildoperation für eine nicht wiederherstellbare Datenbank.

Erstellen Sie die Datenbank nur mit den Tabellenbereichen SYSCATSPACE und USERSP1 erneut:

```
db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP1)
   taken at BK1 without prompting
```

Im Anschluss an die Restoreoperation kann eine Verbindung zur Datenbank hergestellt werden. Wenn Sie den Befehl **LIST TABLESPACES** oder die Tabellenfunktion MON_GET_TABLESPACE ausführen, sehen Sie, dass sich die Tabellenbereiche SYSCATSPACE und USERSP1 im Status NORMAL befinden, während sich der Tabellenbereich USERSP2 im Status *Löschen anstehend* bzw. *Offline* (DELETE_PENDING/OFFLINE) befindet. Sie können nun mit den beiden Tabellenbereichen arbeiten, die sich im Status NORMAL befinden.

Wenn Sie ein Datenbankbackup erstellen wollen, müssen Sie zunächst den Tabellenbereich USERSP2 mithilfe der Anweisung DROP TABLESPACE löschen, da das Backup ansonsten fehlschlägt.

Wenn Sie den Tabellenbereich USERSP2 zu einem späteren Zeitpunkt wiederherstellen wollen, müssen Sie erneut einen Befehl RESTORE DATABASE mit BK1 ausführen.

Überwachen des Fortschritts von Restoreoperationen

Mit dem **LIST UTILITIES** können Restoreoperationen für eine Datenbank überwacht werden.

Vorgehensweise

Setzen Sie den Befehl **LIST UTILITIES** ab und geben Sie den Parameter **SHOW DETAIL** an.

```
LIST UTILITIES SHOW DETAIL
```

Ergebnisse

Für Restoreoperationen wird kein geschätzter Anfangswert angegeben. Stattdessen wird UNKNOWN angegeben. Während die einzelnen Puffer aus dem Image gelesen werden, wird die tatsächliche Menge gelesener Byte aktualisiert. Für automatische inkrementelle Restoreoperationen, bei denen möglicherweise mehrere Images wiederhergestellt werden, wird der Fortschritt anhand von Phasen überwacht. Jede Phase stellt ein Image dar, das aus der Kette von inkrementellen Backups wiederhergestellt werden soll. Zu Beginn wird nur eine Phase angegeben. Nachdem das erste Image wiederhergestellt wurde, wird die Gesamtanzahl der Phasen angezeigt. Beim Wiederherstellen der einzelnen Images werden die Anzahl der abgeschlossenen Phasen und die Anzahl der verarbeiteten Byte aktualisiert.

Beispiel

Es folgt eine Beispielausgabe der Leistungsüberwachung einer Restoreoperation:

```
ID                                   = 6
Typ                                  = RESTORE
Datenbankname                        = SAMPLE
Partitionsnummer                     = 0
Beschreibung                         = db
Startzeit                            = 08/04/2011 12:24:47.494191
Status                               = Wird ausgeführt
Aufruftyp                            = Benutzer
   Fortschrittsüberwachung:
      Abgeschlossene Arbeit          = 4096 Byte
      Startzeit                      = 08/04/2011 12:24:47.494197
```

Optimieren der Leistung des Restoredienstprogramms

Wenn Sie eine Restoreoperation ausführen, wählen DB2-Datenbankprodukte automatisch optimale Werte für die Anzahl Puffer, die Puffergröße und die Parallelitätseinstellungen aus. Die Werte basieren auf der Menge des für Dienstprogramme verfügbaren Zwischenspeichers, der Anzahl verfügbarer Prozessoren und der Datenbankkonfiguration.

Daher sollten Sie je nach der Größe der auf Ihrem System verfügbaren Speicherkapazität in Betracht ziehen, mehr Speicher zuzuordnen, indem Sie den Wert des Konfigurationsparameters **util_heap_sz** erhöhen. Dadurch soll die zum Ausführen

einer Restoreoperation erforderliche Zeit minimiert werden. Wenn Sie für die folgenden Parameter des Befehls **RESTORE DATABASE** nicht explizit Werte angeben, wählen DB2-Datenbankprodukte einen Wert für die Parameter aus:

- **WITH** *anzahl_puffer* **BUFFERS**
- **PARALLELISM** *n*
- **BUFFER** *puffergröße*

Für Restoreoperationen wird immer ein Vielfaches der von der Backup-Operation verwendeten Puffergröße verwendet. Sie können beim Absetzen des Befehls **RESTORE DATABASE** eine Puffergröße angeben, müssen jedoch sicherstellen, dass es sich dabei um ein Vielfaches der Backup-Puffergröße handelt.

Sie können auch eine der folgenden Aktionen ausführen, um die für eine Restoreoperation erforderliche Zeit zu reduzieren:

- Vergrößern Sie die Restorepuffer.

  Die Größe der Restorepuffer muss eine positive ganze Zahl und ein Vielfaches der Backup-Puffergröße sein, die bei der Backup-Operation angegeben wurde. Wird eine nicht korrekte Puffergröße angegeben, erhalten die zugeordneten Puffer die kleinstmögliche Größe.
- Erhöhen Sie die Anzahl der Puffer.

  Der Wert, den Sie angeben, muss ein Vielfaches der Puffergröße sein, die beim Backup verwendet wurde. Andernfalls wird er auf das nächstkleinere Vielfache der beim Backup verwendeten Puffergröße abgerundet.
- Erhöhen Sie den Wert des Parameters **PARALLELISM**.

  Dadurch wird die Anzahl der Puffermanipulatoren erhöht, die verwendet werden, um während der Restoreoperation in die Datenbank zu schreiben.
- Erhöhen Sie den Wert für die Größe des Zwischenspeichers für Dienstprogramme.

  Dies vergrößert den Speicher, der gleichzeitig durch andere Dienstprogramme genutzt werden kann.

Erforderliche Zugriffsrechte und Berechtigungen zur Verwendung von Restore

Sie benötigen die Berechtigung SYSADM, SYSCTRL oder SYSMAINT, um eine *vorhandene* Datenbank mithilfe einer Backupkopie der gesamten Datenbank wiederherstellen zu können. Für das Wiederherstellen in eine *neue* Datenbank ist die Berechtigung SYSADM oder SYSCTRL erforderlich.

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf diese zuzugreifen. Berechtigungsstufen stellen eine Methode dar, um Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zusammenzufassen. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte.

Benutzer können nur auf die Objekte zugreifen, für die sie zugriffsberechtigt sind, d. h., für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Transport von Datenbankschemata

Zum Transport eines Datenbankschemas ist das Backup-Image einer Datenbank erforderlich, das zum Wiederherstellen des Datenbankschemas in einer anderen vorhandenen Datenbank verwendet werden kann.

Beim Transport eines Datenbankschemas werden die Datenbankobjekte in dem transportierten Schema erneut erstellt, sodass sie auf die neue Datenbank verweisen, und die Daten werden in der neuen Datenbank wiederhergestellt.

Ein Datenbankschema muss in seiner Gesamtheit transportiert werden. Wenn ein Tabellenbereich neben dem zu transportierenden Schema noch ein weiteres Schema enthält, müssen alle Datenobjekte aus beiden Schemata transportiert werden. Solche Schemagruppen, die keine Verweise auf andere Datenbankschemata enthalten, werden als *transportierbare Gruppen* bezeichnet. Die Daten in den Tabellenbereichen und logischen Objekten der Schemata einer transportierbaren Gruppe verweisen ausschließlich auf Tabellenbereiche und Schemata in der transportierbaren Gruppe. Beispielsweise sind die zugehörigen Tabellen ausschließlich von anderen Tabellen in der transportierbaren Gruppe abhängig.

Das folgende Diagramm zeigt eine Datenbank mit mehreren Tabellenbereichen und Schemata. In dem Diagramm sind die Tabellenbereiche, auf die die Schemata verweisen, über den Schemata dargestellt. Manche Schemata verweisen auf mehrere Tabellenbereiche und manche Tabellenbereiche werden von mehreren Schemata referenziert.

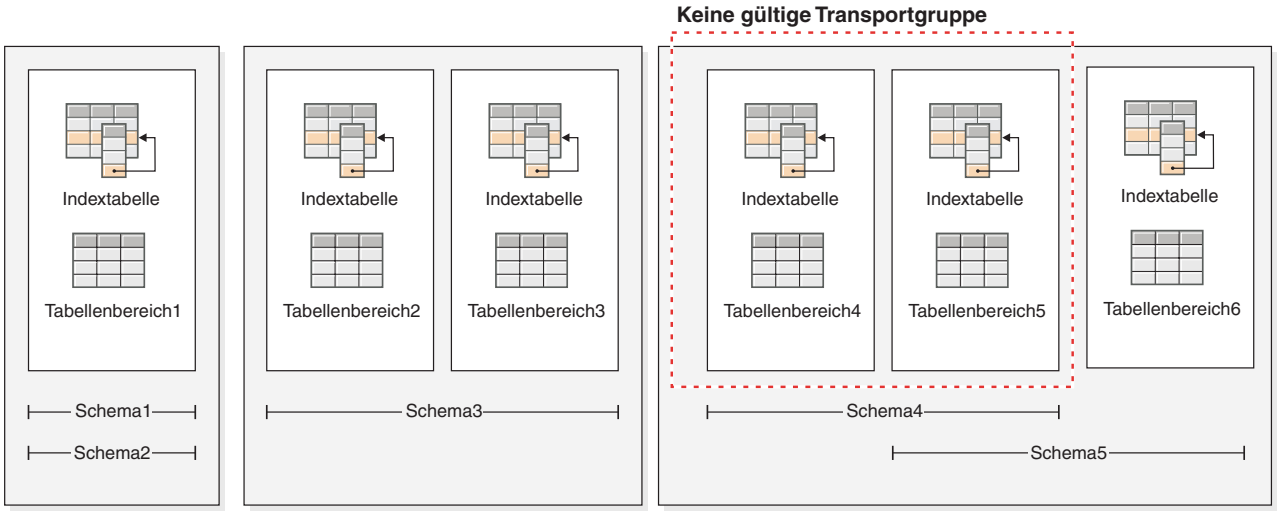


Abbildung 27. Tabellenbereichs- und Schemagruppen

Die folgenden Kombinationen aus Tabellenbereichen und Schemata bilden gültige transportierbare Gruppen:

- Tabellenbereich1 mit Schema1 und Schema2
- Tabellenbereich2 und Tabellenbereich3 mit Schema3
- Tabellenbereich4, Tabellenbereich5 und Tabellenbereich6 mit Schema4 und Schema5
- Eine Kombination aus gültigen transportieren Gruppen bildet ebenfalls eine gültige transportierbare Gruppe:
  - Tabellenbereich1, Tabellenbereich2 und Tabellenbereich3 mit Schema1, Schema2 und Schema3

Die Gruppe 'Tabellenbereich4 und Tabellenbereich5 mit Schema4' ist keine gültige transportierbare Gruppe, weil Verweise zwischen Tabellenbereich5 und Schema5 sowie zwischen Schema5 und Tabellenbereich6 vorhanden sind. Die Gruppe muss Tabellenbereich6 mit Schema5 enthalten, damit sie eine gültige transportierbare Gruppe wird.

Zum Transportieren von Datenbankschemata können Sie den Befehl **RESTORE** mit dem Parameter **TRANSPORT** verwenden.

Beim Transportieren von Datenbankschemata wird eine temporäre Datenbank als Teil der Transportoperation erstellt und benannt. Die Bereitstellungsdatenbank der Transportoperation wird zum Extrahieren logischer Objekte aus dem Backup-Image verwendet, damit diese Objekte in der Zieldatenbank erneut erstellt werden können. Wenn das Backup-Image Protokolle enthält, werden diese mit verwendet, um die Datenbanktransaktionen konsistent zu machen. Anschließend wird das Eigentumsrecht für die transportierten Tabellenbereiche auf die Zieldatenbank übertragen.

Besonderheiten von Datenbankobjekten, die beim Transport von Datenbankschemata neu erstellt werden

Lesen Sie die folgenden Informationen über die Neuerstellung von Datenbankobjekten beim Transport von Datenbankschemata:

Tabelle

Datenbankobjekte	Aspekte beim Transportieren von Schemata
SQL-Routinen (keine externen Routinen, die SQL verwenden)	Eine neue Kopie der SQL-Routine wird in der Zieldatenbank erstellt. Für gespeicherte SQL-Prozeduren wird zusätzlicher Katalogspeicher belegt, weil in der neuen Datenbank eine zusätzliche Kopie des Byte-Codes der gespeicherten Prozedur erstellt wird.
Externe Routinen	Für jede Routine wird ein neuer Katalogeintrag erstellt. Dieser Katalogeintrag verweist auf dieselbe Binärdatei wie die ursprüngliche Quellenroutine. Der Befehl **RESTORE** kopiert die Binärdatei der externen Routine nicht aus dem Quellensystem.
Quellentabellen mit Status, die Zugriffsprobleme verursachen	Die Daten aus Tabellen, die sich bei der Generierung des Backup-Images nicht im normalen Status befanden (z. B. Tabellen mit dem Status 'Überprüfung anstehend' oder 'Laden anstehend'), sind in der Zieldatenbank möglicherweise nicht zugänglich. Um dies zu vermeiden, können Sie die Tabellen vor dem Schematransport in der Quellendatenbank in einen normalen Status versetzen.
Tabellen mit Datenerfassungsattribut	Quellentabellen mit aktivierter Datenerfassung werden mit dem Datenerfassungsattribut in die Zieldatenbank transportiert und protokollieren auch nach dem Transport weiterhin datenbankübergreifende Datenreplikationsinformationen. Replizierte Tabellen extrahieren jedoch keine Informationen aus diesen Tabellen. Nach der Beendigung des Befehls **RESTORE** kann der Benutzer die neue Zieltabelle für die Verwendung als Replikationsquelle registrieren.
Tabellen mit kennsatzbasierter Zugriffssteuerung (Label-Based Access Control, LBAC)	Beim Transportieren von Daten, die durch LBAC geschützt sind, erstellt die Transportoperation die LBAC-Objekte in der Zieldatenbank erneut. Wenn in der Zieldatenbank LBAC-Objekte mit identischen Namen vorhanden sind, schlägt die Transportoperation fehl. Um sicherzustellen, dass der eingeschränkte Datenzugriff nicht beeinträchtigt wird, verwendet die Transportoperation keine bereits vorhandenen LBAC-Objekte in der Zieldatenbank.

Wenn Sie Tabellenbereiche transportieren, wird ein Protokollsatz mit einem speziellen Format in der Zieldatenbank erstellt. Dieses Format kann nicht von früheren DB2-Versionen gelesen werden. Wenn Sie Tabellenbereiche transportieren und dann ein Downgrade zu einer früheren Version von DB2 Version 9.7 Fixpack 2 durchführen, können sie die Zieldatenbank, die die transportierten Tabellenbereiche enthält, nicht wiederherstellen. Sie können die Zieldatenbank auf einen Zeitpunkt vor der Transportoperation aktualisierend wiederherstellen, um sicherzustellen, dass die Zieldatenbank mit früheren DB2-Versionen kompatibel ist.

Wichtig: Wenn die aktualisierende Recovery einer Datenbank einen Protokolleintrag für ein transportiertes Tabellenbereichsschema findet, wird der entsprechende Tabellenbereich offline gesetzt und in den Status "Löschen anstehend" versetzt. Der Grund dafür ist, dass die Datenbank nicht über vollständige Protokolle zu transportierten Tabellenbereichen verfügt, um diese sowie deren Inhalte erneut zu erstellen. Sie können nach Beendigung des Transports einen vollständiges Backup der Zieldatenbank durchführen, damit bei der nachfolgenden aktualisierenden Recoveryoperation der Punkt des Schematransports im Protokolldatenstrom nicht übergangen wird.

Transportierbare Objekte

Wenn Sie Daten von einem Backup-Image in eine Zieldatenbank übertragen, gibt es zwei wesentliche Ergebnisse. Die physischen und logischen Objekte in den Tabellenbereichen, die Sie wiederherstellen, werden in der Zieldatenbank erneut erstellt, und die Tabellenbereichsdefinitionen und Container werden zur Zieldatenbank hinzugefügt.

Die folgenden logischen Objekte werden erneut erstellt:

- Tabellen, erstellte globale temporäre Tabellen und MQTs (Materialized Query Tables)
- Normale und statistische Sichten
- Die folgenden Spaltentypen werden generiert:
  - Ausdruck
  - Identität
  - Zeitmarke für Zeilenänderung
  - Token für Zeilenänderung
- Benutzerdefinierte Funktionen und generierte Funktionen
- Funktionen und Prozeduren (ausgenommen ausführbare Dateien für externe Routinen)
- Benutzerdefinierte Datentypen
- Die folgenden Typen von Integritätsbedingungen:
  - Prüfung
  - Fremdschlüssel
  - Funktionale Abhängigkeit
  - Primär
  - Eindeutigkeit
- Indizes
- Trigger
- Sequenzen
- Objektberechtigungen, Zugriffsrechte, Sicherheit, Zugriffssteuerung und Prüfkonfiguration

- Tabellenstatistik, Profile und Hinweise
- Pakete

Die folgenden Schemakomponenten werden in der Zieldatenbank nicht erstellt:

- Aliasnamen
- Erstellte globale Variablen
- Ausführbare Dateien für externe Routinen
- Funktionszuordnungen und Schablonen
- Hierarchietabellen
- Indexerweiterungen
- Jobs
- Methoden
- Kurznamen
- Externe OLE-DB-Funktionen
- Bereichspartitionierte Tabellen
- Server
- Quellenprozeduren
- Strukturierte Typen
- Systemkataloge
- Typisierte Tabellen und typisierte Sichten
- Nutzungslisten
- Wrapper

Transportbeispiele

Mit dem Befehl **RESTORE DATABASE** und der Option TRANSPORT können Sie eine Gruppe von Tabellenbereichen und SQL-Schemata aus einer Datenbank in eine andere kopieren.

In den folgenden Beispielen wird die Datenbank mit dem Namen ORIGINALDB als Quelle für das Backup-Image verwendet und die Datenbank TARGETDB als Ziel.

Die folgende Abbildung zeigt die Tabellenbereiche und Schemata in der Datenbank ORIGINALDB:

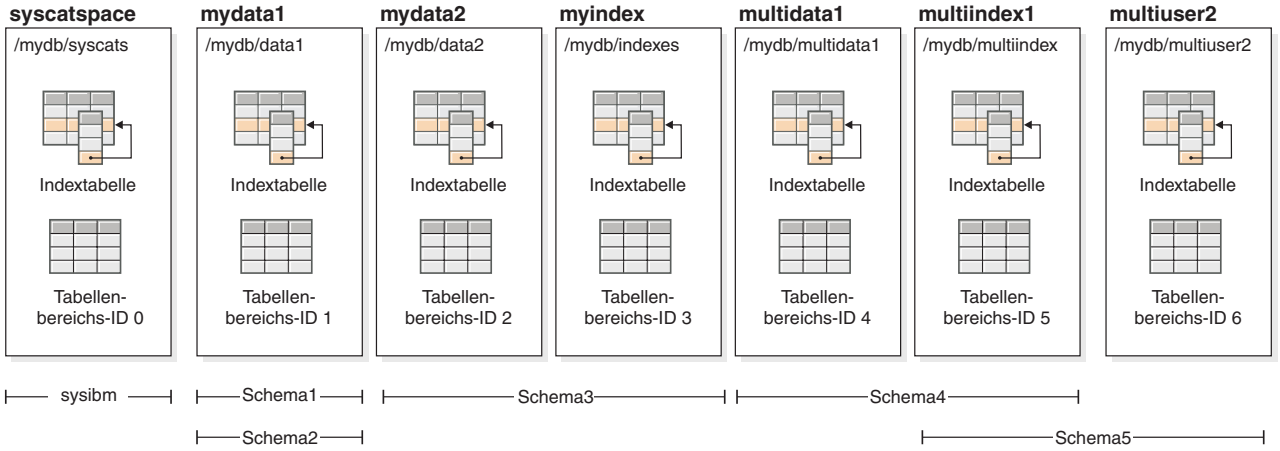


Abbildung 28. Datenbank ORIGINALDB

Die Datenbank ORIGINALDB enthält die folgenden gültigen transportierbaren Gruppen:

- mydata1; schema1 + schema2
- mydata2 + myindex; schema3
- multidata1 + multiindex1 + multiuser2; schema4 + schema5
- Eine Kombination aus gültigen transportierbaren Gruppen ist ebenfalls eine gültige transportierbare Gruppe:
  - mydata1 + mydata2 + myindex; schema1 + schema + schema3

Die folgende Abbildung zeigt die Tabellenbereiche und Schemata der Datenbank TARGETDB:

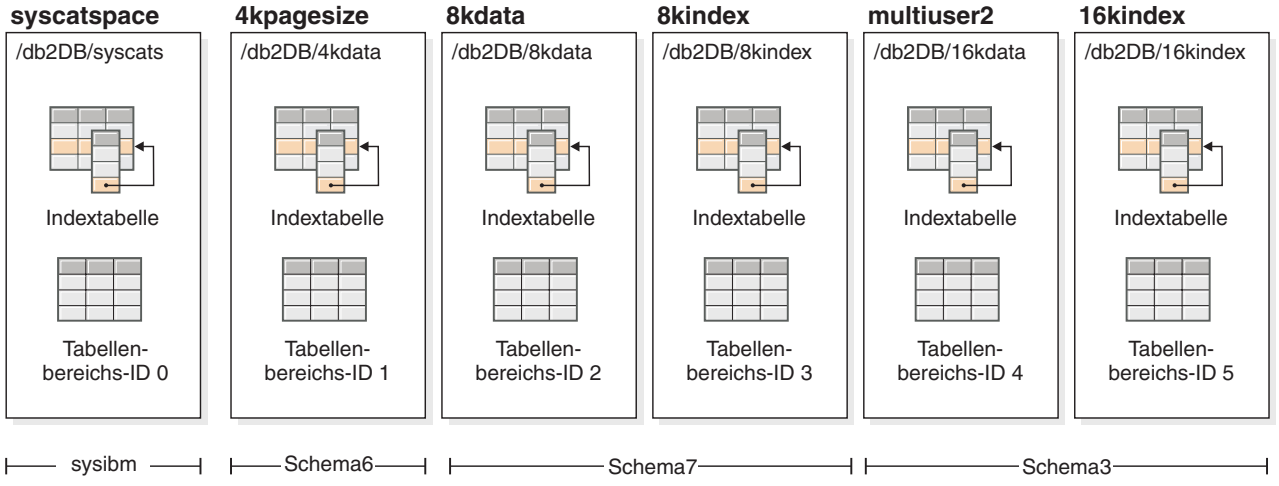


Abbildung 29. Datenbank TARGETDB

Wenn die Quellen- und die Zieldatenbank Schemata mit identischen Schemanamen oder Tabellenbereichsnamen enthalten, können die betreffenden Schemata oder Tabellenbereiche nicht in die Zieldatenbank transportiert werden. Die Transportoperation schlägt fehl, wenn sie Schemata oder Tabellenbereiche enthält, deren Namen mit Schema- oder Tabellenbereichsnamen in der Zieldatenbank identisch sind. Die folgende Gruppierung ist zwar eine gültige transportierbare Gruppe, sie kann jedoch nicht direkt in die Zieldatenbank transportiert werden:

- mydata2 + myindex; schema3 (schema3 ist in der Quellen- und in der Zieldatenbank enthalten)

Wenn ein einzelnes Online-Backup-Image für ORIGINALDB vorhanden ist, das alle Tabellenbereiche der Datenbank enthält, wird dieses Image als Quelle für den Transport verwendet. Dies gilt auch für Backup-Images auf Tabellenbereichsebene.

Sie können die Containerpfade für die zu transportierenden Tabellenbereiche umleiten. Dies ist besonders wichtig, wenn relative Datenbankpfade verwendet wurden.

Beispiele

Beispiel 1: Erfolgreicher Transport von 'schema1' und 'schema2' im Tabellenbereich 'mydata1' in TARGETDB.

```
db2 restore db originaldb tablespace (mydata1) schema(schema1,schema2)
  from <Media_Target_clause> taken at <date-time>
  transport into targetdb redirect
db2 list tablespaces
db2 set tablespace containers for <tablespace ID for mydata1> using (path '/db2DB/data1')

db2 restore db originaldb continue
```

Die resultierende Datenbank TARGETDB enthält den Tabellenbereich 'mydata1' sowie 'schema1' und 'schema2'.

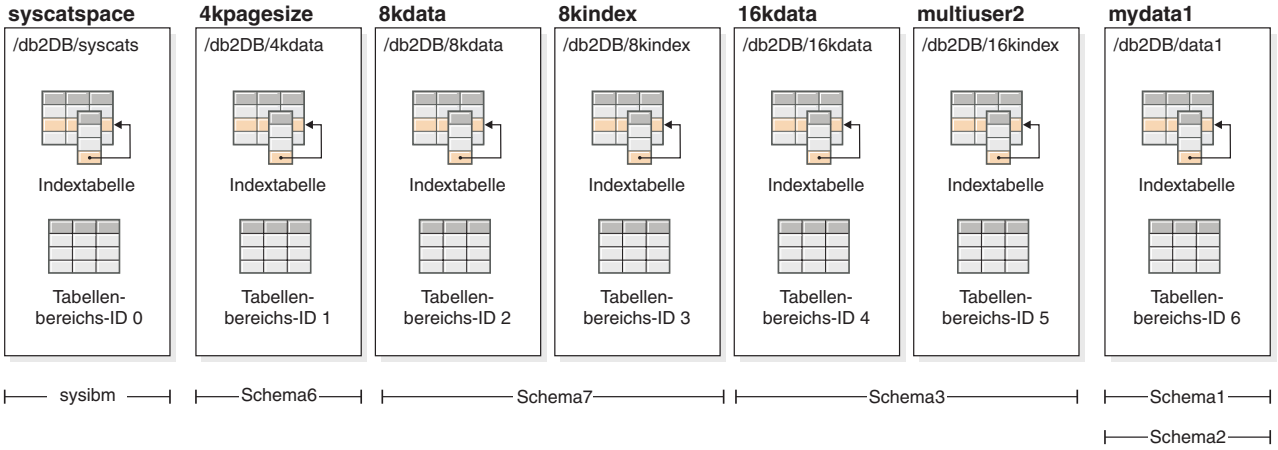


Abbildung 30. Datenbank TARGETDB nach dem Transport

Beispiel 2: Transport des Schemas 'schema3' in den Tabellenbereichen 'mydata2' und 'myindex' in TARGETDB. Sie können ein Schema, das in der Zieldatenbank bereits vorhanden ist, nicht transportieren.

```
db2 restore db originaldb tablespace (mydata2,myindex) schema(schema3)
  transport into targetdb
```

Die Transportoperation schlägt fehl, da das Schema 'schema3' in der Zieldatenbank bereits vorhanden ist. TARGETDB bleibt unverändert. SQLCODE=SQL2590N rc=3.

Beispiel 3: Transport der Schemata 'schema4' und 'schema5' in Tabellenbereichen 'multidata1', 'multiindex1' und 'multiuser2' in Datenbank TARGETDB. Sie können einen Tabellenbereich, der in der Zieldatenbank bereits vorhanden ist, nicht transportieren.

```
db2 restore db originaldb tablespace (multidata1,multiindex1,multiuser2)
  schema(schema4,schema5) transport into targetdb
```

Die Transportoperation schlägt fehl und die Datenbank TARGETDB bleibt unverändert, da Tabellenbereich 'multiuser2' in der Zieldatenbank bereits vorhanden ist. SQLCODE=SQL2590N rc=3.

Beispiel 4: Transport des Tabellenbereichs 'myindex' in TARGETDB. Sie können keine partiellen Schemata transportieren.

```
db2 restore db originaldb tablespace (myindex) schema(schema3)
  transport into targetdb
```

Die Liste der transportierten Tabellenbereiche und Schemata ist keine gültige transportierbare Gruppe. Die Transportoperation schlägt fehl und die Datenbank TARGETDB bleibt unverändert. SQLCODE=SQL2590N rc=1.

Beispiel 5: Restore des Tabellenbereichs 'syscatspace' in TARGETDB durchführen. Es können keine Systemkataloge transportiert werden.

```
db2 restore db originaldb tablespace (syscatspace) schema(sysibm)
  transport into targetdb
```

Die Transportoperation schlägt fehl, da die Systemkataloge nicht transportiert werden können. SQLCODE=SQL2590N rc=4. Sie können benutzerdefinierte Tabellenbereiche transportieren oder für die Systemkataloge ein Restore mit dem Befehl RESTORE DATABASE durchführen, ohne die Transportoption angeben zu müssen.

Beispiel 6: Sie können kein Restore in einer Zieldatenbank durchführen, die auf dem System nicht vorhanden ist.

```
db2 restore db originaldb tablespace (mydata1) schema(schema1,schema2)
  transport into notexists
```

Die Transportoperation schlägt fehl. Tabellenbereiche können nicht in eine Zieldatenbank transportiert werden, die nicht vorhanden ist.

Fehlerbehebung: Transport von Schemata

Wenn in der Quellen- oder Zieldatenbank ein Fehler auftritt, muss die gesamte Wiederherstellungsoperation erneut ausgeführt werden. Alle auftretenden Fehler werden in der Protokolldatei `db2diag` auf dem Zielserver erfasst. Überprüfen Sie das Protokoll **db2diag**, bevor Sie den Befehl **RESTORE** erneut ausführen.

Fehlerbearbeitung

Fehler bei der Wiederherstellung werden auf verschiedene Arten bearbeitet, je nach Typ des kopierten Objekts und Phase der Transportoperation. In bestimmten Situationen (z. B. bei Stromausfall) kann keine vollständige Bereinigung erfolgen.

Die Transportoperation umfasst die folgenden Phasen:

- Erstellen der Bereitstellungsdatenbank
- Wiederherstellen des physischen Tabellenbereichscontainers
- Verarbeitung der aktualisierenden Wiederherstellung
- Schemaprüfung
- Eigentumsübertragung für die Tabellenbereichscontainer
- Schemaneuerstellung in der Zieldatenbank

- Löschen der Bereitstellungsdatenbank (wenn der Parameter **STAGE IN** nicht angegeben ist)

Wenn nach der Schemaneuerstellung Fehler für den Transport physischer Objekte gemeldet werden, schlägt die Wiederherstellungsoperation fehl und es wird ein Fehler zurückgegeben. Alle Objekterstellungsvorgänge in der Zieldatenbank werden rückgängig gemacht und alle intern erstellten Tabellen in der Bereitstellungsdatenbank werden bereinigt. Die Rollback-Operation erfolgt am Ende der Neuerstellungsphase, damit alle auftretenden Fehler in der Protokolldatei **db2diag** aufgezeichnet werden können. Sie können alle Fehlermeldungen überprüfen, bevor Sie den Befehl erneut ausführen.

Die Bereitstellungsdatenbank wird nach erfolgreichem und nach fehlgeschlagenem Transport automatisch gelöscht. Sie wird bei fehlgeschlagenem Transport jedoch nicht gelöscht, wenn der Parameter **STAGE IN** angegeben wurde. Die Bereitstellungsdatenbank muss gelöscht werden, damit der Name der Bereitstellungsdatenbank erneut verwendet werden kann.

Kapitel 14. Aktualisierende Recovery - Übersicht

Es ist nicht möglich, eine Recovery von Transaktionen, die nach der Erstellung des Backup-Images erfolgten, mithilfe der Restore-Tools durchzuführen. Stattdessen können Sie Transaktionen, die seit dem Abschluss des letzten Backup-Befehls erfolgten, mithilfe von Befehlen für die aktualisierende Recovery wiederherstellen. Die Datenbankprotokollierung muss aktiviert werden, damit diese Befehle wirksam sind.

In der einfachsten Form des Befehls **ROLLFORWARD DATABASE** müssen Sie lediglich den Aliasnamen der Datenbank angeben, die Sie aktualisierend wiederherstellen wollen, wie im folgenden Beispiel gezeigt:

```
db2 ROLLFORWARD DB sample
```

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Aktualisierende Recovery von Datenbanken. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Im Folgenden wird eine mögliche Vorgehensweise zur Durchführung einer aktualisierenden Recovery beschrieben:

1. Rufen Sie das Dienstprogramm zur aktualisierenden Recovery ohne die Option **STOP** auf.
2. Rufen Sie das Dienstprogramm zur aktualisierenden Recovery mit der Option **QUERY STATUS** auf.

   Wenn Sie eine Recovery bis zum Ende der Protokolle angeben, kann die Option **QUERY STATUS** angeben, dass eine oder mehrere Protokolldateien fehlen, falls der zurückgegebene Zeitpunkt vor dem erwarteten Zeitpunkt liegt.

   Wenn Sie eine Recovery bis zu einem bestimmten Zeitpunkt angeben, können Sie mit der Option **QUERY STATUS** sicherstellen, dass die aktualisierende Recovery zum richtigen Zeitpunkt beendet wird.
3. Rufen Sie das Dienstprogramm zur aktualisierenden Recovery mit der Option **STOP** auf. Nach Beendigung dieser Operation können keine weiteren Änderungen mehr aktualisierend wiederhergestellt werden.

Eine mögliche Alternative besteht darin, die aktualisierende Recovery folgendermaßen auszuführen:

1. Rufen Sie das Dienstprogramm zur aktualisierenden Recovery mit der Option **AND STOP** auf.
2. Die Notwendigkeit weiterer Schritte hängt vom Ergebnis der aktualisierenden Recoveryoperation ab:
   - Wenn sie erfolgreich ist, wurde die aktualisierende Recovery abgeschlossen. Eine Verbindung zur Datenbank kann hergestellt und die Datenbank selbst genutzt werden. An diesem Punkt können keine weiteren Änderungen mehr aktualisierend wiederhergestellt werden.
   - Wenn Fehler zurückgemeldet werden, führen Sie die erforderlichen Maßnahmen zur Korrektur des Problems durch. Wenn beispielsweise eine Protokolldatei fehlt, suchen Sie die Protokolldatei, oder wenn Abruffehler vorliegen,

stellen Sie sicher, dass die Protokollarchivierung funktioniert. Anschließend rufen Sie das Dienstprogramm zur aktualisierenden Recovery mit der Option **AND STOP** erneut auf.

Eine Datenbank muss erst erfolgreich wiederhergestellt werden (mit dem Restoredienstprogramm), bevor sie aktualisierend wiederhergestellt werden kann. Bei einem Tabellenbereich ist dies jedoch nicht erforderlich. Ein Tabellenbereich kann zeitweilig in den Status *Aktualisierende Recovery anstehend* versetzt werden, erfordert aber keine Restoreoperation, um diesen Status aufzuheben (z. B. nach einem Stromausfall).

Wenn das Dienstprogramm zur aktualisierenden Recovery aufgerufen wird, geschieht Folgendes:

- Wenn sich die Datenbank im Status *Aktualisierende Recovery anstehend* befindet, wird die Datenbank aktualisierend wiederhergestellt. Tabellenbereiche, die aus Backup-Images wiederhergestellt wurden, die nach dem Backup-Image der Datenbank erstellt wurden, und sich momentan im Status "Aktualisierende Recovery anstehend" befinden, werden ebenfalls aktualisierend wiederhergestellt. Tabellenbereiche, die vor dem Backup auf Datenbankebene erstellt und nach der Wiederherstellung des Backups auf Datenbankebene wiederhergestellt wurden, verbleiben im Status "Aktualisierende Recovery anstehend". Zur Wiederherstellung dieser Tabellenbereiche müssen Sie eine nachfolgende aktualisierende Recovery auf Tabellenbereichsebene durchführen.
- Wenn sich die Datenbank *nicht* im Status "Aktualisierende Recovery anstehend" befindet, die Tabellenbereiche in der Datenbank sich jedoch im Status "Aktualisierende Recovery anstehend" *befinden*:
  - Wenn Sie eine Liste mit Tabellenbereichen angeben, werden nur diese Tabellenbereiche aktualisierend wiederhergestellt.
  - Wenn Sie keine Liste mit Tabellenbereichen angeben, werden alle Bereiche aktualisierend wiederhergestellt, die den Status *Aktualisierende Recovery anstehend* haben.

Die aktualisierende Recovery einer Datenbank wird offline durchgeführt. Die Datenbank steht erst dann wieder zur Verfügung, wenn die aktualisierende Recovery erfolgreich beendet wurde. Diese Operation kann jedoch nur beendet werden, wenn beim Aufrufen des Dienstprogramms die Option **STOP** angegeben wurde.

Die aktualisierende Recovery von Tabellenbereichen kann offline durchgeführt werden. Die Datenbank steht erst dann wieder zur Verfügung, wenn die aktualisierende Recovery erfolgreich beendet wurde. Dies geschieht, wenn das Ende der Protokolle erreicht wird oder wenn beim Aufrufen des Dienstprogramms die Option **STOP** angegeben wurde.

Wenn SYSCATSPACE nicht betroffen ist, können Sie die aktualisierende Recovery von Tabellenbereichen auch *online* durchführen. Wenn Sie eine aktualisierende Recovery für einen Tabellenbereich online durchführen, steht dieser während der Operation nicht zur Verfügung. Die anderen Tabellenbereiche in der Datenbank *sind* jedoch verfügbar.

Wenn Sie eine Datenbank erstellen, wird für diese Datenbank zunächst nur die Umlaufprotokollierung aktiviert. Das bedeutet, dass die Protokolle wiederverwendet und nicht gespeichert oder archiviert werden. Bei Verwendung der Umlaufprotokollierung ist eine aktualisierende Recovery nicht möglich. Es kann lediglich eine Recovery nach einem Systemabsturz bzw. eine Versionsrecovery durchgeführt werden. Archivierte Protokolldateien erfassen die Änderungen, die nach dem Erstellen

eines Backup-Images an einer Datenbank vorgenommen werden. Sie können diese Protokollierung (und aktualisierende Recovery) aktivieren, indem Sie den Datenbankkonfigurationsparameter **logarchmeth1** auf einen anderen Wert als den Standardwert OFF setzen. Wenn Sie **logarchmeth1** auf einen Wert ungleich OFF setzen, wird die Datenbank in den Status "Backup anstehend" gesetzt, und Sie müssen ein Offline-Backup der Datenbank ausführen, bevor sie wiederverwendet werden kann.

Anmerkung: Für jede Protokolldatei, die in einer Operation zur aktualisierenden Recovery verwendet wird, wird in der Datei des Recoveryprotokolls ein Eintrag erstellt.
In diesem Beispiel gibt der Befehl Folgendes zurück:

In einer Umgebung mit partitionierten Datenbanken und in einer DB2 pureScale-Umgebung wird für jede Datenbankpartition oder jedes Member diese Statusinformation zurückgegeben:

```
db2 rollforward db mydb to end of logs

                              Status der aktualisierenden Recovery

Aliasname der Eingabedatenbank            = mydb
Anzahl der Member, die Status zurückgaben = 3

Member-ID  Status der aktual.  Nächstfolgendes   Verarbeitete               Zuletzt festgeschriebene
           Wiederherstellung   Protokoll zum Lesen  Protokolldateien          Transaktion
---------- ------------------  -------------------  ------------------------  ----------------------------
0          DB  working         S0000001.LOG         S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC
1          DB  working         S0000010.LOG         S0000000.LOG-S0000009.LOG  2009-05-06-15.28.20.000000 UTC
2          DB  working         S0000005.LOG         S0000000.LOG-S0000004.LOG  2009-05-06-15.27.33.000000 UTC

DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Verwenden der aktualisierenden Recovery

Verwenden Sie den Befehl **ROLLFORWARD DATABASE**, um Transaktionen, die in den Datenbankprotokolldateien aufgezeichnet wurden, auf ein mit RESTORE wiederhergestelltes Backup-Image einer Datenbank bzw. eines Tabellenbereichs anzuwenden.

Vorbereitende Schritte

Es sollte noch keine Verbindung zu der Datenbank bestehen, die aktualisierend wiederhergestellt werden soll. Das Dienstprogramm zur aktualisierenden Recovery stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der aktualisierenden Recovery beendet wird.

Informationen zu diesem Vorgang

Tabellenbereiche dürfen nicht wiederhergestellt werden, ohne dass eine momentan ausgeführte aktualisierende Recovery abgebrochen wird. Andernfalls erhalten Sie möglicherweise eine Tabellenbereichsgruppe, in der sich einige Tabellenbereiche im Status 'Aktualisierende Recovery läuft' befinden und andere im Status 'Aktualisierende Recovery anstehend'. Eine aktive Operation zur aktualisierenden Recovery wirkt sich nur auf Tabellenbereiche aus, die sich im Status 'Aktualisierende Recovery läuft' befinden.

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln.

Für das Dienstprogramm zur aktualisierenden Recovery gelten die folgenden Einschränkungen:

- Sie können jeweils nur eine Operation zur aktualisierenden Recovery aufrufen. Wenn Sie mehrere Tabellenbereiche wiederherstellen wollen, können Sie alle Bereiche in derselben Operation angeben.
- Wenn Sie nach der letzten Backup-Operation einen Tabellenbereich umbenannt haben, müssen Sie bei der aktualisierenden Recovery sicherstellen, dass der neue Name verwendet wird. Der vorherige Tabellenbereichsname wird nicht erkannt.
- Sie können eine laufende Operation zur aktualisierenden Recovery nicht abbrechen. Es können nur Operationen zur aktualisierenden Recovery abgebrochen werden, die zwar beendet wurden, für die jedoch der Parameter **STOP** nicht angegeben wurde, oder Operationen, die vor der erfolgreichen Beendigung fehlgeschlagen sind.
- Sie können eine aktualisierende Recovery eines Tabellenbereichs nicht bis zu einem bestimmten Zeitpunkt *fortsetzen*, wenn die von Ihnen angegebene Zeitmarke vor der vorherigen Zeitmarke liegt. Wenn kein bestimmter Zeitpunkt angegeben wird, wird der vorherige verwendet. Sie können eine aktualisierende Recovery ausgeben, die an einem bestimmten Zeitpunkt dadurch beendet wird, dass Sie lediglich die Option STOP angeben. Dies ist jedoch nur dann zulässig, wenn die betroffenen Tabellenbereiche zuvor alle aus demselben Offline-Backup-Image wiederhergestellt wurden. In diesem Fall ist keine Protokollverarbeitung erforderlich. Wenn Sie für eine Liste anderer Tabellenbereiche eine weitere Operation zur aktualisierenden Recovery starten, bevor die laufende Operation beendet oder abgebrochen wurde, wird eine Fehlernachricht (SQL4908) zurückgegeben. Rufen Sie den Befehl **LIST TABLESPACES** für alle Datenbankpartitionen auf (oder verwenden Sie die Tabellenfunktion MON_GET_TABLESPACE), um festzustellen, für welche Tabellenbereiche momentan eine aktualisierende Recovery durchgeführt wird (Status 'Aktualisierende Recovery läuft') und welche Tabellenbereiche für die aktualisierende Recovery bereit sind (Status 'Aktualisierende Recovery anstehend'). Sie haben folgende Möglichkeiten:
  - Beenden Sie die aktuelle Operation zur aktualisierenden Recovery für alle Tabellenbereiche.
  - Beenden Sie die aktuelle Operation zur aktualisierenden Recovery für eine Untergruppe von Tabellenbereichen. (Dies ist evtl. nicht möglich, wenn die Operation zur aktualisierenden Recovery bis zu einem bestimmten Zeitpunkt ausgeführt werden soll, wofür die Beteiligung aller Datenbankpartitionen erforderlich ist.)
  - Brechen Sie die aktuelle Operation zur aktualisierenden Recovery ab.
- In einer Umgebung mit partitionierten Datenbanken muss das Dienstprogramm zur aktualisierenden Recovery von der Katalogpartition der Datenbank aus aufgerufen werden.
- Die punktuelle aktualisierende Recovery eines Tabellenbereichs wurde in Clients der DB2 Version 9.1 eingeführt. Sie sollten für sämtliche Clients ein Upgrade auf Version 10.1 ausführen, damit für einen Tabellenbereich eine punktuelle aktualisierende Recovery durchgeführt werden kann.
- Für Protokolle aus einem früheren Release kann keine aktualisierende Recovery durchgeführt werden.

Vorgehensweise

Verwenden Sie eine der beiden folgenden Möglichkeiten, um das Dienstprogramm zur aktualisierenden Recovery aufzurufen:

- Befehl **ROLLFORWARD DATABASE**
- Anwendungsprogrammierschnittstelle (API) db2Rollforward

- Öffnen Sie den Taskassistenten in IBM Data Studio für den Befehl **ROLLFORWARD DATABASE**.

Beispiel

Es folgt ein Beispiel für den Befehl **ROLLFORWARD DATABASE**, der über den CLP abgesetzt wird:

```
db2 rollforward db sample to end of logs and stop
```

Fortsetzen einer gestoppten oder fehlgeschlagenen aktualisierenden Recovery

Sie können eine aktualisierende Recovery fortsetzen, falls eine der folgenden Bedingungen eingetreten ist: die vorherige aktualisierende Recovery ist fehlgeschlagen, die vorherige aktualisierende Recovery wurde unterbrochen oder die vorherige aktualisierende Recovery wurde zwar beendet, der Befehl gab jedoch nicht die Meldung STOP oder COMPLETE zurück.

Vorbereitende Schritte

Es sollte noch keine Verbindung zu der Datenbank bestehen, die aktualisierend wiederhergestellt werden soll. Das Dienstprogramm zur aktualisierenden Recovery stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der aktualisierenden Recovery beendet wird.

Informationen zu diesem Vorgang

Zum Fortsetzen einer aktualisierenden Recovery können Sie zum einen ein *erzwungenes Stoppen* verwenden. Hierzu geben Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP**, jedoch unter Angabe von **TO** aus. Bei einem erzwungenen Stoppen ignoriert das Dienstprogramm zur aktualisierenden Recovery bestimmte Fehler, falls dies keine Beeinträchtigungen verursacht. Als ignorierbare Fehler gelten beispielsweise fehlende Protokolldateien, Kontrollsummenfehler, Protokollkettenfehler und ein ausbleibendes Erreichen des Zeitpunktes. Falls DB2 feststellt, dass das Stoppen ohne Beeinträchtigungen ausgeführt werden kann, durchläuft die aktualisierende Recovery eine Aufhebungsphase und die Datenbank ist für normale Verbindungen verfügbar. Stellt DB2 fest, dass das Stoppen nicht ohne Weiteres möglich ist, schlägt die aktualisierende Recovery fehl und die Datenbank verbleibt im Status 'Aktualisierende Recovery anstehend'.

Einschränkungen

Wenn Sie eine aktualisierende Recovery fortsetzen, die bis zu einem bestimmten Zeitpunkt durchgeführt werden sollte, muss die neue aktualisierende Recovery eine der folgenden Operationen sein:

- Aktualisierende Recovery bis zum selben Zeitpunkt
- Aktualisierende Recovery bis zu einem späteren Zeitpunkt
- Aktualisierende Recovery bis zum Ende der Protokolle
- Aktualisierende Recovery mit Option STOP oder COMPLETE, jedoch ohne Zeitangabe, Option END OF LOGS oder Option END OF BACKUP

Vorgehensweise

Gehen Sie folgendermaßen vor, um eine aktualisierende Recovery fortzusetzen:

Ergebnisse

Beispiel

Nächste Schritte

Aktualisierende Recovery von Änderungen in einem Tabellenbereich

Wenn die Datenbank zur aktualisierenden Recovery aktiviert ist, können Sie Backups, Restores und aktualisierende Recoverys nicht nur für die gesamte Datenbank, sondern auch für Tabellenbereiche durchführen.

Sie können eine aktualisierende Recovery von Änderungen in einem Tabellenbereich unabhängig von anderen Tabellenbereichen in der Datenbank durchführen, oder Sie können eine aktualisierende Recovery von Änderungen für alle Tabellenbereich gleichzeitig durchführen.

Es kann sinnvoll sein, eine Recoverystrategie für einzelne Tabellenbereiche zu implementieren, da sich dadurch möglicherweise Zeit einsparen lässt: Eine Recovery eines Teils der Datenbank benötigt weniger Zeit als eine Recovery der gesamten Datenbank.

Wenn zum Beispiel eine Festplatte fehlerhaft ist und nur einen Tabellenbereich enthält, kann der Tabellenbereich von einem Backup wiederhergestellt und aktualisierend wiederhergestellt werden, ohne dass die gesamte Datenbank wiederhergestellt werden muss und ohne den Benutzerzugriff auf die übrigen Teile der Datenbank zu beeinträchtigen. Wenn der beschädigte Tabellenbereich jedoch die Systemkatalogtabellen enthält, können Sie keine Verbindung zur Datenbank herstellen. (Der Tabellenbereich mit den Systemkatalogtabellen kann unabhängig von der Datenbank wiederhergestellt werden, wenn für diesen Tabellenbereich ein Backup auf Tabellenbereichsebene verfügbar ist.) Außerdem bieten Backups auf Tabellenbereichsebene die Möglichkeit, kritische Teile der Datenbank häufiger als andere Teile zu sichern, und sind weniger zeitintensiv als Backups der gesamten Datenbank.

Nach dem Restore eines Tabellenbereichs befindet sich dieser immer im Status *Aktualisierende Recovery anstehend*. Um den Tabellenbereich verwendbar zu machen, müssen Sie eine aktualisierende Recovery für ihn durchführen. In den meisten Fällen haben Sie dabei die Möglichkeit, die aktualisierende Recovery bis zum Ende der Protokolldateien oder bis zu einem bestimmten Zeitpunkt auszuführen. Eine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt kann jedoch nicht für Tabellenbereiche ausgeführt werden, die Systemkatalogtabellen enthalten. Diese Tabellen müssen bis zum Ende der Protokolle aktualisierend wiederhergestellt werden, um sicherzustellen, dass alle Tabellenbereiche in der Datenbank konsistent bleiben.

Stellen Sie sicher, dass die Registrierdatenbankvariable **DB2_COLLECT_TS_REC_INFO** auf ON (Standardeinstellung) gesetzt ist, wenn Sie die Protokolldateien überspringen möchten, die bekanntermaßen keine Protokollsätze enthalten, die den Tabellenbereich betreffen. Die Registrierdatenbankvariable muss vor der Erstellung und Verwendung der Protokolldateien gesetzt werden, um sicherzustellen, dass die zum Überspringen der Protokolldateien erforderlichen Informationen erfasst werden. Wenn **DB2_COLLECT_TS_REC_INFO** auf OFF gesetzt ist, verarbeitet DB2 alle Protokolldateien, auch wenn sie keine Protokollsätze enthalten, die den Tabellenbereich betreffen, wenn dieser Tabellenbereich aktualisierend wiederherstellt wird.

Anmerkung: Das Überspringen von Protokollen wird einer DB2 pureScale-Umgebung nicht unterstützt.

Die im Datenbankverzeichnis vorhandene Protokolldatei der Tabellenbereichsänderungen (DB2TSCHG.HIS) enthält Informationen dazu, welche Protokolle für die einzelnen Tabellenbereiche verarbeitet werden sollen. Sie können den Inhalt dieser Datei mithilfe des Dienstprogramms **db2logsForRfwd** anzeigen und mit dem Befehl **PRUNE HISTORY** Einträge aus der Datei löschen. Während einer Restoreoperation für eine Datenbank wird die Datei DB2TSCHG.HIS aus dem Backup-Image wiederhergestellt und während der aktualisierenden Recovery aktualisiert. Falls für eine Protokolldatei keine Informationen verfügbar sind, wird die Datei behandelt, als ob sie für die Recovery aller Tabellenbereiche erforderlich wäre.

Da die Informationen für jede Protokolldatei nach der Inaktivierung des Protokolls auf Platte geschrieben werden, können diese Informationen infolge eines Absturzes verloren gehen. Wenn eine Recoveryoperation in der Mitte einer Protokolldatei beginnt, wird daher zur Kompensation das gesamte Protokoll behandelt, als ob es Änderungen an allen Tabellenbereichen im System enthielte. Anschließend werden die aktiven Protokolle verarbeitet, und die Informationen zu diesen Protokollen werden erneut erstellt. Falls Informationen für ältere Protokolle oder archivierte Protokolldateien bei einem Absturz verloren gehen und in der Datendatei keine Informationen für die Protokolle vorhanden sind, werden die Protokolldateien während der Recoveryoperation für den Tabellenbereich behandelt, als ob sie Änderungen für alle Tabellenbereiche enthielten.

Verwenden Sie vor der aktualisierenden Recovery eines Tabellenbereichs die Tabellenfunktion MON_GET_TABLESPACE zur Ermittlung des *Mindestrecoveryzeitpunkts*. Dies ist der früheste Zeitpunkt, bis zu dem der Tabellenbereich aktualisierend wiederhergestellt werden kann. Der Mindestrecoveryzeitpunkt wird aktualisiert, wenn DDL-Anweisungen (DDL - Datendefinitionssprache) für den Tabellenbereich oder für Tabellen in diesem Tabellenbereich ausgeführt werden. Der Tabellenbereich muss mindestens bis zum Mindestrecoveryzeitpunkt aktualisierend wiederhergestellt werden, um mit den Informationen in den Systemkatalogtabellen synchron zu sein. Wenn Sie mehr als einen Tabellenbereich wiederherstellen, müssen die Tabellenbereiche bis zum frühesten Mindestrecoveryzeitpunkt für alle Tabellenbereiche aktualisierend wiederhergestellt werden. Sie können einen Tabellenbereich nicht bis zu einem Zeitpunkt aktualisierend wiederherstellen, der vor der Zeitmarke des Backups liegt. In einer Umgebung mit partitionierten Datenbanken müssen die Tabellenbereiche bis zum frühesten Mindestrecoveryzeitpunkt für alle Tabellenbereiche auf allen Datenbankpartitionen aktualisierend wiederhergestellt werden.

Wenn Sie Tabellenbereiche bis zu einem bestimmten Zeitpunkt aktualisierend wiederherstellen und eine Tabelle in mehreren Tabellenbereichen enthalten ist, müssen alle Tabellenbereiche, die die Tabelle enthalten, gleichzeitig aktualisierend wiederhergestellt werden. Wenn zum Beispiel die Tabellendaten in einem Tabellenbereich gespeichert sind und der Index für die Tabelle sich in einem anderen Tabellenbereich befindet, müssen beide Tabellenbereiche gleichzeitig bis zum selben Zeitpunkt aktualisierend wiederhergestellt werden.

Wenn die Daten und langen Objekte einer Tabelle in getrennten Tabellenbereichen gespeichert sind und die LOB-Daten reorganisiert wurden, müssen die Tabellenbereiche sowohl für die Daten als auch für die langen Objekte gemeinsam von einem Backup wiederhergestellt und aktualisierend wiederhergestellt werden. Es ist ratsam, nach dem Reorganisieren der Tabelle ein Backup der betroffenen Tabellenbereiche zu erstellen.

Angenommen, Sie wollen einen Tabellenbereich bis zu einem bestimmten Zeitpunkt aktualisierend wiederherstellen, und eine Tabelle im Tabellenbereich entspricht einem der beiden folgenden Typen:

- Eine zugrunde liegende Tabelle für eine MQT (Materialized Query Table - gespeicherte Abfragetabelle) oder Zwischenspeichertabelle, die sich in einem anderen Tabellenbereich befindet
- Eine MQT oder Zwischenspeichertabelle für eine Tabelle, die sich in einem anderen Tabellenbereich befindet

In diesem Fall müssen Sie beide Tabellenbereiche bis zum selben Zeitpunkt aktualisierend wiederherstellen. Wenn Sie dies nicht tun, wird die MQT oder Zwischenspeichertabelle am Ende der aktualisierenden Recovery in den Status *Festlegen der Integrität anstehend* versetzt. Die MQT muss vollständig aktualisiert werden, und die Zwischenspeichertabelle wird als unvollständig markiert.

Wenn Sie einen Tabellenbereich bis zu einem bestimmten Zeitpunkt aktualisierend wiederherstellen wollen und eine Tabelle in dem Tabellenbereich an einer referenziellen Integritätsbeziehung mit einer anderen Tabelle beteiligt ist, die in einem anderen Tabellenbereich enthalten ist, sollten Sie beide Tabellenbereiche gleichzeitig bis zum selben Zeitpunkt aktualisierend wiederherstellen. Wenn Sie dies nicht tun, wird die untergeordnete Tabelle in der referenziellen Integritätsbeziehung am Ende der aktualisierenden Recovery in den Status *Festlegen der Integrität anstehend* versetzt. Wenn die untergeordnete Tabelle später auf ungültige Integritätsbedingungen hin überprüft wird, ist eine Überprüfung der gesamten Tabelle erforderlich. Wenn eine der folgenden Tabellen existiert, werden sie ebenfalls zusammen mit der untergeordneten Tabelle in den Status 'Festlegen der Integrität anstehend' versetzt:

- Jede untergeordnete MQT für die untergeordnete Tabelle
- Jede untergeordnete Zwischenspeichertabelle für die untergeordnete Tabelle
- Jede untergeordnete Fremdschlüsseltabelle der untergeordneten Tabelle

Wenn Sie diese Tabellen aus dem Status *Festlegen der Integrität anstehend* herausnehmen wollen, ist eine vollständige Integritätsverarbeitung erforderlich. Wenn Sie beide Tabellenbereiche zur selben Zeit aktualisierend wiederherstellen, bleibt die Integritätsbedingung am Ende der aktualisierenden Recovery bis zu einem bestimmten Zeitpunkt aktiv.

Stellen Sie sicher, dass die aktualisierende Recovery von Tabellenbereichen bis zu einem bestimmten Zeitpunkt nicht dazu führt, dass eine Transaktion in einigen Tabellenbereichen rückgängig gemacht und in anderen festgeschrieben wird. Dies kann in den folgenden Fällen geschehen:

- Eine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt wird für eine Untergruppe der Tabellenbereiche durchgeführt, die durch eine Transaktion aktualisiert wurden, und dieser Zeitpunkt liegt vor dem Zeitpunkt, an dem die Transaktion festgeschrieben wurde.
- Eine Tabelle, die in dem Tabellenbereich enthalten ist, der bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt wird, hat einen zugeordneten Trigger oder wird von einem Trigger aktualisiert, der auf andere Tabellenbereiche als den zugreift, der aktualisierend wiederhergestellt wird.

Sie sollten einen geeigneten Zeitpunkt suchen, der dies verhindert.

Sie können den Befehl **QUIESCE TABLESPACES FOR TABLE** absetzen, um einen transaktionskonsistenten Zeitpunkt für die aktualisierende Recovery von Tabellenbereichen zu erstellen. Diese Wartezeitanforderung (im Modus SHARE, INTENT TO UPDATE oder EXCLUSIVE) wartet durch Sperrung, bis alle aktiven Transaktionen für diese Tabellenbereiche beendet wurden, und blockiert neue Anforderungen.

Wenn die Wartezeitanforderung bestätigt wird, befinden sich die Tabellenbereiche in einem konsistenten Status. Sie können in der Datei des Recoveryprotokolls nach Wartezeitpunkten suchen und prüfen, ob sie nach dem Mindestrecoveryzeitpunkt liegen, um einen geeigneten Zeitpunkt für den Stopp der aktualisierenden Recovery festzulegen.

Nach Beendigung der aktualisierenden Recovery bis zu einem bestimmten Zeitpunkt wird der Tabellenbereich wieder in den Status *Backup anstehend* versetzt. Sie müssen ein Backup des Tabellenbereichs erstellen, weil alle Aktualisierungen für den Zeitraum zwischen dem Zeitpunkt, bis zu dem die aktualisierende Recovery erfolgte, und dem aktuellen Zeitpunkt entfernt wurden. Sie können den Tabellenbereich nicht mehr von einem vorherigen Backup der Datenbank bzw. des Tabellenbereichs bis zum aktuellen Zeitpunkt aktualisierend wiederherstellen. Das folgende Beispiel zeigt, warum das Backup des Tabellenbereichs erforderlich ist und wie es verwendet wird. (Um den Tabellenbereich verfügbar zu machen, können Sie entweder die gesamte Datenbank sichern, oder nur den Tabellenbereich, der den Status *Backup anstehend* hat, oder eine Gruppe von Tabellenbereichen, die diesen letztgenannten Tabellenbereich enthält.)

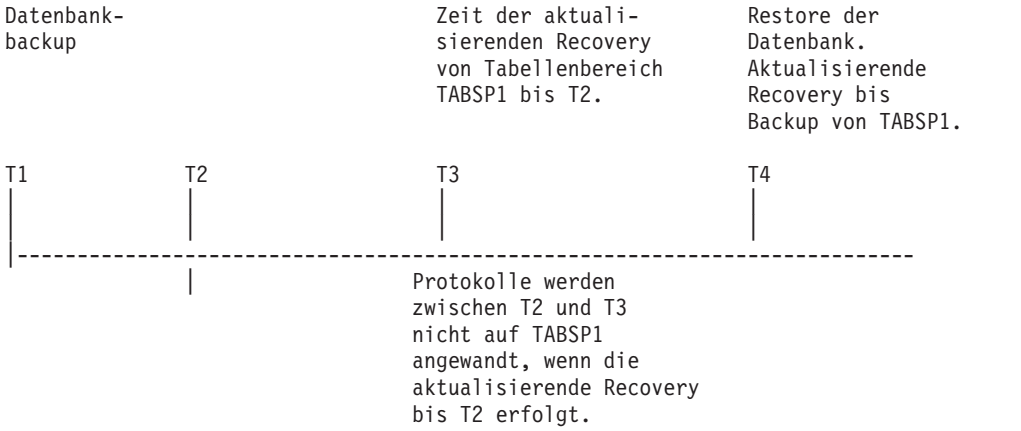


Abbildung 31. Backup-Anforderung für Tabellenbereiche

Im vorstehenden Beispiel wird die Datenbank zum Zeitpunkt T1 gesichert. Anschließend erfolgt für den Tabellenbereich TABSP1 zum Zeitpunkt T3 eine aktualisierende Recovery bis zu einem bestimmten Zeitpunkt (T2). Der Tabellenbereich wird nach Zeitpunkt T3 gesichert. Da sich der Tabellenbereich im Status *Backup anstehend* befindet, muss ein Backup-Image erstellt werden. Die Zeitmarke des Backup-Images des Tabellenbereichs weist eine Zeit nach T3 aus, während sich der Tabellenbereich in dem Status von Zeitpunkt T2 befindet. Die Protokollsätze für den Zeitraum zwischen T2 und T3 werden auf TABSP1 nicht angewendet. Die Datenbank wird unter Verwendung des bei T1 erstellten Backup-Images zum Zeitpunkt T4 wiederhergestellt und bis zum Ende der Protokolle aktualisierend wiederhergestellt. Der Tabellenbereich TABSP1 wird zum Zeitpunkt T3 in den Status *Restore anstehend* versetzt, da der Datenbankmanager annimmt, dass zwischen T3 und T4 Operationen an TABSP1 ausgeführt wurden, ohne dass die Protokolländerungen zwischen T2 und T3 auf den Tabellenbereich angewendet wurden. Wären diese Protokolländerungen tatsächlich als Teil der aktualisierenden Recovery der Datenbank angewendet worden, wäre diese Annahme falsch. Das erforderliche Backup eines Tabellenbereichs, das nach der aktualisierenden Recovery bis zu einem bestimmten Zeitpunkt erstellt werden muss, ermöglicht es, diesen Tabellenbereich bis nach dem Zeitpunkt einer vorherigen aktualisierenden Recovery (in diesem Beispiel T3) aktualisierend wiederherzustellen.

Wenn Sie den Tabellenbereich TABSP1 beispielsweise bis zum Zeitpunkt T4 wiederherstellen wollen, würden Sie den Tabellenbereich von einem Backup wiederherstellen, das nach T3 erstellt wurde (entweder das erforderliche Backup oder ein anderes), und anschließend den Tabellenbereich TABSP1 bis zum Ende der Protokolle aktualisierend wiederherstellen.

Im vorangegangenen Beispiel wäre die effizienteste Vorgehensweise zum Restore des Tabellenbereichs bis zum Zeitpunkt T4 die Ausführung der erforderlichen Schritte in folgender Reihenfolge:

1. Stellen Sie die Datenbank von einem Backup wieder her.
2. Stellen Sie den Tabellenbereich von einem Backup wieder her.
3. Stellen Sie die Datenbank aktualisierend wieder her.

Da Sie den Tabellenbereich vor der aktualisierenden Recovery der Datenbank von einem Backup wiederherstellen, werden keine Ressourcen zum Anwenden der Protokollsätze auf den Tabellenbereich verwendet, wenn die Datenbank aktualisierend wiederhergestellt wird.

Wenn Sie das Backup-Image von TABSP1 für einen Zeitpunkt nach T3 nicht mehr finden können oder den Tabellenbereich TABSP1 auf einem Stand vor oder bis T3 wiederherstellen wollen, haben Sie folgende Möglichkeiten:

- Stellen Sie den Tabellenbereich bis zum Zeitpunkt T3 aktualisierend wieder her. Sie brauchen den Tabellenbereich nicht erneut wiederherzustellen, weil er vom Datenbank-Backup wiederhergestellt wurde.
- Stellen Sie den Tabellenbereich erneut von dem Datenbank-Backup-Image wieder her, das Sie zum Zeitpunkt T1 erstellt haben, und stellen Sie anschließend den Tabellenbereich bis zu einem Zeitpunkt vor T3 aktualisierend wieder her.
- Löschen Sie den Tabellenbereich.

In einer Umgebung mit partitionierten Datenbanken müssen Sie Folgendes beachten:

- Sie müssen alle Teile des Tabellenbereichs bis zum selben Zeitpunkt gleichzeitig aktualisierend wiederherstellen. Dadurch wird sichergestellt, dass der Tabellenbereich datenbankpartitionsübergreifend konsistent ist.
- Wenn sich einige Datenbankpartitionen im Status *Aktualisierende Recovery anstehend* befinden und in anderen Datenbankpartitionen einige Tabellenbereiche ebenfalls diesen Status haben (die Datenbankpartitionen selbst jedoch nicht), müssen Sie zuerst die Datenbankpartitionen und anschließend die Tabellenbereiche aktualisierend wiederherstellen.
- Wenn Sie einen Tabellenbereich bis zum Ende der Protokolle aktualisierend wiederherstellen wollen, müssen Sie ihn nicht in jeder Datenbankpartition wiederherstellen, sondern nur in den Partitionen, für die eine Recovery erforderlich ist. Wenn Sie einen Tabellenbereich jedoch bis zu einem bestimmten Zeitpunkt aktualisierend wiederherstellen wollen, müssen Sie ihn in jeder Datenbankpartition wiederherstellen.

In einer Datenbank mit partitionierten Tabellen:

- Wenn Sie einen Tabellenbereich, der einen Teil einer partitionierten Tabelle enthält, aktualisierend bis zu einem bestimmten Zeitpunkt wiederherstellen, müssen Sie auch alle anderen Tabellenbereiche, in denen sich die Tabelle befindet, bis zu demselben Zeitpunkt aktualisierend wiederherstellen. Die aktualisierende Recovery eines einzelnen Tabellenbereichs, der einen Teil einer partitionierten Tabelle enthält, bis zum Ende der Protokolle ist jedoch zulässig. Wenn eine partitionierte

Tabelle (mit ATTACH) zugeordnete, (mit DETACH) getrennte oder (mit DROP) gelöschte Datenpartitionen enthält, muss die aktualisierende Recovery bis zu einem bestimmten Zeitpunkt auch alle Tabellenbereiche für diese Datenpartitionen berücksichtigen. Um zu bestimmen, ob eine partitionierte Tabelle zugeordnete, getrennte oder gelöschte Datenpartitionen enthält, fragen Sie die Katalogsicht SYSCAT.DATAPARTITIONS ab.

Aktualisierende Recovery für eine Datenbank in einer DB2 pureScale-Umgebung

In einer DB2 pureScale-Umgebung besitzt jedes Member einen eigenen Protokolldatenstrom. Damit der Befehl **ROLLFORWARD DATABASE** erfolgreich ausgeführt werden kann, werden jedoch die Protokolldatenströme von allen Membern benötigt.

Während einer aktualisierenden Recovery für eine Datenbank werden Protokollsätze aus allen Protokolldatenströmen zusammengeführt und angewendet, um die Datenbank konsistent zu machen. Der Zeitpunkt, den Sie im Befehl **ROLLFORWARD DATABASE** angeben, bezieht sich auf den zusammengeführten Protokolldatenstrom. Um eine Datenbank in einem konsistenten Zustand wiederherzustellen, muss der angegebene Zeitpunkt nach der *Mindestrecoveryzeit* (Minimum Recovery Time - MRT) liegen. Die Mindestrecoveryzeit ist der frühestmögliche Zeitpunkt während einer aktualisierenden Recovery, an dem Objekte, die im Datenbankkatalog aufgelistet sind, mit den Objekten übereinstimmen, die physisch auf Platte vorhanden sind. Falls Sie als Basis für den Restore ein Image verwenden, das während einer Online-Backup-Operation erstellt wurde, muss der angegebene Zeitpunkt für die aktualisierende Recovery nach dem Zeitpunkt liegen, an dem die Online-Backup-Operation abgeschlossen war. Dies gewährleistet die Konsistenz der Datenbank.

Der angegebene Zeitpunkt für die nachfolgende aktualisierende Recovery muss größer-gleich der Mindestrecoveryzeit im zusammengeführten Protokolldatenstrom sein. Andernfalls schlägt die aktualisierende Recovery fehl (Nachricht SQL1276N) und die Zeitmarke der Mindestrecoveryzeit wird mit der Fehlernachricht zurückgegeben. Alternativ können Sie die Option END OF BACKUP verwenden, um automatisch eine aktualisierende Recovery zur Mindestrecoveryzeit durchzuführen.

Es empfiehlt sich, die Systemzeiten der Member zu synchronisieren. Unter Umständen ist es jedoch nicht möglich, dass diese jederzeit synchronisiert sind. Dies kann dazu führen, dass Protokollsätze dieselbe Zeitmarke aufweisen und dass Protokolldatenströme mit Protokollsätzen zusammengeführt werden, von denen die Reihenfolge der Zeitmarken scheinbar nicht eingehalten wird. In einer DB2 pureScale-Umgebung wird eine zeitpunktgesteuerte aktualisierende Recovery für eine Datenbank gestoppt, sobald der erste Protokollsatz gefunden wird, dessen Zeitmarke größer als die angegebene Zeitmarke aus einem der Protokolldatenströme ist, und der Protokollsatz verarbeitet wurde, der der Mindestrecoveryzeit für die Datenbank entspricht.

Nach einer unvollständigen oder unterbrochenen aktualisierenden Recovery verbleibt die Datenbank im Status 'Aktualisierende Recovery anstehend'. Setzen Sie in einem solchen Fall einen weiteren Befehl **ROLLFORWARD DATABASE** ab. In einer DB2 pureScale-Umgebung können nachfolgende Befehle **ROLLFORWARD DATABASE** für dasselbe oder für ein anderes Member ausgeführt werden.

Falls Sie in einer DB2 pureScale-Umgebung eine Restoreoperation für eine Datenbank in einer neuen Datenbank ausführen und hierzu ein online erstelltes Daten-

bank-Backup-Image verwenden wollen, richtet sich das korrekte Verfahren danach, ob alle Protokolldateien oder ob nur Protokolldateien aus dem Backup-Image verfügbar sind.

- Falls der Zugriff auf bereits vorhandene Protokolldateien oder auf archivierte Protokolldateien möglich ist, ist die folgende aktualisierende Recovery geeignet:

  ```
  db2 rollforward db datenbankname to end of logs and stop
  ```

  Anmerkung: Bevor Sie ein Backup ausführen, müssen Sie sicherstellen, dass für den Protokollarchivierungspfad ein gemeinsam genutztes Verzeichnis festgelegt ist, damit alle Member in der Lage sind, bei nachfolgenden aktualisierenden Recoverys auf die Protokolle zuzugreifen. Kann ein Member, für das die aktualisierende Recovery durchgeführt wird, nicht auf den Archivpfad zugreifen, wird die Nachricht SQL1273N zurückgegeben. Der folgende Befehl ist ein Beispiel dafür, wie das gemeinsam genutzte Verzeichnis als Protokollpfad festgelegt wird:

  ```
  db2 update db cfg using logarchmeth1 DISK:/db2fs/gpfs1/svtdbm5/svtdbm5/ArchiveLOGS
  ```

  (Hierbei ist *gpfs1* das gemeinsam genutzte Verzeichnis für die Member und *ArchiveLOGS* das eigentliche Verzeichnis, in dem die Protokolle archiviert werden.)
- Falls ausschließlich auf Protokolldateien zugegriffen werden kann, die aus dem Backup-Image stammen, ist die folgende aktualisierende Recovery geeignet:

  ```
  db2 rollforward db datenbankname to end of backup and stop
  ```

  Dieser Befehl gibt alle Protokollsätze wieder, die zum Erreichen des konsistenten Datenbankzustands benötigt werden, der bei Beendigung der Backup-Operation vorlag. Sie können diesen Befehl ebenfalls verwenden, wenn der Zugriff auf bereits vorhandene oder auf archivierte Protokolldateien möglich ist. Der Befehl wird jedoch an der Stelle gestoppt, an der die Backup-Operation endete. Zusätzliche Protokolle, die nach Beendigung der Backup-Operation generiert wurden, werden nicht verwendet.

  Ein Befehl **ROLLFORWARD DATABASE** mit angegebener Option END OF LOGS würde in diesem Fall die Nachricht SQL1273N zurückgeben. Ein nachfolgender Befehl **ROLLFORWARD DATABASE** mit der Option STOP wird erfolgreich ausgeführt und die Datenbank ist verfügbar, sofern die fehlenden Protokolldateien nicht benötigt werden. Sind die fehlenden Protokolldateien jedoch erforderlich (und ist das Stoppen der aktualisierenden Recovery keine sichere Option), gibt die aktualisierende Recovery wiederum die Nachricht SQL1273N zurück.

Beispiel

Ausgangspunkt sind die beiden Member M1 und M2. Die Systemzeit von M2 liegt 5 Sekunden nach der Systemzeit von M1. Der Protokolldatenstrom von M2 enthält die folgenden Protokollsätze:

A1 mit der Zeitmarke 2010-04-03-14.21.56

A2 mit der Zeitmarke 2010-04-03-14.21.56

B mit der Zeitmarke 2010-04-03-14.21.58

C mit der Zeitmarke 2010-04-03-14.22.01

Der Protokolldatenstrom von M1 enthält die folgenden Protokollsätze:

D mit der Zeitmarke 2010-04-03-14.21.55

E mit der Zeitmarke 2010-04-03-14.21.56

F mit der Zeitmarke 2010-04-03-14.21.57

Die Mindestrecoveryzeit für die Datenbank von M2 ist der Zeitpunkt 2010-04-03-14.21.55. Da die Systemzeit von M1 fünf Sekunden zurückliegt, sind die Protokollsätze D, E und F im zusammengeführten Protokolldatenstrom später angegeben:

```
MRT: 2010-04-03-14.21.55 (M2)
A1:  2010-04-03-14.21.56 (M2)
A2:  2010-04-03-14.21.56 (M2)
B:   2010-04-03-14.21.58 (M2)
D:   2010-04-03-14.21.55 (M1) --> korrespondierende Zeit von M2 ist 14.22.00
C:   2010-04-03-14.22.01 (M2)
E:   2010-04-03-14.21.56 (M1) --> korrespondierende Zeit von M2 ist 14.22.01
F:   2010-04-03-14.21.57 (M1) --> korrespondierende Zeit von M2 ist 14.22.02
```

Die Buchstaben (A1, A2, B usw.) bilden die Reihenfolge ab, in der die entsprechenden Protokollsätze tatsächlich zur Ausführungszeit (memberübergreifend) geschrieben wurden. Sie können feststellen, dass die Protokollsätze A1 und A2 des Members M2 dieselbe Zeitmarke aufweisen. Dies kann auftreten, wenn der DB2-Datenserver versucht, die Leistung zu optimieren, indem er den Commitprotokollsatz aus mehreren Transaktionen einschließt, wenn Daten aus dem Protokollpuffer in eine Protokolldatei geschrieben werden.

Der folgende Befehl gibt die Nachricht SQL1276N zurück (Der Status 'Aktualisierende Recovery anstehend' der Datenbank 'test' kann erst beendet werden, wenn die aktualisierende Recovery einen Zeitpunkt erreicht hat, der größer bzw. gleich '2010-04-03-14.21.55' ist, da der Knoten *knotennummer* Informationen enthält, die aktueller als der angegebene Zeitpunkt sind):

```
db2 rollforward db test to 2010-04-03-14.21.54
```

Der folgende Befehl führt jedoch eine aktualisierende Recovery der Datenbank bis einschließlich zum Protokollsatz A2 durch:

```
db2 rollforward db test to 2010-04-03-14.21.56
```

Da die Zeitmarke beider Protokollsätze (A1 und A2) kleiner-gleich der im Befehl angegebenen Zeit ist, werden beide Sätze angewendet. Der Protokollsatz B, dessen Zeitmarke (2010-04-03-14.21.58) größer als der im Befehl angegebene Wert (2010-04-03-14.21.56) ist, stoppt die aktualisierende Recovery und wird nicht angewendet. Der Protokollsatz D wird ebenfalls nicht angewendet, obwohl seine Zeitmarke kleiner als der angegebene Wert ist, da der höhere Wert des Protokollsatzes B (2010-04-03-14.21.58) zuerst festgestellt wurde. Der folgende Befehl führt eine aktualisierende Recovery der Datenbank bis einschließlich zum Protokollsatz D durch:

```
db2 rollforward db test to 2010-04-03-14.21.58
```

Der Protokollsatz C, dessen Zeitmarke (2010-04-03-14.22.01) größer als der angegebene Wert (2010-04-03-14.21.58) ist, stoppt die aktualisierende Recovery und wird nicht angewendet. Auch der Protokollsatz E wird nicht angewendet, obwohl seine Zeitmarke kleiner als der angegebene Wert ist.

Überwachen einer aktualisierenden Recovery

Sie können den Befehl **db2pd** oder den Befehl **LIST UTILITIES** verwenden, um den Fortschritt von aktualisierenden Recoveryoperationen für eine Datenbank zu überwachen.

Vorgehensweise

- Setzen Sie den Befehl **LIST UTILITIES** ab und geben Sie den Parameter **SHOW DETAIL** an.

  ```
  LIST UTILITIES SHOW DETAIL
  ```

- Setzen Sie den Befehl **db2pd** ab und geben Sie den Parameter **-recovery** an:
  db2pd -recovery

Ergebnisse

Für die aktualisierende Recovery gibt es zwei Phasen der Fortschrittsüberwachung: FORWARD und BACKWARD. In der Phase FORWARD werden Protokolldateien gelesen und die Protokollsätze auf die Datenbank angewendet. Für die aktualisierende Recovery wird zu Beginn dieser Phase für den geschätzten Gesamtaufwand UNBEKANNT angegeben. Die Menge der verarbeiteten Byte wird während der weiteren Ausführung des Prozesses aktualisiert.

Während der Phase BACKWARD werden alle nicht festgeschriebenen Änderungen, die während der Phase FORWARD angewendet wurden, rückgängig gemacht. Es wird eine Schätzung der Menge der zu verarbeitenden Protokolldaten in Byte angegeben. Die Menge der verarbeiteten Byte wird während der weiteren Ausführung des Prozesses aktualisiert.

Beispiel

Es folgt eine Beispielausgabe der Leistungsüberwachung einer aktualisierenden Recovery mit dem Befehl **db2pd**:

```
Recovery:
Recovery Status     0x00000401
Current Log         S0000005.LOG
Current LSN         0000001F07BC
Current LSO         000002551BEA
Job Type            ROLLFORWARD RECOVERY
Job ID              7
Job Start Time      (1107380474) Wed Feb  2 16:41:14 2005
Job Description     Database Rollforward Recovery
Invoker Type        User
Total Phases        2
Current Phase       1

Progress:
Address            PhaseNum Description StartTime               CompletedWork  TotalWork
0x0000000200667160 1        Forward     Wed Feb  2 16:41:14 2005 2268098 bytes  Unknown
0x0000000200667258 2        Backward    NotStarted              0 bytes        Unknown
```

Es folgt eine Beispielausgabe der Leistungsüberwachung einer aktualisierenden Recovery für eine Datenbank mit dem Befehl **LIST UTILITIES** unter Angabe der Option SHOW DETAIL:

```
ID                                = 7
Typ                               = ROLLFORWARD RECOVERY
Datenbankname                     = TESTDB
Membernummer                      = 0
Beschreibung                      = Aktualisierende Recovery einer Datenbank
Startzeit                         = 01/11/2012 16:56:53.770404
Status                            = Wird ausgeführt
Aufruftyp                         = Benutzer
Fortschrittsüberwachung:
   Geschätzte Fertigstellung      = 50
   Phasennummer                   = 1
      Beschreibung                = Forward
      Gesamte Arbeit              = 928236 Byte
      Abgeschlossene Arbeit       = 928236 Byte
      Startzeit                   = 01/11/2012 16:56:53.770492

   Phasennummer [Aktuell]         = 2
      Beschreibung                = Backward
      Gesamte Arbeit              = 928236 Byte
      Abgeschlossene Arbeit       = 0 Byte
      Startzeit                   = 01/11/2012 16:56:56.886036
```

Es folgt eine Beispielausgabe der Leistungsüberwachung einer aktualisierenden Recovery für Tabellenbereiche mit dem Befehl **LIST UTILITIES** unter Angabe der Option SHOW DETAIL:

```
ID                                   = 17
Typ                                  = ROLLFORWARD RECOVERY
Datenbankname                        = TESTDB
Membernummer                         = 0
Beschreibung                         = Aktualisierende Recovery für Tabellenbereiche (offline):3
Startzeit                            = 01/11/2012 17:04:27.269171
Status                               = Wird ausgeführt
Aufruftyp                            = Benutzer
Fortschrittsüberwachung:
   Geschätzte Fertigstellung (%) = 63
   Phasennummer                      = 1
      Beschreibung                   = Forward
      Gesamte Arbeit                 = 142
      Abgeschlossene Arbeit          = 90
      Startzeit                      = 01/11/2012 17:04:27.269283

   Phasennummer [Aktuell]            = 2
      Beschreibung                   = Backward
      Gesamte Arbeit                 = 0
      Abgeschlossene Arbeit          = 0
      Startzeit                      = Nicht gestartet
```

Erforderliche Berechtigungen für aktualisierende Recovery

Sie benötigen die Berechtigung SYSADM, SYSCTRL oder SYSMAINT, um das Dienstprogramm zur aktualisierenden Recovery verwenden zu können.

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf sie zuzugreifen. Berechtigungsstufen stellen eine Methode dar, um Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zusammenzufassen. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte.

Benutzer können nur auf die Objekte zugreifen, für die sie zugriffsberechtigt sind, d. h., für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Sitzungen zur aktualisierenden Recovery - Beispiele für CLP

Befehle für die aktualisierende Recovery können über die Befehlszeilenaufforderung eingegeben werden. Bevor Sie einen Befehl für eine aktualisierende Recovery eingeben, können Sie sich anhand einiger Beispielsitzungen mit der Vorgehensweise vertraut machen.

Beispiel 1

Der Befehl ROLLFORWARD DATABASE ermöglicht die Spezifikation mehrerer Operationen gleichzeitig, die jeweils mit dem Schlüsselwort AND voneinander getrennt werden müssen. Sie benötigen z. B. die folgenden Einzelbefehle, um eine aktualisierende Recovery bis zum Ende der Protokolle auszuführen und zu beenden:

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete
```

Diese Befehle können wie folgt kombiniert werden:

```
db2 rollforward db sample to end of logs and complete
```

Obwohl die beiden Befehle gleichbedeutend sind, wird empfohlen, solche Operationen in zwei Schritten auszuführen. Es ist wichtig, zu überprüfen, ob die aktualisierende Recoveryoperation wie erwartet fortgeschritten ist, bevor Sie sie stoppen, sodass keine Protokolle fehlen.

Wenn der ROLLFORWARD-Befehl auf einen Fehler stößt, wird die aktualisierende Recoveryoperation nicht beendet. Der Fehler wird zurückgegeben, sodass Sie den Fehler beheben und den Befehl erneut ausführen können. Wenn Sie den Fehler jedoch nicht beheben können, können Sie die Beendigung der aktualisierenden Recovery mit dem folgenden Befehl erzwingen:

```
db2 rollforward db sample complete
```

Dieser Befehl macht die Datenbank an dem Punkt in den Protokollen vor dem Fehler online verfügbar.

Beispiel 2

Geben Sie zur Ausführung einer aktualisierenden Recovery der Datenbank bis zum Protokollende (zwei Tabellenbereiche wurden wiederhergestellt) folgende Befehle ein:

```
db2 rollforward db sample to end of logs
db2 rollforward db sample to end of logs and stop
```

Diese beiden Anweisungen sind gleichbedeutend. Weder AND STOP noch AND COMPLETE sind für eine aktualisierende Tabellenbereichsrecovery bis zum Protokollende erforderlich. Tabellenbereichsnamen sind nicht erforderlich. Falls keine Namen angegeben werden, werden alle Tabellenbereiche, für die eine aktualisierende Recovery erforderlich ist, mit einbezogen. Wenn nur eine Untermenge dieser Tabellenbereiche aktualisierend wiederhergestellt werden soll, müssen die Namen der Tabellenbereiche angegeben werden.

Beispiel 3

Führen Sie, nachdem drei Tabellenbereiche wiederhergestellt wurden, für einen Tabellenbereich die aktualisierende Recovery bis zum Ende der Protokolldateien und für die anderen beiden bis zu einem bestimmten Zeitpunkt aus. Beides muss online erfolgen:

```
    db2 rollforward db sample to end of logs tablespace(TBS1) online
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
         tablespace(TBS2, TBS3) online
```

Beachten Sie, dass zwei aktualisierende Recoveryoperationen nicht gleichzeitig ausgeführt werden können. Sie können den zweiten Befehl erst aufrufen, nachdem die erste aktualisierende Recoveryoperation erfolgreich beendet wurde.

Beispiel 4

Führen Sie nach dem Restore der Datenbank die aktualisierende Recovery bis zu einem bestimmten Zeitpunkt unter Verwendung von OVERFLOW LOG PATH aus, womit Sie das Verzeichnis angeben, indem das Benutzerexitprogramm archivierte Protokolldateien speichert:

```
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
         overflow log path (/logs)
```

Beispiel 5

Im folgenden Beispiel wird angenommen, dass eine Datenbank namens 'sample' vorhanden ist. Für die Datenbank wird ein Backup durchgeführt, und die Recoveryprotokolle werden in das Backup-Image eingefügt; die Datenbank wird wiederhergestellt; dann wird eine aktualisierende Recovery bis zum Ende der Zeitmarke des Backups ausgeführt.

Durchführen eines Backups der Datenbank, einschließlich der Recoveryprotokolle im Backup-Image:

```
db2 backup db sample online include logs
```

Wiederherstellen der Datenbank mit diesem Backup-Image:

```
db2 restore db sample
```

Ausführen einer aktualisierenden Recovery der Datenbank bis zum Ende der Zeitmarke des Backups:

```
db2 rollforward db sample to end of backup
```

Beispiel 6 (Umgebungen mit partitionierten Datenbanken)

Es gibt drei Datenbankpartitionen: 0, 1 und 2. Der Tabellenbereich TBS1 ist in allen Datenbankpartitionen definiert, der Tabellenbereich TBS2 nur in den Datenbankpartitionen 0 und 2. Nachdem Sie die Datenbank in Datenbankpartition 1 und TBS1 in den Datenbankpartitionen 0 und 2 wiederhergestellt haben, führen Sie eine aktualisierende Recovery der Datenbank in Datenbankpartition 1 wie folgt aus:

```
db2 rollforward db sample to end of logs and stop
```

Dieser Befehl gibt die Warnung SQL1271 („Die Datenbank "name" wurde wiederhergestellt. Auf dem bzw. den Knoten 0 und 2 ist jedoch mindestens ein Tabellenbereich offline.”) zurück.

```
db2 rollforward db sample to end of logs
```

Durch diesen Befehl wird TBS1 in den Datenbankpartitionen 0 und 2 aktualisierend wiederhergestellt. In diesem Fall ist die Klausel TABLESPACE(TBS1) optional.

Beispiel 7 (Umgebungen mit partitionierten Datenbanken)

Im folgenden Beispiel wird angenommen, dass eine partitionierte Datenbank namens 'sample' vorhanden ist. Für alle Datenbankpartitionen wird ein SSV-Backup (SSV = Single System View, Einzelsystemsicht) durchgeführt; die Datenbank wird auf allen Datenbankpartitionen wiederhergestellt; dann wird eine aktualisierende Recovery der Datenbank bis zum Ende der Zeitmarke des Backups ausgeführt.

Durchführen eines SSV-Backups:

```
db2 backup db sample on all nodes online include logs
```

Wiederherstellen der Datenbank auf allen Datenbankpartitionen:

```
db2_all "db2 restore db sample taken at 1998-04-03-14.21.56"
```

Ausführen einer aktualisierenden Recovery der Datenbank bis zum Ende der Zeitmarke des Backups:

```
db2 rollforward db sample to end of backup on all nodes
```

Beispiel 8 (Umgebungen mit partitionierten Datenbanken)

Im folgenden Beispiel wird angenommen, dass eine partitionierte Datenbank namens 'sample' vorhanden ist. Für alle Datenbankpartitionen wird mit einem Befehl ein Backup unter Verwendung von 'db2_all' durchgeführt; die Datenbank wird auf allen Datenbankpartitionen wiederhergestellt; dann wird eine aktualisierende Recovery der Datenbank zum Ende der Zeitmarke des Backups ausgeführt.

Durchführen eines Backups für alle Datenbankpartitionen mit einem Befehl unter Verwendung von 'db2_all':

```
db2_all "db2 backup db sample include logs to //dir/"
```

Wiederherstellen der Datenbank auf allen Datenbankpartitionen:

```
db2_all "db2 restore db sample from //dir/"
```

Ausführen einer aktualisierenden Recovery der Datenbank bis zum Ende der Zeitmarke des Backups:

```
db2 rollforward db sample to end of backup on all nodes
```

Beispiel 9 (Umgebungen mit partitionierten Datenbanken)

Nachdem Sie den Tabellenbereich TBS1 nur in den Datenbankpartitionen 0 und 2 wiederhergestellt haben, stellen Sie TBS1 wie folgt in den Datenbankpartitionen 0 und 2 aktualisierend wieder her:

```
db2 rollforward db sample to end of logs
```

Datenbankpartition 1 wird ignoriert.

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

Dieser Befehl schlägt fehl, weil TBS1 auf Datenbankpartition 1 nicht für eine aktualisierende Recovery bereit ist. SQL4906N wird dokumentiert.

```
db2 rollforward db sample to end of logs on
         dbpartitionnums (0, 2) tablespace(TBS1)
```

Dieser Befehl wird erfolgreich beendet.

```
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
         tablespace(TBS1)
```

Dieser Befehl schlägt fehl, weil TBS1 in Datenbankpartition 1 nicht für eine aktualisierende Recovery bereit ist. Alle Teile müssen gemeinsam aktualisierend wiederhergestellt werden.

Anmerkung: Bei der aktualisierenden Tabellenbereichsrecovery bis zu einem Zeitpunkt wird die DBPARTITIONNUM-Klausel nicht akzeptiert. Die aktualisierende Recoveryoperation muss in allen Datenbankpartitionen ausgeführt werden, in denen sich der Tabellenbereich befindet.

Geben Sie nach dem Restore von TBS1 in Datenbankpartition 1 Folgendes ein:

```
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
         tablespace(TBS1)
```

Dieser Befehl wird erfolgreich beendet.

Beispiel 10 (Umgebungen mit partitionierten Datenbanken)

Nachdem Sie einen Tabellenbereich in allen Datenbankpartitionen wiederhergestellt haben, stellen Sie ihn wie folgt aktualisierend bis zum Zeitpunkt PIT2 wieder her. Geben Sie jedoch AND STOP nicht an. Die aktualisierende Recoveryoperation wird weiterhin ausgeführt. Brechen Sie sie wie folgt ab, und führen Sie eine aktualisierende Recovery bis zum Zeitpunkt PIT1 aus:

```
db2 rollforward db sample to pit2 tablespace(TBS1)
   db2 rollforward db sample cancel tablespace(TBS1)

** TBS1 in allen Datenbankpartitionen wiederherstellen **

   db2 rollforward db sample to pit1 tablespace(TBS1)
   db2 rollforward db sample stop tablespace(TBS1)
```

Beispiel 11 (Umgebungen mit partitionierten Datenbanken)

Stellen Sie wie folgt einen Tabellenbereich wieder her, der sich in den in der Datei db2nodes.cfg aufgelisteten acht Datenbankpartitionen (3 bis 10) befindet:

```
db2 rollforward database dwtest to end of logs tablespace (tssprodt)
```

Diese Operation bis zum Protokollende (nicht bis zu einem Zeitpunkt) wird erfolgreich beendet. Die Datenbankpartitionen, in denen sich die Tabellenbereiche befinden, müssen nicht angegeben werden. Das Dienstprogramm verwendet standardmäßig die Datei db2nodes.cfg.

Beispiel 12 (Umgebungen mit partitionierten Datenbanken)

Stellen Sie wie folgt sechs kleine Tabellenbereiche aktualisierend wieder her, die sich in einer Datenbankpartitionsgruppe mit nur einer Datenbankpartition befinden (in Datenbankpartition 6):

```
db2 rollforward database dwtest to end of logs on dbpartitionnum (6)
         tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

Diese Operation bis zum Protokollende (nicht bis zu einem Zeitpunkt) wird erfolgreich beendet.

Beispiel 13 (partitionierte Tabellen - aktualisierende Recovery bis zum Ende der Protokolle in allen Datenbankpartitionen)

Eine partitionierte Tabelle wird in den Tabellenbereichen tbsp1, tbsp2, tbsp3 mit einem Index in tbsp0 erstellt. Später fügt ein Benutzer der Tabelle Datenpartitionen in tbsp4 hinzu und ordnet (ATTACH) Datenpartitionen aus der Tabelle in tbsp5 zu. Alle Tabellenbereiche können bis zum Ende der Protokolle aktualisierend wiederhergestellt werden.

```
db2 rollforward db PBARDB to END OF LOGS and stop
        tablespace(tbsp0, tbsp1, tbsp2, tbsp3, tbsp4, tbsp5)
```

Dieser Befehl wird erfolgreich beendet.

Beispiel 14 (partitionierte Tabellen - aktualisierende Recovery bis zum Ende der Protokolle in einem Tabellenbereich)

Eine partitionierte Tabelle wird zu Anfang in den Tabellenbereichen tbsp1, tbsp2, tbsp3 mit einem Index in tbsp0 erstellt. Später fügt ein Benutzer der Tabelle Datenpartitionen in tbsp4 hinzu und ordnet (ATTACH) Datenpartitionen aus der Tabelle

in tbsp5 zu. Der Tabellenbereich tbsp4 wird beschädigt und erfordert einen Restore sowie eine aktualisierende Recovery bis zum Ende der Protokolle.

```
db2 rollforward db PBARDB to END OF LOGS and stop tablespace(tbsp4)
```

Dieser Befehl wird erfolgreich beendet.

Beispiel 15 (partitionierte Tabellen - aktualisierende Recovery bis zu einem Zeitpunkt aller Datenpartitionen einschließlich der hinzugefügten (ADD), zugeordneten (ATTACH), getrennten (DETACH) oder derer mit Indizes)

Eine partitionierte Tabelle wird in den Tabellenbereichen tbsp1, tbsp2, tbsp3 mit einem Index in tbsp0 erstellt. Später fügt ein Benutzer der Tabelle Datenpartitionen in tbsp4 hinzu, ordnet (ATTACH) Datenpartitionen aus der Tabelle in tbsp5 zu und hebt die Zuordnung von Datenpartitionen aus tbsp1 auf (DETACH). Der Benutzer führt eine aktualisierende Recovery bis zu einem Zeitpunkt mit allen Tabellenbereichen aus, die von der partitionierten Tabelle verwendet werden, einschließlich der Tabellenbereiche, die in der Klausel INDEX IN angegeben wurden.

```
db2 rollforward db PBARDB to 2005-08-05-05.58.53 and stop
        tablespace(tbsp0, tbsp1, tbsp2, tbsp3, tbsp4, tbsp5)
```

Dieser Befehl wird erfolgreich beendet.

Beispiel 16 (partitionierte Tabellen - aktualisierende Recovery bis zu einem Zeitpunkt für eine Untergruppe der Tabellenbereiche)

Eine partitionierte Tabelle wird in drei Tabellenbereichen (tbsp1, tbsp2, tbsp3) erstellt. Später hebt der Benutzer die Zuordnung aller Datenpartitionen aus tbsp3 auf. Die aktualisierende Recovery bis zu einem Zeitpunkt ist nur für die Tabellenbereiche tbsp1 und tbsp2 zulässig.

```
db2 rollforward db PBARDB to 2005-08-05-06.02.42 and stop
         tablespace( tbsp1, tbsp2)
```

Dieser Befehl wird erfolgreich beendet.

Kapitel 15. Datenrecovery mit IBM Tivoli Storage Manager (TSM)

Wenn Sie den Befehl **BACKUP DATABASE** oder **RESTORE DATABASE** aufrufen, können Sie angeben, dass Sie die Backup- oder Restoreoperation für die Datenbank bzw. den Tabellenbereich mit IBM Tivoli Storage Manager (TSM) verwalten möchten.

Der erforderliche Mindeststand der TSM-Client-API ist Version 4.2.0 mit Ausnahme folgender Systeme, die eine andere Version erfordern:

- 64-Bit-Solaris-Systeme, für die Version 4.2.1 der TSM-Client-API erforderlich ist.
- 64-Bit-Windows-Betriebssysteme, für die Version 5.1 der TSM-Client-API erforderlich ist.
- Alle Windows x64-Systeme, für die Version 5.3.2 der TSM-Client-API erforderlich ist.
- 32-Bit-Linux für IBM Power Systems und pSeries, für die mindestens Version 5.1.5 der TSM-Client-API erforderlich ist.
- 64-Bit-Linux für IBM Power Systems und pSeries, für die mindestens Version 5.2.2 der TSM-Client-API erforderlich ist.
- 64-Bit-Linux auf AMD Opteron-Systeme, für die mindestens Version 5.2.0 der TSM-Client-API erforderlich ist.
- Linux für zSeries-Systeme, für die mindestens Version 5.2.2 der TSM-Client-API erforderlich ist.

Konfigurieren eines Tivoli Storage Manager-Clients

Bevor der DB2-Datenbankmanager einen TSM-Client (TSM = IBM Tivoli Storage Manager) zur Verwaltung von Backup- bzw. Restoreoperationen für Datenbanken oder Tabellenbereiche verwenden kann, müssen Sie die TSM-Umgebung konfigurieren.

Vorbereitende Schritte

Es muss ein funktionierender TSM-Client und -Server installiert und konfiguriert werden. Darüber hinaus muss auf jedem DB2-Datenbankserver die TSM-Client-API installiert sein. TSM-Client-Proxy-Knoten werden unterstützt, wenn der TSM-Server entsprechend konfiguriert ist. Informationen zur Serverkonfiguration und zur Unterstützung von Proxy-Knoten finden Sie im Abschnitt „Aspekte der Verwendung von Tivoli Storage Manager" auf Seite 476 oder in der Tivoli-Dokumentation.

Vorgehensweise

Gehen Sie wie folgt vor, um die TSM-Umgebung für die Verwendung durch DB2-Datenbanksysteme zu konfigurieren:

1. Definieren Sie die von der TSM-Client-API verwendeten Umgebungsvariablen:

   DSMI_DIR
   Gibt den benutzerdefinierten Verzeichnispfad an, in dem sich die API-Datei für den Trusted Agent (`dsmtca`) befindet.

   DSMI_CONFIG
   Gibt den benutzerdefinierten Verzeichnispfad zur Datei `dsm.opt` an, die die TSM-Benutzeroptionen enthält. Im Unterschied zu den beiden ande-

ren Variablen müssen Sie bei dieser Variablen einen vollständig qualifizierten Pfad und einen Dateinamen angeben.

DSMI_LOG
Gibt den benutzerdefinierten Verzeichnispfad an, in dem das Fehlerprotokoll (`dsierror.log`) erstellt wird.

Anmerkung: In einer Umgebung mit partitionierter Datenbank müssen diese Einstellungen in der Datei `sqllib/userprofile` angegeben werden.

2. Falls an diesen Umgebungsvariablen Änderungen vorgenommen werden, während der Datenbankmanager ausgeführt wird, stoppen Sie den Datenbankmanager und starten Sie ihn erneut. Beispiel:
   - Stoppen Sie den Datenbankmanager mit dem Befehl **db2stop**.
   - Starten Sie den Datenbankmanager mit dem Befehl **db2start**.
3. Abhängig von der Konfiguration des Servers benötigt der Tivoli-Client für den Datenaustausch mit dem TSM-Server möglicherweise ein Kennwort.

   Falls die TSM-Umgebung für die Verwendung von `PASSWORDACCESS=generate` konfiguriert ist, muss für den Tivoli-Client ein Kennwort eingerichtet sein.

   Die ausführbare Datei `dsmapipw` ist im Verzeichnis `sqllib/adsm` des Instanzeigners installiert. Mithilfe dieser ausführbaren Datei können Sie das TSM-Kennwort definieren und zurücksetzen.

   Damit Sie den Befehl **dsmapipw** ausführen können, müssen Sie als lokaler Administrator oder als Root angemeldet sein. Bei der Ausführung dieses Befehls werden Sie aufgefordert, die folgenden Informationen einzugeben:
   - *Altes Kennwort*, d. h. das aktuelle (vom TSM-Server erkannte) Kennwort für den TSM-Knoten. Wenn Sie diesen Befehl zum ersten Mal ausführen, ist dies das Kennwort, das vom TSM-Administrator bei der Registrierung Ihres Knotens auf dem TSM-Server vergeben wurde.
   - *Neues Kennwort*, d. h. das neue Kennwort für den TSM-Knoten, das auf dem TSM-Server gespeichert wird. (Sie werden zweimal zur Eingabe des neuen Kennworts aufgefordert, um Eingabefehler festzustellen.)

   Anmerkung: Benutzer, die den Befehl **BACKUP DATABASE** oder den Befehl **RESTORE DATABASE** aufrufen, brauchen dieses Kennwort nicht zu kennen. Sie müssen den Befehl **dsmapipw** nur ausführen, wenn Sie für die erste Verbindung ein Kennwort festlegen und nachdem das Kennwort auf dem TSM-Server zurückgesetzt wurde.

Nächste Schritte

Entsprechend Ihrer Strategie für die Backup- und Protokollarchivierung müssen Sie möglicherweise weitere Konfigurationsschritte für die TSM-Clients ausführen, um Proxy-Knoten zu verwenden. Proxy-Knoten ermöglichen die Konsolidierung von Backups und Protokollarchiven für Datenbanken auf mehreren Clientknoten oder für mehrere Benutzer unter einem gemeinsamen Zielknotennamen auf dem TSM-Server. Diese Konfiguration ist hilfreich, wenn das Backup von verschiedenen Administratoren oder Computern ausgeführt werden kann (z. B. innerhalb von Clustern). Die Option `asnodename` ermöglicht außerdem die Datenwiederherstellung von einem anderen Computer oder Benutzer, der nicht das Backup ausgeführt hat.

Wenn Sie TSM in Ihrer DB2 pureScale-Umgebung verwenden wollen, sind Konfigurationen mit Proxy-Knoten zu empfehlen, da jedes Member als TSM-Client bzw. -Knoten dargestellt und einem allgemeinen Proxy-Knoten zugeordnet werden kann.

Wenn die Proxy-Knoten nicht standardmäßig verwendet werden sollen, ist keine zusätzliche Clientkonfiguration erforderlich. Wenn Sie Backup- oder Restore-Operationen über Proxy-Knoten ausführen möchten, geben Sie den Wert `asnodename` im Parameter **OPTIONS** beim Aufrufen des Befehls **BACKUP DATABASE** oder **RESTORE DATABASE** an.

Verwenden Sie die folgenden Methoden, wenn TSM-Proxy-Knoten standardmäßig verwendet werden sollen:

- Aktualisieren Sie die Datenbankkonfigurationsparameter so, dass verschiedene Proxy-Knoten für verschiedene Datenbanken verwendet werden.
- Aktualisieren Sie die Datei `dsm.sys` so, dass für alle Benutzer und Datenbanken auf einer Maschine derselbe Proxy-Knoten verwendet wird.

  Anmerkung: Jede Benutzer-Host-Kombination, die denselben TSM-Proxy-Namen verwendet, wird für TSM als dieselbe DB2-Instanz angezeigt. Wenn mehrere DB2-Instanzen denselben Datenbanknamen in einer Proxy-Konfiguration des TSM-Clientknotens verwenden, kann dies bedeuten, dass diese Instanzen sich möglicherweise gegenseitig die Protokollarchive und Backup-Images überschreiben. Gehen Sie wie folgt vor, um dies zu vermeiden:
  - Erstellen Sie für jede DB2-Instanz einen anderen Proxy-Hostnamen.
  - Verwenden Sie die Proxy-Funktion des TSM-Clientknotens nicht, wenn mehrere DB2-Instanzen möglicherweise Datenbanken mit demselben TSM-Proxy-Namen erstellen.

TSM-Clientkonfiguration mit `vendoropt`, `logarchopt1` und `logarchopt2`

Sie können einen oder mehrere der folgenden Datenbankkonfigurationsparameter festlegen, um unterschiedliche Proxy-Knoteneinstellungen für die einzelnen Datenbanken zu verwenden:

- Damit Befehle in TSM (z. B. Backup und Restore) Proxy-Knoten verwenden können, geben Sie die Option `asnodename` im Datenbankkonfigurationsparameter **vendoropt** wie folgt an:

  ```
  db2 update db cfg for dbname using vendoropt "'-asnodename=proxy-knoten'"
  ```

  Dabei steht *proxy-knoten* für den Namen des gemeinsam genutzten TSM-Proxy-Knotens.
- Setzen Sie zum Konfigurieren der Protokollarchivierung auf dem TSM-Server den Datenbankkonfigurationsparameter **logarchmeth1** auf 'TSM' und geben Sie den Namen des Proxy-Knotens in der Option '`asnodename`' des Datenbankkonfigurationsparameters **logarchopt1** wie folgt an:

  ```
  db2 update db cfg for dbname using logarchmeth1 tsm
  logarchopt1 "'-asnodename=proxy-knoten'"
  ```

  Dabei steht *proxy-knoten* für den Namen des gemeinsam genutzten TSM-Proxy-Knotens.

  In den Datenbankkonfigurationsparametern **logarchmeth2** und **logarchopt2** können ähnliche Änderungen vorgenommen werden.

In DB2 pureScale-Umgebungen sind diese Datenbankkonfigurationsparameter globale Parameter und Sie können sie von jedem Member aus festlegen.

TSM-Clientkonfiguration mit der Datei `dsm.sys`

1. Fügen Sie die Proxy-Knoteninformationen in der Datei dsm.sys wie folgt hinzu:

   ```
   asnodename proxy-knoten
   ```

Dabei steht *proxy-knoten* für den Namen des gemeinsam genutzten TSM-Proxy-Knotens.

2. Stellen Sie wie folgt sicher, dass die angegebene Datei `dsm.opt` im Pfad **DSMI_CONFIG** den Namen des TSM-Servers enthält:

   ```
   servername servername
   ```

   Dabei ist *servername* der Name des TSM-Servers.

Aspekte der Verwendung von Tivoli Storage Manager

Mithilfe von Tivoli Storage Manager können Sie Backup-Images einer DB2-Datenbank erstellen. Bei der Entscheidung darüber, welches Tool für das Backup der Datenbank verwendet werden soll, müssen die Features und Einschränkungen aller verfügbaren Optionen berücksichtigt werden.

- Bestimmte Funktionen innerhalb von Tivoli Storage Manager (TSM) können Sie möglicherweise nur verwenden, indem Sie den vollständig qualifizierten Pfadnamen des Objekts angeben, das diese Funktion verwendet. (Beachten Sie, dass unter Windows-Betriebssystemen das Zeichen \ statt / eingegeben werden muss.) Für vollständig qualifizierte Pfadnamen gilt Folgendes:
  - Der Pfadname eines Objekts für eine Datenbankgesamtrecovery setzt sich wie folgt zusammen: `/datenbank/DBPARTnnn/FULL_BACKUP.zeitmarke.folgenr`
  - Der Pfadname eines Objekts für eine inkrementelle Datenbankrecovery setzt sich wie folgt zusammen: `/datenbank/DBPARTnnn/DB_INCR_BACKUP.zeitmarke.folgenr`
  - Der Pfadname eines Objekts für eine inkrementelle Datenbankdelta-Recovery setzt sich wie folgt zusammen: `/datenbank/DBPARTnnn/DB_DELTA_BACKUP.zeitmarke.folgenr`
  - Der Pfadname eines Objekts für eine Tabellenbereichsgesamtrecovery setzt sich wie folgt zusammen: `/datenbank/DBPARTnnn/TSP_BACKUP.zeitmarke.folgenr`
  - Der Pfadname eines Objekts für eine inkrementelle Tabellenbereichsrecovery setzt sich wie folgt zusammen: `/datenbank/DBPARTnnn/TSP_INCR_BACKUP.zeitmarke.folgenr`
  - Der Pfadname für eine inkrementelle Tabellenbereichsdelta-Recovery setzt sich wie folgt zusammen: `/datenbank/DBPARTnnn/TSP_DELTA_BACKUP.zeitmarke.folgenr`

  Dabei steht *datenbank* für den Datenbankaliasnamen und `DBPARTnnn` für die Nummer der Datenbankpartition. Die Namen in Großbuchstaben müssen genau wie dargestellt eingegeben werden.
- Wenn Sie mehrere Backup-Images mit demselben Datenbankaliasnamen haben, dienen die in einem vollständig qualifizierten Namen angegebene Zeitmarke und die Folgenummer als Unterscheidungsmerkmal. Sie müssen TSM abfragen, um die zu verwendende Backup-Version zu ermitteln.
- Wenn Sie eine Online-Backup-Operation ausführen und die Optionen USE TSM und INCLUDE LOGS angeben, kann es zu einem Deadlock kommen, wenn die beiden Prozesse gleichzeitig versuchen, auf dasselbe Bandlaufwerk zu schreiben. Wenn Sie ein Bandlaufwerk als Speichereinheit für Protokolle und Backup-Images verwenden, müssen Sie zwei separate Bandpools für TSM definieren, d. h. einen für das Backup-Image und einen für die archivierten Protokolle.
- Damit Client-Proxy-Knoten verwendet werden können, muss der TSM-Administrator die folgenden Schritte auf dem TSM-Server ausführen:

1. Wenn die DB2-Clients noch nicht auf dem TSM-Server registriert sind, registrieren Sie jeden Client mit dem TSM-Befehl **register node**.
2. Registrieren Sie auf dem TSM-Server einen (virtuellen) allgemeinen TSM-Knotennamen, der von der Clientgruppe verwendet werden solle, mit dem TSM-Befehl **register node**.
3. Erteilen Sie allen Computern in der Gruppe die Proxy-Berechtigung mit dem TSM-Befehl **grant proxynode**.

Informationen zum Einrichten von Proxy-Knotenclients finden Sie im Abschnitt „Konfigurieren eines Tivoli Storage Manager-Clients" auf Seite 473 oder in der Tivoli-Dokumentation.

- Bei inkrementellen Backups, bei denen nur wenige Seiten geändert wurden, muss möglicherweise der TSM-Parameter **IDLETIMEOUT** so erhöht werden, dass sein Wert höher ist als die Zeit, die für ein erfolgreiches Backup des größten Tabellenbereichs benötigt wird. Dadurch wird verhindert, dass TSM die Sitzung vor Beendigung eines inkrementellen Backups schließt.

Kapitel 16. DB2 Advanced Copy Services (ACS)

DB2 ACS (Advanced Copy Services, erweiterte Kopierservices) ermöglicht Ihnen die Verwendung der leistungsstarken Kopiertechnologie einer Speichereinheit, um im Rahmen von Backup- und Restoreoperationen Daten kopieren zu können.

Bei einer herkömmlichen Backup- oder Restoreoperation kopiert der Datenbankmanager mithilfe von Systemaufrufen Daten auf eine bzw. von einer Platte oder Speichereinheit. Die Möglichkeit, Daten unter Verwendung der Speichereinheit zu kopieren, beschleunigt die Backup- und Restoreoperationen erheblich. Eine Backup-Operation, die DB2 ACS verwendet, wird als Momentaufnahmebackup bezeichnet.

Zur Ausführung von Backup- und Restoreoperationen für Momentaufnahmen benötigen Sie einen DB2 ACS-API-Treiber für Ihre Speichereinheit. Eine Liste der unterstützten Speicherhardware für den integrierten Treiber enthält die Tivoli-Dokumentation im Abschnitt Unterstützte Speichersubsysteme.

DB2 ACS-Schnittstellen mit einer Speicherlösung wie Tivoli Storage FlashCopy Manager sind ab Version 10.1 im DB2-Produktpaket enthalten. Ausführliche Anweisungen zur Konfiguration und Verwendung von Tivoli Storage FlashCopy Manager finden Sie in der Tivoli-Dokumentation unter der Adresse http://www.ibm.com/developerworks/wikis/display/tivolidoccentral/Tivoli+Storage+FlashCopy+Manager.

DB2 ACS - Best Practices

Beachten Sie bei Installation und Konfiguration von DB2 ACS die folgenden Best Practices.

Angabe einer dedizierten Datenträgergruppe für Protokollpfade
Es empfiehlt sich, Protokollpfade zu verwenden, die auf dem zugehörigen Momentaufnahmedatenträger enthalten und somit unabhängig vom Datenbankverzeichnis und Datenbankcontainern sind.

Angabe einer Datenträgergruppe für jede einzelne Datenbankpartition
In einer Umgebung mit partitionierten Datenbanken muss sich jede einzelne Datenbankpartition auf einer Gruppe von Momentaufnahmedatenträgern befinden, die unabhängig von den übrigen Datenbankpartitionen ist.

Einschränkungen für die integrierte Version von Tivoli Storage FlashCopy Manager

Falls Sie DB2 Advanced Copy Services (ACS) mit der Version von Tivoli Storage FlashCopy Manager verwenden, die mit DB2 ausgeliefert sind, müssen Sie einige Einschränkungen beachten.

Eine Datenträgerfreigabe wird nicht unterstützt. Wenn sich eine Datenbankpartition auf demselben Speicherdatenträger befindet wie eine andere Datenbankpartition, sind Momentaufnahmeoperationen nicht zulässig. Zur Verwendung von Tivoli Storage FlashCopy Manager ohne eine Lizenzdatei (also zur Verwendung der integrierten Version) müssen sich darüber hinaus die Datenbank- und Protokolldatenträger in einem Dateisystem befinden, das Anforderungen FREEZE und THAW

unterstützt. Tabelle 18 listet eine Reihe von funktionalen Einschränkungen auf, die bei Nutzung einer Volllizenz von IBM Tivoli Storage Manager (TSM) nicht bestehen.

Tabelle 18. Vergleich der unterstützten Funktionen zwischen der integrierten Version von Tivoli Storage FlashCopy Manager, die zusammen mit DB2 ausgeliefert wird, und der Vollversion des IBM Tivoli Storage Manager-Produkts

Funktionselement	Unterstützung bei der integrierten Version von Tivoli Storage FlashCopy Manager	Unterstützung bei Tivoli Storage FlashCopy Manager
Lokale Momentaufnahmebackupversionen	Maximal werden zwei Momentaufnahmeversionen unterstützt.	Keine Produktbegrenzung. Tivoli Storage FlashCopy Manager unterstützt so viele Versionen, wie die Speichereinheit und die verfügbaren Ressourcen zulassen.
Momentaufnahmeunterstützung in Kombination mit Spiegelung	Keine Unterstützung.	Unterstützung von Momentaufnahmen von Quellen- oder Zielspiegelgruppen für AIX LVM-Spiegelung (LVM = Logical Volume Manager).
Integriertes Backup des Momentaufnahmeimages auf Band	Keine integrierte Unterstützung. Traditionelle und Momementaufnahmebackups ergänzen einander, sind aber nicht integriert.	Voll integrierte Unterstützung des Backups des Momentaufnahmeimages auf TSM. Mit einem einzelnen Backup-Befehl kann das Momentaufnahmebackup und das Backup auf TSM ausgeführt werden.
Backup auf Band - ausgelagert vom Produktionsserver	Keine integrierte Unterstützung des Backups auf Band	Voll integrierte Unterstützung der Ausführung des Backups auf TSM von einem sekundären Host.
Integriertes Backup des Tivoli Storage FlashCopy Manager-Repositorys	Keine Unterstützung. Externes Backup kann erfolgen, wenn das Repository inaktiv ist oder heruntergefahren wurde.	Automatisches Backup auf TSM

Weitere Informationen zu Tivoli Storage FlashCopy Manager enthält die aktuelle Dokumentation, die über die Seite Tivoli Documentation Central zugänglich ist.

Aktivieren von DB2 Advanced Copy Services (ACS)

Zur Verwendung von DB2 Advanced Copy Services (ACS) oder zur Ausführung von Momentaufnahmebackup-Operationen muss DB2 ACS installiert, aktiviert und konfiguriert werden.

Vorbereitende Schritte

Zur Ausführung von Backup- und Restoreoperationen für Momentaufnahmen benötigen Sie einen DB2 ACS-API-Treiber für Ihre Speichereinheit. Eine Liste der unterstützten Speicherhardware für den integrierten Treiber enthält die Tivoli-Dokumentation im Abschnitt Unterstützte Speichersubsysteme.

Vorgehensweise

1. Installieren Sie DB2 ACS. Siehe hierzu „Installieren von DB2 Advanced Copy Services (ACS)".
2. Erstellen Sie die Datenbankmanagerinstanz(en), für die Sie DB2 ACS verwenden möchten.

   Bei der Erstellung einer neuen Datenbankmanagerinstanz wird das Verzeichnis `acs` im Verzeichnis `sqllib` der neuen Instanz erstellt. Da jede Datenbankmanagerinstanz über das Verzeichnis `acs` verfügt, können die einzelnen Datenbankmanagerinstanzen unterschiedlich konfiguriert werden.
3. Führen Sie für jede Datenbankmanagerinstanz, mit der Sie DB2 ACS verwenden möchten, die folgenden Schritte aus:
   a. Aktivieren Sie DB2 ACS. Siehe hierzu „Manuelles Aktivieren von DB2 Advanced Copy Services (ACS)" auf Seite 482.
   b. Konfigurieren Sie DB2 ACS. Siehe hierzu „Konfigurieren von DB2 Advanced Copy Services (ACS)" auf Seite 483.

Ergebnisse

Nachdem Sie DB2 ACS aktiviert haben, müssen Sie Ihre Speicherlösung konfigurieren, bevor Sie Momentaufnahmebackups durchführen können. Anweisungen zum Konfigurieren und Verwenden der Version von Tivoli Storage FlashCopy Manager, die in Ihrem DB2-Produkt integriert ist, können Sie der aktuellen Dokumentation entnehmen, die über die Seite Tivoli Documentation Central zugänglich ist.

Installieren von DB2 Advanced Copy Services (ACS)

Die Dateien und Bibliotheken, die für DB2 Advanced Copy Services (ACS) erforderlich sind, werden nur während der Standard- sowie der angepassten Installation installiert.

Vorbereitende Schritte

Vor der Installation von ACS müssen folgende Bibliotheken installiert sein:

Unter AIX:

- `ln -s /opt/freeware/lib/powerpc-ibm-aix5.3.0//libgcc_s.a /usr/lib/libgcc_s.a`

Sie sollten außerdem die Informationen in den folgenden Abschnitten prüfen:

- „DB2 ACS Installation und Konfiguration - Best Practices". Siehe „DB2 ACS - Best Practices" auf Seite 479.
- „Einschränkungen für die integrierte Version von Tivoli Storage FlashCopy Manager". Siehe hierzu „Einschränkungen für die integrierte Version von Tivoli Storage FlashCopy Manager" auf Seite 479.

Unter Red Hat Enterprise Linux:

- `ln -s libssl.so.0.9.7xxx libssl.so.0.9.7`
- `ln -s libcrypto.so.0.9.7xxx libcrypto.so.0.9.7`
- `ln -s libssl.so.0.9.7xxx libssl.so`
- `ln -s libssl.so.0.9.7xxx libssl.so.0`

Einschränkungen

DB2 ACS unterstützt einen Teil der Hardware und Betriebssysteme, die von IBM Data Server unterstützt werden. Eine Liste der von DB2 ACS unterstützten Hardware und Betriebssysteme enthält die Tivoli-Dokumentation im Abschnitt Unterstützte Speichersubsysteme.

Vorgehensweise

1. Installieren Sie DB2 for Linux, UNIX and Windows mit dem Befehl **db2_install**, einer Standardinstallation, einer angepassten Installation, oder einer Anpassungsantwortdatei unter Verwendung des Antwortdateischlüsselworts **ACS**.

   Wichtig: Der Befehl **db2_install** wird nicht weiter unterstützt und wird in einem zukünftigen Release möglicherweise entfernt. Verwenden Sie stattdessen den Befehl **db2setup** mit einer Antwortdatei.

   Anmerkung: DB2 ACS wird nicht mehr automatisch während der Kompaktinstallation installiert, wie dies in früheren Releases der Fall war. Wenn Sie bereits eine Kompaktinstallation ausgeführt haben und später DB2 ACS installieren möchten, führen Sie die Installation mit einer Anpassungsantwortdatei und dem Schlüsselwort 'ACS' durch.
2. Fügen Sie in die TCP/IP-Servicedatei einen Port für den DB2 ACS-Agenten ein. Beispiel:

   ```
   db2acs  5400/tcp # DB2 ACS service port
   ```

Nächste Schritte

Nachdem DB2 ACS installiert wurde, müssen Sie DB2 ACS aktivieren und DB2 ACS konfigurieren. Weitere Informationen finden Sie in „Manuelles Aktivieren von DB2 Advanced Copy Services (ACS)" und „Konfigurieren von DB2 Advanced Copy Services (ACS)" auf Seite 483.

Manuelles Aktivieren von DB2 Advanced Copy Services (ACS)

Bevor Sie mit DB2 Advanced Copy Services (ACS) ein Momentaufnahmebackup für eine bestimmte Datenbankmanagerinstanz durchführen können, muss DB2 ACS für diese Instanz aktiviert werden.

Bei der Erstellung der Datenbankmanagerinstanz oder bei einem Upgrade von IBM Data Server ruft der Datenbankmanager das Script **setup_db2.sh** automatisch auf, um die DB2 ACS-Funktionalität zu aktivieren. Sie können DB2 ACS jedoch auch durch eine manuelle Ausführung des Scripts aktivieren.

Vorbereitende Schritte

Damit DB2 ACS aktiviert werden kann, muss es bereits installiert worden sein (außer bei Verwendung einer Kompaktinstallation ist dies normalerweise schon erfolgt). Darüber hinaus müssen die Datenbankmanagerinstanz oder die Datenbankmanagerinstanzen, mit der/denen Sie DB2 ACS verwenden wollen, erstellt worden sein.

Informationen zu diesem Vorgang

Bei der Erstellung der Datenbankmanagerinstanz oder bei einem Upgrade von IBM Data Server ruft der Datenbankmanager das Script **setup_db2.sh** automatisch auf, um die DB2 ACS-Funktionalität zu aktivieren. Auf AIX-Systemen wird auch die in-

tegrierte Version von Tivoli Storage FlashCopy Manager automatisch installiert. Auf Linux-Systemen müssen Sie das Setup-Script explizit aufrufen, um Tivoli Storage FlashCopy Manager zu installieren.

Sie können DB2 ACS auch manuell aktivieren.

Vorgehensweise

Führen Sie zum manuellen Aktivieren von DB2 ACS das Script **setup_db2.sh** als Benutzer mit Rootberechtigung und mit den für die Aktivierung von DB2 ACS erforderlichen Parametern aus.
Weitere Informationen zum Script **setup_db2.sh** finden Sie in „Script 'setup_db2.sh'" auf Seite 484.

Ergebnisse

Ein wichtiges Ergebnis bei der Ausführung des Scripts **setup_db2.sh** ist die Überprüfung des Eigentumsrechts und der Berechtigungen für die ausführbaren Dateien von DB2 ACS im Verzeichnis `sqllib/acs`.

Nächste Schritte

Nach der Aktivierung von DB2 ACS müssen Sie DB2 ACS konfigurieren, bevor Sie Momentaufnahmebackup-Operationen ausführen können.

Konfigurieren von DB2 Advanced Copy Services (ACS)

Bevor Sie mit DB2 Advanced Copy Services (ACS) ein Momentaufnahmebackup durchführen können, müssen Sie DB2 ACS konfigurieren. Zum Konfigurieren von DB2 ACS werden Konfigurationsdateien verwendet.

Vorbereitende Schritte

Vor dem Konfigurieren von DB2 ACS müssen Sie die folgenden Tasks ausführen:

1. Installieren Sie DB2 ACS. Weitere Informationen finden Sie in „Installieren von DB2 Advanced Copy Services (ACS)" auf Seite 481.
2. Erstellen Sie die Datenbankmanagerinstanz(en), für die Sie DB2 ACS verwenden möchten..
3. Aktivieren Sie DB2 ACS. Weitere Informationen finden Sie in „Manuelles Aktivieren von DB2 Advanced Copy Services (ACS)" auf Seite 482.

Vorgehensweise

Führen Sie das Script **setup_db2.sh** im Verzeichnis `sqllib/acs` ohne Angabe von Parametern aus. Daraufhin wird ein interaktiver, textbasierter Assistent aufgerufen, der DB2 ACS konfiguriert. Der Assistent erstellt eine Konfigurationsprofildatei und ändert `/etc/initab` auf der Maschine, um den Start des DB2 ACS-Dämons auszulösen.

Konfigurieren des DB2 ACS-Verzeichnisses

Bei der Erstellung einer neuen Datenbankmanagerinstanz wird das Verzeichnis `acs` im Verzeichnis `sqllib` der neuen Instanz erstellt. DB2 Advanced Copy Services (ACS) verwendet das Verzeichnis `acs` zum Speichern von Konfigurationsdateien wie beispielsweise der Steuerdatei des Zieldatenträgers oder des gemeinsamen Repositorys für Recoveryobjekte.

Die Änderung oder Konfiguration des Verzeichnisses acs unterliegt gewissen Einschränkungen.

Informationen zu diesem Vorgang

1. Das Verzeichnis acs darf nicht an DB2 ACS- oder Momentaufnahmebackup-Operationen beteiligt sein.
2. Das Verzeichnis acs kann für ein Momentaufnahmebackup unter Verwendung von IBM Tivoli Storage Manager (TSM) auf allen Datenbankpartitionen und auf dem Backup-System in ein NFS exportiert und dort gemeinsam genutzt werden.

Script 'setup_db2.sh'

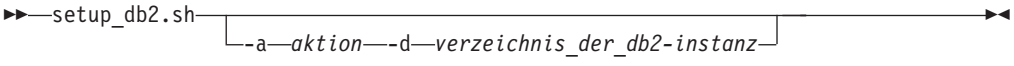
Das Script **setup_db2.sh** aktiviert und konfiguriert DB2 ACS (Advanced Copy Services, erweiterte Kopierservices) und führt eine manuelle Installation von Tivoli Storage FlashCopy Manager durch.

Position

Das Script **setup_db2.sh** befindet sich im Verzeichnis sqllib/acs.

Syntax

Die Syntax für das Script **setup_db2.sh** lautet wie folgt:



Dabei kann *aktion* eine der folgenden Aktionen sein:

disable
Diese Option stoppt Tivoli Storage FlashCopy Manager und entfernt alle Einträge aus der Position /etc/inittab. Um diese Option verwenden zu können, müssen Sie die Rootberechtigung besitzen oder der Instanzeigner sein.

install
Diese Option installiert Tivoli Storage FlashCopy Manager. Um diese Option verwenden zu können, müssen Sie die Rootberechtigung besitzen.

start
Diese Option startet eine zuvor installierte und konfigurierte Version von Tivoli Storage FlashCopy Manager. Um diese Option verwenden zu können, müssen Sie die Rootberechtigung besitzen oder der Instanzeigner sein.

stop
Diese Option stoppt die gegenwärtig aktive Version von Tivoli Storage FlashCopy Manager. Um diese Option verwenden zu können, müssen Sie die Rootberechtigung besitzen oder der Instanzeigner sein.

Verwendung

Bei der Erstellung der Datenbankmanagerinstanz oder bei einem Upgrade von IBM Data Server ruft der Datenbankmanager das Script **setup_db2.sh** automatisch auf, um die DB2 ACS-Funktionalität zu aktivieren. Auf AIX-Systemen wird auch die integrierte Version von Tivoli Storage FlashCopy Manager automatisch installiert. Auf Linux-Systemen müssen Sie das Setup-Script explizit aufrufen, um Tivoli Storage FlashCopy Manager zu installieren.

Sie können das Script **setup_db2.sh** manuell aufrufen, um die folgenden Tasks auszuführen:

Aktivieren von DB2 ACS

Sie können DB2 ACS aktivieren, indem Sie das Script **setup_db2.sh** als Benutzer mit Rootberechtigung unter Verwendung der zuvor beschriebenen Parameter ausführen.

Konfigurieren von DB2 ACS

Sie können DB2 ACS konfigurieren, indem Sie das Script **setup_db2.sh** ohne Parameter ausführen. Falls Sie das Script **setup_db2.sh** ohne Parameter ausführen, werden Sie von einem Assistenten durch die Konfiguration von DB2 ACS geführt.

Installieren von Tivoli Storage FlashCopy Manager

Auf Linux-Systemen müssen Sie das Script **setup_db2.sh** manuell ausführen, um Tivoli Storage FlashCopy Manager zu installieren.

Ein wichtiges Ergebnis bei der Ausführung des Scripts **setup_db2.sh** ist die Überprüfung des Eigentumsrechts und der Berechtigungen für die ausführbaren Dateien von DB2 ACS im Verzeichnis `sqllib/acs`.

Deinstallieren von DB2 Advanced Copy Services (ACS)

DB2 ACS wird automatisch deinstalliert, wenn Sie das DB2-Produkt deinstallieren. Ab Version 9.7 Fixpack 2 können Sie DB2 ACS auch allein deinstallieren, indem Sie den Befehl **db2_deinstall**, den **DB2-Installationsassistenten** oder eine Antwortdatei verwenden.

Vorgehensweise

Deinstallieren Sie DB2 ACS mit einer der folgenden Methoden:

- Geben Sie den Befehl **db2_deinstall** mit dem Parameter **-F ACS** wie folgt ein:
  ```
  db2_deinstall -F ACS
  ```
- Klicken Sie im **DB2-Installationsassistenten** auf **Mit Vorhandenen arbeiten** und entfernen Sie die Auswahl der DB2 ACS-Komponente aus der bereits installierten DB2-Kopie.
- Fügen Sie das Schlüsselwort **ACS** wie folgt zu Ihrer Antwortdatei hinzu:
  ```
  REMOVE_COMP = ACS
  ```

Nächste Schritte

Überprüfen Sie die Nachrichten in der Protokolldatei. Die Protokolldatei befindet sich in den folgenden Verzeichnissen:

- Für Installation mit Rootberechtigung: `/tmp/db2_deinstall.log.`*`prozess-ID`*. Dabei steht *prozess-ID* für die Prozess-ID des DB2-Installationsprogramms.
- Für Installationen ohne Rootberechtigung: `/tmp/db2_deinstall_`*`benutzer-ID`*`.log`. Dabei steht *benutzer-ID* für die Benutzer-ID, die der Eigner der Installation ohne Rootberechtigung ist.

Stellen Sie sicher, dass DB2 ACS entfernt wird, indem Sie den Befehl **db2ls** wie folgt verwenden, um alle installierten Komponenten aufzulisten:

```
db2ls -q -a -b basisinstallationspfad
```

Dabei steht *basisinstallationspfad* für das Verzeichnis, das Sie gerade abfragen.

Manuelles Installieren von Tivoli Storage FlashCopy Manager (Linux)

Die Version von Tivoli Storage FlashCopy Manager, die in Ihrem IBM Produkt integriert ist, muss unter Linux-Betriebssystemen manuell installiert werden. Bei AIX-Betriebssystemen wird sie automatisch installiert.

Vorbereitende Schritte

Zur Installation von Tivoli Storage FlashCopy Manager müssen Sie die Rootberechtigung besitzen. Darüber hinaus muss die DB2-Instanz, mit der Sie DB2 ACS verwenden wollen, bereits erstellt worden sein.

Informationen zu diesem Vorgang

Diese Task beschreibt, wie Sie die spezifischen Tivoli Storage FlashCopy Manager-Pakete für Linux erhalten und das Setup-Script aufrufen, das die Binärdateien in das instanzspezifische Installationsverzeichnis kopiert und die geeigneten Zugriffsrechte für die Binärdateien definiert.

Vorgehensweise

1. Suchen Sie auf der separaten CD mit der Beschriftung `fcm_linux` nach den Linux-Paketen für Tivoli Storage FlashCopy Manager. Alternativ können Sie die Pakete von `fcm_linux` auch auf derselben Website herunterladen, auf der Sie das Produktimage erhalten haben.
2. Dekomprimieren Sie die Pakete im ACS-Verzeichnis des Installationspfads. Beispiel: `/opt/IBM/db2/V10.1/acs`.
3. Rufen Sie das Setup-Script **setup_db2.sh** folgendermaßen auf:

   ```
   setup_db2.sh -a install -d instanzverzeichnis
   ```

Ergebnisse

Sie sollten nun in der Lage sein, Momentaufnahmebackups durchzuführen. Bitte beachten Sie, dass die im DB2-Produkt integrierte Version von Tivoli Storage FlashCopy Manager im Vergleich zur Vollversion des Produkts, die mit dem Produkt 'IBM Tivoli Storage Manager' ausgeliefert wird, einigen Einschränkungen unterliegt.

DB2 ACS-API (Advanced Copy Services-API)

Die DB2 ACS-API (ACS = Advanced Copy Services, erweiterte Kopierservices) definiert eine Reihe von Funktionen, die der Datenbankmanager zur Kommunikation mit der Speicherhardware verwendet, um Momentaufnahmebackup-Operationen auszuführen.

Zur Ausführung von Backup- und Restoreoperationen für Momentaufnahmen benötigen Sie einen DB2 ACS-API-Treiber für Ihre Speichereinheit. Eine Liste der unterstützten Speicherhardware für den integrierten Treiber enthält die Tivoli-Dokumentation im Abschnitt Unterstützte Speichersubsysteme.

DB2 ACS-API-Funktionen

Der Datenbankmanager übermittelt DB2 ACS-Anforderungen mithilfe der DB2 ACS-API-Funktionen an die Speicherhardware.

db2ACSQueryApiVersion - Aktuelle Version der DB2 ACS-API zurückgeben

Gibt die aktuelle Version der DB2 ACS-API zurück.

API-Include-Datei

```
db2ACSApi.h
```

API- und Datenstruktursyntax

```
db2ACS_Version db2ACSQueryApiVersion();
```

Parameter

Keine

Hinweise zur Verwendung

Mögliche Rückgabewerte:

- `DB2ACS_API_VERSION1`
- `DB2ACS_API_VERSION_UNKNOWN`

db2ACSInitialize - DB2 ACS-Sitzung initialisieren

Initialisiert eine neue DB2 ACS-Sitzung. Dieser Aufruf stellt die Kommunikation zwischen der DB2 ACS-Bibliothek des Datenbankmanagers und dem DB2 ACS-API-Treiber für die Speicherhardware her.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
/* ============================================================================
 * Session Initialization
 * ============================================================================ */
db2ACS_RC db2ACSInitialize(
               db2ACS_CB              * pControlBlock,
               db2ACS_ReturnCode      * pRC );
```

Parameter

pControlBlock

Datentyp: `db2ACS_CB *`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Vor dem Aufruf von db2ACSInitialize() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->session
pControlBlock->options

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 19. Rückkehrcodes

Rückkehrcode	Beschreibung	Anmerkungen
DB2ACS_RC_OK	Die Operation war erfolgreich.	
DB2ACS_RC_INIT_FAILED	Der Datenbankmanager hat versucht, eine DB2 ACS-Sitzung zu initialisieren, aber die Initialisierung ist fehlgeschlagen.	
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_INV_DEV_HANDLE	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_DEV_ERROR	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_IO_ERROR	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_COMM_ERROR	Es ist ein Kommunikationsfehler mit einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_NO_DEV_AVAIL	Derzeit ist keine Speichereinheit, wie z. B. ein Bandlaufwerk, verfügbar.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.

- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

Bevor der Datenbankmanager DB2 ACS-API-Aufrufe ausführen kann (Ausnahme: Aufrufe an db2ACSQueryAPIVersion()), muss der Datenbankmanager db2ACSInitialize() aufrufen. Nach der Herstellung einer DB2 ACS-Sitzung durch Aufruf von db2ACSInitialize() kann der Datenbankmanager in DB2 ACS beliebige Kombinationen von Abfrage-, Lese-, Schreib- oder Löschoperationen ausführen. Der Datenbankmanager kann die DB2 ACS-Sitzung durch Aufruf von db2ACSTerminate() beenden.

db2ACSTerminate - DB2 ACS-Sitzung beenden

Beendet eine DB2 ACS-Sitzung.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
/* ============================================================================
 * Session Termination
 * ============================================================================ */
db2ACS_RC db2ACSTerminate(
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

Parameter

pControlBlock

Datentyp: `db2ACS_CB *`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Der Datenbankmanager ordnet den Speicher für diesen Parameter vor dem Aufruf von db2ACSInitialize() zu. Der Datenbankmanager ist für die Freigabe dieses Speichers nach Ausführung von db2ACSTerminate() verantwortlich.

Vor dem Aufruf von db2ACSTerminate() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->options

Es besteht die Möglichkeit, dass der DB2 ACS-API-Treiber den Speicher in `pControlBlock->vendorInfo.vendorCB` inaktiviert und freigibt.

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 20. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
DB2ACS_RC_OK	Die Operation war erfolgreich.	Geben Sie den Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation.
DB2ACS_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

Der DB2 ACS-API-Treiber muss den gesamten Speicher freigeben, den er für die DB2 ACS-Sitzung in db2ACSTerminate() zugeordnet hat.

Unabhängig davon, ob db2ACSTerminate() fehlerfrei beendet wird, kann der Datenbankmanager in dieser DB2 ACS-Sitzung keine DB2 ACS-Funktionen mehr aufrufen, ohne zuvor db2ACSInitialize() aufzurufen.

db2ACSPrepare - Ausführung einer Momentaufnahmebackup-Operation vorbereiten

Bei der Durchführung eines Momentaufnahmebackups wird die Datenbank vom Datenbankmanager ausgesetzt. db2ACSPrepare() führt alle Schritte zur Vorbereitung einer Momentaufnahmebackup-Operation bis zu dem Punkt aus, an dem der Datenbankmanager die Datenbank aussetzt.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
/* ============================================================================
 * Prepare
 * ============================================================================ */
db2ACS_RC db2ACSPrepare(
```

```
db2ACS_GroupList        * pGroupList,
db2ACS_CB               * pControlBlock,
db2ACS_ReturnCode       * pRC );
```

Parameter

pGroupList

Datentyp: `db2ACS_GroupList *`

`db2ACS_GroupList` enthält eine Liste der Gruppen, die in die Momentaufnahmebackup-Operation eingefügt werden sollen.

Wenn **pGroupList** `NULL` ist, werden alle Gruppen (Pfade) in die Momentaufnahmebackup-Operation eingefügt.

Wenn **pGroupList** nicht `NULL` ist:

- **pGroupList** enthält eine Liste der Gruppen (Pfade), die in die Momentaufnahmebackup-Operation eingefügt werden sollen.
- Der Datenbankmanager ist für die Zuordnung und Freigabe des Speichers für **pGroupList** verantwortlich.
- Der Datenbankmanager füllt vor der Übergabe von **pGroupList** an db2ACSPrepare() die folgenden Felder:

  pGroupList->numGroupID
  pGroupList->id

pControlBlock

Datentyp: `db2ACS_CB *`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSPrepare() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 21. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
`DB2ACS_RC_OK`	Die Operation war erfolgreich.	

Tabelle 21. Rückkehrcodes (Forts.)

Rückkehrcode	Beschreibung	Hinweise
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_INV_DEV_HANDLE	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_DEV_ERROR	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_IO_ERROR	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

Wenn db2ACSPrepare() erfolgreich ist, setzt der Datenbankmanager die Datenbank aus, bevor db2ACSSnapshot() aufgerufen wird.

db2ACSBeginOperation - DB2 ACS-Operation starten

Startet eine DB2 ACS-Operation.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
/* ============================================================================
 * Operation Begin
 *
 * A valid ACS operation is specified by passing an ObjectType OR'd with one of
 * the following Operations, such as:
 *
 *    (DB2ACS_OP_CREATE | DB2ACS_OBJTYPE_SNAPSHOT)
 * ============================================================================ */
db2ACS_RC db2ACSBeginOperation(
```

```
db2ACS_Operation          operation,
db2ACS_CB              * pControlBlock,
db2ACS_ReturnCode      * pRC );
```

Parameter

operation

Datentyp: `db2ACS_Operation`

operation ist eine Bitmaske, die die zu startende DB2 ACS-Operation sowie den beteiligten Objekttyp angibt.

Operationstypen:

DB2ACS_OP_CREATE
DB2ACS_OP_READ
DB2ACS_OP_DELETE

Objekttypen:

DB2ACS_OBJTYPE_BACKUP
DB2ACS_OBJTYPE_LOG
DB2ACS_OBJTYPE_LOADCOPY
DB2ACS_OBJTYPE_SNAPSHOT

Beispiel: `( DB2ACS_OP_CREATE | DB2ACS_OBJTYPE_SNAPSHOT )` oder `( DB2ACS_OP_DELETE | DB2ACS_OBJTYPE_LOADCOPY )`.

Der Datenbankmanager übergibt **operation** an den Funktionsaufruf db2ACSBeginOperation().

pControlBlock

Datentyp: `db2ACS_CB *`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSBeginOperation() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

Wenn **operation** den Wert `DB2ACS_OP_CREATE` oder `DB2ACS_OP_READ` aufweist, füllt der Datenbankmanager außerdem das folgende Feld:

pControlBlock->operation

Die in `pControlBlock->operation` enthaltenen Informationen sind nur im Kontext einer bestimmten DB2 ACS-Operation gültig. `pControlBlock->operation` wird während der Ausführung von db2ACSBeginOperation() gesetzt und bleibt unverändert, bis die Rückgabe von db2ACSEndOperation() erfolgt. Der Datenbankmanager und der DB2 ACS-API-Treiber sollten außerhalb des Kontexts einer DB2 ACS-Operation nicht auf `pControlBlock->operation` verweisen.

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der

Inhalt des Parameters db2ACS_ReturnCode für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 22. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
DB2ACS_RC_OK	Die Operation war erfolgreich.	
DB2ACS_RC_INV_OPTIONS	Der Datenbankmanager hat ungültige Optionen angegeben.	
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

Keine

db2ACSEndOperation - DB2 ACS-Operation beenden

Beendet eine DB2 ACS-Operation.

Include-Datei

db2ACSApi.h

Syntax und Datenstrukturen

```
/* ============================================================================
 * Operation End
 * ============================================================================ */
db2ACS_RC db2ACSEndOperation(
               db2ACS_EndAction          endAction,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

Parameter

endAction

Datentyp: `db2ACS_EndAction`

endAction ist eine Bitmaske, die angibt, wie der DB2 ACS-API-Treiber die DB2 ACS-Operation beenden soll.

Werte:

DB2ACS_END_COMMIT
DB2ACS_END_ABORT

Der Datenbankmanager übergibt **endAction** an den Funktionsaufruf db2ACSEndOperation().

pControlBlock

Datentyp: `db2ACS_CB`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSEndOperation() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 23. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
`DB2ACS_RC_OK`	Die Operation war erfolgreich.	
`DB2ACS_RC_INV_ACTION`	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_COMMIT_FAILED`	Der DB2 ACS-API-Treiber konnte eine Transaktion nicht festschreiben.	
`DB2ACS_RC_ABORT_FAILED`	Der Datenbankmanager hat versucht, eine DB2 ACS-Operation abzubrechen, aber der Versuch ist fehlgeschlagen.	

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

Wenn der Datenbankmanager `DB2ACS_END_ABORT` als Parameter **`endAction`** übergibt, sollten die Momentaufnahmebackup-Objekte gelöscht werden.

db2ACSBeginQuery - Abfrage zu Momentaufnahmebackup-Objekten starten

Startet eine DB2 ACS-Abfrageoperation für Momentaufnahmebackup-Objekte, die für Restoreoperationen verfügbar sind.

Include-Datei

`db2ACSApi.h`

Syntax und Datenstrukturen

```
db2ACS_RC db2ACSBeginQuery(
              db2ACS_QueryInput       * pQueryInput,
              db2ACS_CB               * pControlBlock,
              db2ACS_ReturnCode       * pRC );
```

Parameter

pQueryInput

Datentyp: `db2ACS_QueryInput *`

`db2ACS_QueryInput` verfügt über dieselben Felder wie `db2ACS_ObjectInfo`. `db2ACS_ObjectInfo` enthält Informationen zu einem Objekt, das mit der DB2 ACS-API erstellt wurde.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Vor dem Aufruf von db2ACSBeginQuery() füllt der Datenbankmanager die Felder von **`pQueryInput`**.

Der DB2 ACS-API-Treiber muss die Verwendung der folgenden Platzhalterzeichen in der Abfrage unterstützen:

- `DB2ACS_WILDCARD` in Feldern für Zeichenfolgen
- `DB2ACS_ANY_PARTITIONNUM` in Feldern für Datenbankpartitionen
- `DB2ACS_ANY_UINT32` in Feldern mit 32 Bit großen ganzen Zahl ohne Vorzeichen (Uint32)

pControlBlock

Datentyp: `db2ACS_CB *`

db2ACS_CB enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSBeginQuery() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: db2ACS_ReturnCode *

db2ACS_ReturnCode enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters db2ACS_ReturnCode für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 24. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
DB2ACS_RC_OK	Die Operation war erfolgreich.	
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_INV_DEV_HANDLE	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_DEV_ERROR	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_IO_ERROR	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

db2ACSBeginQuery() gibt keine Abfragedaten zurück.

db2ACSGetNextObject - Nächstes für Restore verfügbares Momentaufnahmebackup-Objekt auflisten

Gibt das nächste Element in einer Liste mit Momentaufnahmebackup-Objekten zurück, die für Restoreoperationen verwendet werden können.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
db2ACS_RC db2ACSGetNextObject(
              db2ACS_QueryOutput      * pQueryOutput,
              db2ACS_CB               * pControlBlock,
              db2ACS_ReturnCode       * pRC );
```

Parameter

pQueryOutput
Datentyp: `db2ACS_QueryOutput *`

`db2ACS_QueryOutput` enthält Abfrageergebnisse für Momentaufnahmebackup-Objekte.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pQueryOutput**.

pControlBlock
Datentyp: `db2ACS_CB *`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSGetNextObject() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 25. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
DB2ACS_RC_OK	Die Operation war erfolgreich.	
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_INV_DEV_HANDLE	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_DEV_ERROR	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_IO_ERROR	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_OBJ_NOT_FOUND	Der DB2 ACS-API-Treiber konnte das vom Datenbankmanager angegebene Momentaufnahmebackup-Objekt nicht finden.	Der Funktionsaufruf ist nicht fehlgeschlagen, aber es sind keine Momentaufnahmebackup-Objekte vorhanden, die mit den an db2ACSBeginQuery() übergebenen Bedingungen übereinstimmen.
DB2ACS_RC_END_OF_DATA	Der DB2 ACS-API-Treiber kann keine weiteren Momentaufnahmebackup-Objekte finden.	Der Funktionsaufruf ist nicht fehlgeschlagen, aber es sind keine weiteren Momentaufnahmebackup-Objekte vorhanden, die mit den an db2ACSBeginQuery() übergebenen Bedingungen übereinstimmen.
DB2ACS_RC_MORE_DATA	Es sind weitere Daten zur Übertragung von der Speicherposition an den Datenbankmanager vorhanden.	Es werden Informationen zu einem Momentaufnahmebackup-Objekt zurückgegeben, die mit den an db2ACSBeginQuery() übergebenen Bedingungen übereinstimmen; darüber hinaus sind weitere Momentaufnahmebackup-Objekte vorhanden, die mit den an db2ACSBeginQuery() übergebenen Bedingungen übereinstimmen.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

Der Datenbankmanager muss db2ACSBeginQuery() aufrufen, bevor db2ACSGetNextObject() aufgerufen wird. Der Datenbankmanager gibt die Suchbedingungen im Parameter `db2ACS_QueryInput` an, der an db2ACSBeginQuery() übergeben wird.

db2ACSGetNextObject() gibt Informationen zu einem Momentaufnahmebackup-Objekt zurück, das mit den an db2ACSBeginQuery() übergebenen Suchbedingungen übereinstimmt. Wenn db2ACSGetNextObject() den Rückkehrcode `DB2ACS_RC_MORE_DATA` zurückgibt, kann der Datenbankmanager db2ACSGetNextObject() erneut aufrufen, um Informationen zu einem weiteren Momentaufnahmebackup-Objekt zu erhalten, das mit den Suchbedingungen übereinstimmt. Wenn db2ACSGetNextObject() den Rückkehrcode `DB2ACS_RC_END_OF_DATA` zurückgibt, sind keine weiteren Momentaufnahmebackup-Objekte vorhanden, die mit den Suchbedingungen übereinstimmen.

db2ACSEndQuery - Abfrage zu Momentaufnahmebackup-Objekten beenden

Der Datenbankmanager verwendet die DB2 ACS-API-Funktionen db2ACSBeginQuery() und db2ACSGetNextObject() für Abfragen zu Momentaufnahmebackup-Objekten, die für Restoreoperation verfügbar sind. db2ACSEndQuery() beendet die DB2 ACS-Abfragesitzung.

Include-Datei

`db2ACSApi.h`

Syntax und Datenstrukturen

```
db2ACS_RC db2ACSEndQuery(
              db2ACS_CB              * pControlBlock,
              db2ACS_ReturnCode      * pRC );
```

Parameter

pControlBlock

Datentyp: `db2ACS_CB *`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSEndQuery() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 26. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
DB2ACS_RC_OK	Die Operation war erfolgreich.	
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_INV_DEV_HANDLE	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_DEV_ERROR	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_IO_ERROR	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)” auf Seite 527.

Hinweise zur Verwendung

Der Datenbankmanager kann db2ACSGetNextObject() in dieser DB2 ACS-Sitzung erst dann erneut aufrufen, wenn zuvor db2ACSBeginQuery() aufgerufen wurde.

db2ACSSnapshot - DB2 ACS-Operation ausführen

Führt eine DB2 ACS-Operation aus.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
typedef union db2ACS_ReadList
{
   db2ACS_GroupList           group;
} db2ACS_ReadList;


db2ACS_RC db2ACSSnapshot(
               db2ACS_Action            action,
               db2ACS_ObjectID          objectID,
               db2ACS_ReadList        * pReadList,
               db2ACS_CB              * pControlBlock,
               db2ACS_ReturnCode      * pRC );
```

Parameter

action Datentyp: db2ACS_Action

Der Typ der auszuführenden DB2 ACS-Aktion. Werte:

DB2ACS_ACTION_WRITE
DB2ACS_ACTION_READ_BY_OBJECT
DB2ACS_ACTION_READ_BY_GROUP

Der Datenbankmanager übergibt **action** an db2ACSSnapshot().

objectID

Datentyp: db2ACS_ObjectID

db2ACS_ObjectID gibt eine eindeutige ID für jedes gespeicherte Objekt an, die von einer Abfrage an das Speicherrepository zurückgegeben wird. Der Wert für db2ACS_ObjectID ist in jedem Falle eindeutig und nur für den Zeitraum einer DB2 ACS-Sitzung vorhanden.

Wenn der Datenbankmanager im Aufruf an db2ACSBeginOperation() als Operation DB2ACS_OP_READ oder DB2ACS_OP_DELETE angegeben hat, übergibt er den Wert für die Objekt-ID (**objectID**) an db2ACSSnapshot().

pReadList

Datentyp: db2ACS_ReadList *

db2ACS_ReadList enthält eine Liste mit Gruppen.

pReadList wird nur dann verwendet, wenn **action** den Wert DB2ACS_ACTION_READ_BY_GROUP aufweist.

Wenn **action** den Wert DB2ACS_ACTION_READ_BY_GROUP aufweist, ist der Datenbankmanager vor dem Aufruf von db2ACSSnapshot() für die Zuordnung des Speichers und das Füllen der Felder für **pReadList** und danach für die Freigabe des Speichers für **pReadList** verantwortlich.

pControlBlock

Datentyp: db2ACS_CB *

db2ACS_CB enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSSnapshot() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 27. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
`DB2ACS_RC_OK`	Die Operation war erfolgreich.	
`DB2ACS_RC_INV_ACTION`	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_INV_DEV_HANDLE`	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_DEV_ERROR`	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_IO_ERROR`	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)” auf Seite 527.

Hinweise zur Verwendung

Der Datenbankmanager ruft db2ACSBeginOperation() auf, bevor db2ACSPartition(), db2ACSPrepare() und db2ACSSnapshot() aufgerufen werden. Der Datenbankmanager gibt den Typ der DB2 ACS-Operation, die der DB2 ACS-

API-Treiber ausführen soll, im Parameter **operation** des Aufrufs an db2ACSBeginOperation() an.

db2ACSPartition - Zieldaten für eine Datenbankpartition gruppieren

Ordnet jedem Pfad, den der Datenbankmanager als einer Datenbankpartition zugehörig auflistet, eine Gruppen-ID zu.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
/* ============================================================================
 * Partition
 * ============================================================================ */
db2ACS_RC db2ACSPartition(
               db2ACS_PathList         * pPathList,
               db2ACS_CreateObjectInfo * pCreateObjInfo,
               db2ACS_CB               * PControlBlock,
               db2ACS_ReturnCode       * pRC );
```

Parameter

pPathList

Datentyp: db2ACS_PathList

db2ACS_PathList enthält eine Liste mit Datenbankpfaden sowie einige zusätzliche Informationen zu den einzelnen Pfaden, die sich auf DB2 ACS-Operationen beziehen.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Das Feld **entry** der Struktur db2ACS_PathList ist ein Array von Elementen des Typs db2ACS_PathEntry. db2ACS_PathEntry enthält Informationen zu einem Datenbankpfad.

Vor dem Aufruf von db2ACSPartition füllt der Datenbankmanager die folgenden Felder für jeden Eintrag db2ACS_PathEntry in **pPathList**:

- **path**
- **type**
- **toBeExcluded**

Jeder Pfad, der vom Datenbankmanager als zur Datenbankpartition gehörig erkannt wird, erhält vom DB2 ACS-API-Treiber eine Gruppen-ID. Der DB2 ACS-API-Treiber füllt vor der Rückgabe das Feld **groupID** für jeden Eintrag db2ACS_PathEntry in **pPathList** aus.

pCreateObjInfo

Datentyp: db2ACS_CreateObjectInfo

db2ACS_CreateObjectInfo enthält Informationen zur Erstellung des DB2 ACS-Backup-Objekts.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der Datenbankmanager füllt vor dem Aufruf von db2ACSPartition die Felder von **pCreateObjInfo**.

pControlBlock
Datentyp: `db2ACS_CB *`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSPartition() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 28. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
`DB2ACS_RC_OK`	Die Operation war erfolgreich.	
`DB2ACS_RC_INIT_FAILED`	Der Datenbankmanager hat versucht, eine DB2 ACS-Sitzung zu initialisieren, aber die Initialisierung ist fehlgeschlagen.	
`DB2ACS_RC_INV_ACTION`	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_INV_DEV_HANDLE`	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_DEV_ERROR`	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_IO_ERROR`	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_OBJ_OUT_OF_SCOPE`	Der Datenbankmanager hat versucht, eine DB2 ACS-Operation für ein Recoveryobjekt auszuführen, das nicht vom DB2 ACS-API-Treiber verwaltet wird.	

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

Die Verarbeitung der Daten auf einer Datenbankpartition durch DB2 Advanced Copy Services erfolgt atomar. Das heißt: Die Daten für eine Datenbankpartition werden unabhängig von anderen Datenbankpartitionen zusammen gesichert oder wiederhergestellt, auch wenn die Aktion Teil einer Operation ist, an der mehrere Datenbankpartitionen beteiligt sind. db2ACSPartition gruppiert die Informationen zu den Datenbankpfaden einer einzelnen Datenbankpartition.

Der Datenbankmanager ruft db2ACSPartition auf, bevor db2ACSSnapshot aufgerufen wird. Der Datenbankmanager listet alle Pfade, die dieser Datenbankpartition zugeordnet sind, im Parameter **pPathList** auf. Der Datenbankmanager kann eine DB2 ACS-Operation für eine Untergruppe der in **pPathList** aufgelisteten Pfade ausführen; dazu muss diese Untergruppe im Parameter **pReadList** angegeben werden, der an db2ACSSnapshot übergeben wird.

db2ACSVerify - Überprüfen, ob eine DB2 ACS-Operation erfolgreich ausgeführt wurde

Überprüft, ob eine DB2 ACS-Operation erfolgreich war.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
/* ============================================================================
 * Verify
 * ============================================================================ */
db2ACS_RC db2ACSVerify(
               db2ACS_PostObjectInfo   * pPostObjInfo,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

Parameter

pPostObjInfo

Datentyp: `db2ACS_PostObjectInfo`

`db2ACS_DB2ID` besteht aus einer Datengruppe, die zum Zeitpunkt der Erstellung eines Momentaufnahmebackup-Objekts nicht bekannt sein kann, aber im Objektrepository verwaltet werden muss.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der Datenbankmanager füllt vor dem Aufruf von db2ACSVerify die Felder von **pCreateObjInfo**. **pPostObjInfo** enthält Informationen, die nach der DB2 ACS-Operation relevant sind. Beispiel: Nach einem erfolgreichen Momentaufnahmebackup enthält **pPostObjInfo** möglicherweise die erste aktive Protokolldatei. Sind nach der DB2 ACS-Operation keine relevanten Daten vorhanden, wird **pPostObjInfo** vom Datenbankmanager auf NULL gesetzt.

pControlBlock
Datentyp: db2ACS_CB *

db2ACS_CB enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSVerify() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: db2ACS_ReturnCode *

db2ACS_ReturnCode enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters db2ACS_ReturnCode für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 29. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
DB2ACS_RC_OK	Die Operation war erfolgreich.	
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_INV_DEV_HANDLE	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_DEV_ERROR	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_IO_ERROR	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

Wenn db2ACSVerify zurückgibt, dass eine Momentaufnahmebackup-Operation erfolgreich war, sind die vom Momentaufnahmebackup generierten Recoveryobjekte für Restoreoperationen verfügbar.

db2ACSDelete - Recoveryobjekte löschen, die mit DB2 ACS erstellt wurden

Löscht Recoveryobjekte, die mithilfe von DB2 Advanced Copy Services (ACS) erstellt wurden.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
/* ============================================================================
 * Delete
 * ============================================================================ */
db2ACS_RC db2ACSDelete(
               db2ACS_ObjectID          objectID,
               db2ACS_CB              * pControlBlock,
               db2ACS_ReturnCode      * pRC );
```

Parameter

objectID

Datentyp: `db2ACS_ObjectID`

`db2ACS_ObjectID` gibt eine eindeutige ID für jedes gespeicherte Objekt an, die von einer Abfrage an das Speicherrepository zurückgegeben wird. Der Wert für `db2ACS_ObjectID` ist in jedem Falle eindeutig und nur für den Zeitraum einer DB2 ACS-Sitzung vorhanden.

Der Datenbankmanager kann mit db2ACSQuery() eine gültige Objekt-ID (**`objectID`**) abrufen und an db2ACSDelete() übergeben.

pControlBlock

Datentyp: `db2ACS_CB *`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSDelete() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 30. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
`DB2ACS_RC_OK`	Die Operation war erfolgreich.	Das angegebene Objekt wird gelöscht. Es können keine weiteren DB2 ACS-Operationen für dieses Objekt ausgeführt werden.
`DB2ACS_RC_DELETE_FAILED`	Der DB2 ACS-API-Treiber konnte die vom Datenbankmanager angegebenen Momentaufnahmebackup-Objekte nicht löschen.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_INV_DEV_HANDLE`	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_DEV_ERROR`	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_IO_ERROR`	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
`DB2ACS_RC_OBJ_NOT_FOUND`	Der DB2 ACS-API-Treiber konnte das vom Datenbankmanager angegebene Momentaufnahmebackup-Objekt nicht finden.	

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)" auf Seite 527.

Hinweise zur Verwendung

Wenn der Datenbankmanager db2ACSDelete aufruft, löscht der DB2 ACS-API-Treiber das über die Objekt-ID identifizierte Recoveryobjekt.

Der Datenbankmanager ruft db2ACSDelete auf, wenn ein Benutzer **db2acsutil** mit dem Parameter DELETE aufruft.

db2ACSStoreMetaData - Metadaten für ein mit DB2 ACS generiertes Recoveryobjekt speichern

Speichert Metadaten zu einem Recoveryobjekt, das mit DB2 Advanced Copy Services (ACS) erstellt wurde.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
db2ACS_RC db2ACSStoreMetaData(
              db2ACS_MetaData         * pMetaData,
              db2ACS_CB               * pControlBlock,
              db2ACS_ReturnCode       * pRC );
```

Parameter

pMetaData

Datentyp: db2ACS_MetaData

db2ACS_MetaData speichert Metadaten zu Momentaufnahmebackups.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Die im Feld **data** von **pMetaData** gespeicherten Metadaten sind interne Daten des Datenbankmanagers, die sich im Laufe der Zeit ändern können; deshalb behandelt der DB2 ACS-API-Treiber diese Daten als binären Datenstrom.

pControlBlock

Datentyp: db2ACS_CB *

db2ACS_CB enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSStoreMetaData() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: db2ACS_ReturnCode *

db2ACS_ReturnCode enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters db2ACS_ReturnCode für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 31. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
DB2ACS_RC_OK	Die Operation war erfolgreich.	
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_INV_DEV_HANDLE	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_DEV_ERROR	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_IO_ERROR	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)” auf Seite 527.

Hinweise zur Verwendung

Eine Momentaufnahmebackup-Operation besteht aus mehreren DB2 ACS-API-Funktionsaufrufen wie db2ACSInitialize, db2ACSBeginOperation, db2ACSPrepare und db2ACSSnapshot. db2ACSStoreMetaData ist ebenfalls Teil der Gesamtoperation. Alle API-Aufrufe (auch db2ACSStoreMetaData) müssen erfolgreich sein, damit die Momentaufnahmebackup-Operation erfolgreich ausgeführt werden kann. Wenn db2ACSStoreMetaData fehlschlägt, kann das von der DB2 ACS-Backup-Operation generierte Recoveryobjekt nicht verwendet werden.

db2ACSRetrieveMetaData - Metadaten zu einem mit DB2 ACS generierten Recoveryobjekt abrufen

Ruft Metadaten zu einem Recoveryobjekt ab, das mit DB2 Advanced Copy Services (ACS) erstellt wurde.

Include-Datei

```
db2ACSApi.h
```

Syntax und Datenstrukturen

```
db2ACS_RC db2ACSRetrieveMetaData(
              db2ACS_MetaData         * pMetaData,
              db2ACS_ObjectID           objectID,
              db2ACS_CB               * pControlBlock,
              db2ACS_ReturnCode       * pRC );
```

Parameter

pMetaData

Datentyp: `db2ACS_MetaData`

`db2ACS_MetaData` speichert Metadaten zu Momentaufnahmebackups.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Die im Feld **data** von **pMetaData** gespeicherten Metadaten sind interne Daten des Datenbankmanagers, die sich im Laufe der Zeit ändern können; deshalb behandelt der DB2 ACS-API-Treiber diese Daten als binären Datenstrom.

objectID

Datentyp: `db2ACS_ObjectID`

`db2ACS_ObjectID` gibt eine eindeutige ID für jedes gespeicherte Objekt an, die von einer Abfrage an das Speicherrepository zurückgegeben wird. Der Wert für `db2ACS_ObjectID` ist in jedem Falle eindeutig und nur für den Zeitraum einer DB2 ACS-Sitzung vorhanden.

Der Datenbankmanager kann mit db2ACSQuery() eine gültige Objekt-ID (**objectID**) abrufen und an db2ACSRetrieveMetaData() übergeben.

pControlBlock

Datentyp: `db2ACS_CB *`

`db2ACS_CB` enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

Vor dem Aufruf von db2ACSRetrieveMetaData() füllt der Datenbankmanager die folgenden Felder:

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC Datentyp: `db2ACS_ReturnCode *`

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

Der Datenbankmanager ordnet den Speicher für diesen Parameter zu und übergibt einen Verweis auf dieses instanziierte Objekt an die Funktion. Der Datenbankmanager ist für die Freigabe des Speichers verantwortlich.

Der DB2 ACS-API-Treiber füllt vor der Rückgabe die Felder von **pRC**.

Rückkehrcodes

Tabelle 32. Rückkehrcodes

Rückkehrcode	Beschreibung	Hinweise
DB2ACS_RC_OK	Die Operation war erfolgreich.	
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_INV_DEV_HANDLE	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_DEV_ERROR	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_IO_ERROR	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.
DB2ACS_RC_OBJ_NOT_FOUND	Der DB2 ACS-API-Treiber konnte das vom Datenbankmanager angegebene Momentaufnahmebackup-Objekt nicht finden.	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt. Der Datenbankmanager kann die DB2 ACS-API-Sitzung nicht verwenden.

Wenn der DB2 ACS-API-Treiber einen Fehler feststellt, bricht der Treiber möglicherweise eine DB2 ACS-Operation ab. Die DB2 ACS-Sitzung kann nur für die folgenden Aktionen verwendet werden:

- Wenn ein Aufruf an db2ACSBeginQuery() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndQuery() aufrufen.
- Wenn ein Aufruf an db2ACSBeginOperation() zuvor erfolgreich war, kann der Datenbankmanager db2ACSEndOperation() aufrufen.
- Wenn ein Aufruf an db2ACSInitialize() zuvor erfolgreich war, kann der Datenbankmanager db2ACSTerminate() aufrufen.

Weitere Informationen zu DB2 ACS-API-Rückkehrcodes finden Sie in „Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)” auf Seite 527.

Hinweise zur Verwendung

Keine

Datenstrukturen der DB2 ACS-API (Advanced Copy Services-API)

Wenn Sie die Funktionen der DB2 ACS-API aufrufen möchten, müssen Sie die Datenstrukturen der DB2 ACS-API verwenden.

db2ACS_BackupDetails - Datenstruktur der DB2 ACS-API

db2ACS_BackupDetails enthält Informationen zu einer Momentaufnahmebackup-Operation.

```
/* ------------------------------------------------------------------------ */
typedef struct db2ACS_BackupDetails
{
   /* A traditional DB2 backup can consist of multiple objects (logical tapes),
    * where each object is uniquely numbered with a non-zero natural number.
    * ------------------------------------------------------------------------ */
   db2Uint32                 sequenceNum;

   char                      imageTimestamp[SQLU_TIME_STAMP_LEN + 1];
} db2ACS_BackupDetails;
```

sequenceNum

Datentyp: db2Uint32

Identifiziert ein Backup-Objekt durch die zugehörige eindeutige Zahl.

imageTimestamp

Datentyp: char[]

Eine Zeichenfolge mit der Länge SQLU_TIME_STAMP_LEN + 1.

db2ACS_CB - Datenstruktur der DB2 ACS-API

db2ACS_CB enthält grundlegende Informationen, die zum Initialisieren und Beenden einer DB2 ACS-Sitzung erforderlich sind.

```
/* ============================================================================
 * DB2 Backup Adapter Control Block
 * ============================================================================ */
typedef struct db2ACS_CB
{
   /* Output:  Handle value for this session.
    * ------------------------------------------------------------------------ */
   db2Uint32                 handle;
   db2ACS_VendorInfo         vendorInfo;

   /* Input fields and parameters.
    * ------------------------------------------------------------------------ */
   db2ACS_SessionInfo        session;
   db2ACS_Options            options;

   /* Operation info is optional, possibly NULL, and is only ever valid
    * within the context of an operation (from call to BeginOperation() until
    * the EndOperation() call returns).
    *
    * The operation info will be present during creation or read operations
    * of snapshot and backup objects.
    * ------------------------------------------------------------------------ */
   db2ACS_OperationInfo    * operation;
} db2ACS_CB;
```

handle

Datentyp: db2Uint32

Eine Kennung, die auf die DB2 ACS-Sitzung verweist.

vendorInfo

Datentyp: db2ACS_VendorInfo

db2ACS_VendorInfo enthält Informationen zum DB2 ACS-API-Treiber.

session

Datentyp: db2ACS_SessionInfo

db2ACS_SessionInfo enthält alle Informationen zur DB2 ACS-Sitzung.

options

Datentyp: db2ACS_Options

db2ACS_Options gibt Optionen an, die für eine DB2 ACS-Operation verwendet werden. Der Inhalt dieser Zeichenfolge bezieht sich auf den DB2 ACS-API-Treiber.

operation

Datentyp: db2ACS_OperationInfo *

db2ACS_OperationInfo enthält Informationen zu einer Momentaufnahmebackup-Operation.

db2ACS_CreateObjectInfo - Datenstruktur der DB2 ACS-API

db2ACS_CreateObjectInfo enthält Informationen zur Erstellung des DB2 ACS-Backup-Objekts.

```
/* ============================================================================
 * Object Creation Parameters.
 * ============================================================================ */
typedef struct db2ACS_CreateObjectInfo
{
   db2ACS_ObjectInfo          object;
   db2ACS_DB2ID               db2ID;

   /* ------------------------------------------------------------------------
    * The following fields are optional information for the database manager
    * to use as it sees fit.
    * ------------------------------------------------------------------------ */

   /* Historically both the size estimate and management
    * class parameters have been used by the TSM client API for traditional
    * backup objects, log archives, and load copies, but not for snapshot
    * backups.
    * ------------------------------------------------------------------------ */
      db2Uint64 sizeEstimate;
   char                       mgmtClass[DB2ACS_MAX_MGMTCLASS_SZ + 1];

   /* The appOptions is a copy of the iOptions field of flags passed to DB2's
    * db2Backup() API when this execution was initiated.  This field will
    * only contain valid data when creating a backup or snapshot object.
    * ------------------------------------------------------------------------ */
   db2Uint32                  appOptions;
} db2ACS_CreateObjectInfo;
```

Objekt

Datentyp: db2ACS_ObjectInfo

db2ACS_ObjectInfo enthält Informationen zu einem Objekt, das mit der DB2 ACS-API erstellt wurde.

db2ID Datentyp: db2ACS_DB2ID

db2ACS_DB2ID identifiziert IBM Data Server.

sizeEstimate

Datentyp: db2Uint64

Eine Schätzung der Größe der Backup-Objekte, die erstellt werden. Diese Schätzung gilt nicht für Protokollarchive, Ladekopien oder Momentaufnahmebackup-Objekte.

mgmtClass

Datentyp: db2ACS_MgmtClass

Eine Zeichenfolge mit der Länge db2ACS_MAX_MGMTCLASS_SZ + 1.

Gilt nicht für Momentaufnahmebackup-Objekte.

appOptions
Datentyp: db2Uint32.

Eine Kopie der Backup-Optionen wird an den Befehl BACKUP übergeben, der das Momentaufnahmebackup initialisiert hat.

db2ACS_DB2ID - Datenstruktur der DB2 ACS-API

db2ACS_DB2ID identifiziert IBM Data Server.

```
/* ============================================================================
 * DB2 Data Server Identifier
 * ============================================================================ */
typedef struct db2ACS_DB2ID
{
   db2Uint32                  version;
   db2Uint32                  release;
   db2Uint32                  level;
   char                       signature[DB2ACS_SIGNATURE_SZ + 1];
} db2ACS_DB2ID;
```

version
Datentyp: db2Uint32

Version von IBM Data Server. Beispiel: 9

release
Datentyp: db2Uint32

Release-Level von IBM Data Server. Beispiel: 5

level Datentyp: db2Uint32

Aktualitäts-ID für IBM Data Server. Beispiel: 0

signature
Datentyp: char[]

Eine Zeichenfolge mit der Länge DB2ACS_SIGNATURE_SZ + 1. Beispiel: SQL09050

db2ACS_GroupList - Datenstruktur der DB2 ACS-API

db2ACS_GroupList enthält eine Liste der Gruppen, die in die Momentaufnahmebackup-Operation eingefügt werden sollen.

```
/* ============================================================================
 * Snapshot Group List
 *
 * This is an array of size 'numGroupIDs', indicating the set of groups that
 * are to be included in the snapshot operation.
 * ============================================================================ */
typedef struct db2ACS_GroupList
{
   db2Uint32                  numGroupIDs;
   db2Uint32                * id;
} db2ACS_GroupList;
```

numGroupIDs
Datentyp: db2Uint32

Die Anzahl der Gruppen im Array **id**.

id Datentyp: db2Uint32 *

Ein Array von Gruppen-IDs. Die identifizierten Gruppen sind die Gruppen (oder Pfadlisten), die in die Momentaufnahmebackup-Operation eingefügt werden sollen.

db2ACS_LoadcopyDetails - Datenstruktur der DB2 ACS-API

db2ACS_LoadcopyDetails enthält Informationen zu einer Ladekopie-Operation.

```
/* ------------------------------------------------------------------------ */
typedef struct db2ACS_LoadcopyDetails
{
   /* Just like the BackupDetails, a DB2 load copy can consist of multiple
    * objects (logical tapes), where each object is uniquely numbered with a
    * non-zero natural number.
    * ------------------------------------------------------------------------ */
   db2Uint32                  sequenceNum;

   char                       imageTimestamp[SQLU_TIME_STAMP_LEN + 1];
} db2ACS_LoadcopyDetails;
```

sequenceNum

Datentyp: db2Uint32

Identifiziert ein Backup-Objekt durch die zugehörige eindeutige Zahl.

imageTimestamp

Datentyp: char[]

Eine Zeichenfolge mit der Länge SQLU_TIME_STAMP_LEN + 1.

db2ACS_LogDetails - Datenstruktur der DB2 ACS-API

db2ACS_LogDetails enthält Informationen, die eine bestimmte Datenbankprotokolldatei identifizieren.

```
/* ------------------------------------------------------------------------ */
typedef struct db2ACS_LogDetails
{
   db2Uint32                  fileID;
   db2Uint32                  chainID;
} db2ACS_LogDetails;
```

fileID Datentyp: db2Uint32

Eine Zahl, die den Dateinamen der Datenbankprotokolldatei darstellt.

chainID

Datentyp: db2Uint32

Eine Zahl, die die Datenbankprotokolldateikette identifiziert, zu der die Datenbankprotokolldatei **fileID** gehört.

db2ACS_ObjectInfo - Datenstruktur der DB2 ACS-API

db2ACS_ObjectInfo enthält Informationen zu einem Objekt, das mit der DB2 ACS-API erstellt wurde.

```
/* ========================================================================
 * Object Description and Associated Information.
 *
 * This structure is used for both input and output, and its contents define
 * the minimum information that must be recorded about any object created
 * through this interface.
 * ======================================================================== */
typedef struct db2ACS_ObjectInfo
{
   db2ACS_ObjectType          type;
      SQL_PDB_NODE_TYPE dbPartitionNum;

   char                       db[SQL_DBNAME_SZ + 1];
   char                       instance[DB2ACS_MAX_OWNER_SZ + 1];
   char                       host[SQL_HOSTNAME_SZ + 1];
   char                       owner[DB2ACS_MAX_OWNER_SZ + 1];

   Union-Verknüpfung   {
```

```
      db2ACS_BackupDetails    backup;
      db2ACS_LogDetails       log;
      db2ACS_LoadcopyDetails  loadcopy;
      db2ACS_SnapshotDetails  snapshot;
   } details;
} db2ACS_ObjectInfo;
```

type Datentyp: db2ACS_ObjectType

Gibt den Typ der Momentaufnahmebackup-Objekte an. Werte:

DB2ACS_OBJTYPE_ALL
DB2ACS_OBJTYPE_BACKUP
DB2ACS_OBJTYPE_LOG
DB2ACS_OBJTYPE_LOADCOPY
DB2ACS_OBJTYPE_SNAPSHOT

DB2ACS_OBJTYPE_ALL kann nur als Filter für Abfragen verwendet werden. Es gibt keine Objekte des Typs 0.

dbPartitionNum
Datentyp: SQL_PDB_NODE_TYPE

Eine Kennung für diese Datenbankpartition.

db Datentyp: char[]

Eine Zeichenfolge mit der Länge SQL_DBNAME_SZ + 1.

instance
Datentyp: char[]

Eine Zeichenfolge mit der Länge DB2ACS_MAX_OWNER_SZ + 1.

host Datentyp: char[]

Eine Zeichenfolge mit der Länge SQL_HOSTNAME_SZ + 1.

owner Datentyp: char[]

Eine Zeichenfolge mit der Länge DB2ACS_MAX_OWNER_SZ + 1.

details

backup
Datentyp: db2ACS_BackupDetails

db2ACS_BackupDetails enthält Informationen zu einer Momentaufnahmebackup-Operation.

log Datentyp: db2ACS_LogDetails

db2ACS_LogDetails enthält Informationen, die eine bestimmte Datenbankprotokolldatei identifizieren.

loadcopy
Datentyp: db2ACS_LoadcopyDetails

db2ACS_LoadcopyDetails enthält Informationen zu einer Ladekopie-Operation.

snapshot
Datentyp: db2ACS_SnapshotDetails

db2ACS_SnapshotDetails enthält Informationen zu einer Momentaufnahmebackup-Operation.

db2ACS_ObjectStatus - Datenstruktur der DB2 ACS-API

db2ACS_ObjectStatus enthält Informationen zum Status oder Fortschritt einer Momentaufnahmebackup-Operation bzw. zum Status oder zur Benutzerfreundlichkeit eines Momentaufnahmebackup-Objekts.

```
typedef struct db2ACS_ObjectStatus
{
   /* The total and completed bytes refer only to the ACS snapshot backup
    * itself, not to the progress of any offloaded tape backup.
    *
    * A bytesTotal of 0 indicates that the progress could not be determined.
    * ------------------------------------------------------------------------ */
   db2Uint64                 bytesCompleted;
   db2Uint64                 bytesTotal;
   db2ACS_ProgressState      progressState;
   db2ACS_UsabilityState     usabilityState;
} db2ACS_ObjectStatus;
```

bytesCompleted

Datentyp: db2Uint64

Der Teil des Momentaufnahmebackups, der bereits abgeschlossen ist (in Byte).

bytesTotal

Datentyp: db2Uint64

Die Größe des abgeschlossenen Momentaufnahmebackups (in Byte).

progressState

Datentyp: db2ACS_ProgressState

Der Status der Momentaufnahmebackup-Operation. Werte:

DB2ACS_PSTATE_UNKNOWN
DB2ACS_PSTATE_IN_PROGRESS
DB2ACS_PSTATE_SUCCESSFUL
DB2ACS_PSTATE_FAILED

usabilityState

Datentyp: db2ACS_UsabilityState

Der Status des Momentaufnahmebackup-Objekts bzw. wie das Momentaufnahmebackup-Objekt verwendet werden kann. Werte:

DB2ACS_USTATE_UNKNOWN
DB2ACS_USTATE_LOCALLY_MOUNTABLE
DB2ACS_USTATE_REMOTELY_MOUNTABLE
DB2ACS_USTATE_REPETITIVELY_RESTORABLE
DB2ACS_USTATE_DESTRUCTIVELY_RESTORABLE
DB2ACS_USTATE_SWAP_RESTORABLE
DB2ACS_USTATE_PHYSICAL_PROTECTION
DB2ACS_USTATE_FULL_COPY
DB2ACS_USTATE_DELETED
DB2ACS_USTATE_FORCED_MOUNT
DB2ACS_USTATE_BACKGROUND_MONITOR_PENDING
DB2ACS_USTATE_TAPE_BACKUP_PENDING
DB2ACS_USTATE_TAPE_BACKUP_IN_PROGRESS
DB2ACS_USTATE_TAPE_BACKUP_COMPLETE

db2ACS_OperationInfo - Datenstruktur der DB2 ACS-API

db2ACS_OperationInfo enthält Informationen zu einer Momentaufnahmebackup-Operation.

```
/* ============================================================================
 * Operation Info
 *
 * The information contained within this structure is only valid within the
 * context of a particular operation.  It will be valid at the time
 * BeginOperation() is called, and will remain unchanged until EndOperation()
 * returns, but must not be referenced outside the scope of an operation.
 * ============================================================================ */
typedef struct db2ACS_OperationInfo
{
   db2ACS_SyncMode            syncMode;

   /* List of database and backup operation partitions.
    *
    * For details, refer to the db2ACS_PartitionList definition.
    * ------------------------------------------------------------------------- */
   db2ACS_PartitionList     * dbPartitionList;
} db2ACS_OperationInfo;
```

syncMode

Datentyp: `db2ACS_SyncMode`

Die Synchronisationsebene zwischen den Backup-Operationen auf unterschiedlichen Datenbankpartitionen.

Werte:

DB2ACS_SYNC_NONE
Für zusammengehörige Operationen auf mehreren Datenbankpartitionen findet keine Synchronisation statt. Wird bei Operationen verwendet, die auf mehreren Datenbankpartitionen ausgeführt werden und keine Synchronisation verwenden.

DB2ACS_SYNC_SERIAL
Wird bei der Ausführung gleichzeitiger Momentaufnahmebackup-Operationen auf mehreren Datenbankpartitionen verwendet. Auf jeder Datenbankpartition wird die Ein-/Ausgabe (E/A) bei Ausgabe der Momentaufnahmebackup-Operation ausgesetzt; anschließend wird die E/A auf diesen Datenbankpartitionen seriell und nicht gleichzeitig fortgesetzt.

SYNC_PARALLEL
Eine Momentaufnahmenoperation wird auf mehreren Partitionen gleichzeitig ausgeführt. Nachdem alle an der Momentaufnahmebackup-Operation beteiligten Datenbankpartitionen die Vorbereitungen für die Momentaufnahmebackup-Operation abgeschlossen haben, wird die E/A auf allen Datenbankpartitionen ausgesetzt. Die verbleibenden Schritte für das Momentaufnahmebackup werden auf allen beteiligten Datenbankpartitionen gleichzeitig ausgeführt.

dbPartitionList

Datentyp: `db2ACS_PartitionList *`

`db2ACS_PartitionList` enthält Informationen zu den Datenbankpartitionen in der Datenbank, die an einer DB2 ACS-Operation beteiligt sind.

db2ACS_Options - Datenstruktur der DB2 ACS-API

`db2ACS_Options` gibt Optionen an, die für eine DB2 ACS-Operation verwendet werden. Der Inhalt dieser Zeichenfolge bezieht sich auf den DB2 ACS-API-Treiber.

```
/* ============================================================================
 * DB2 Backup Adapter User Options
 * ============================================================================ */
```

```
typedef struct db2ACS_Options
{
   db2Uint32                 size;
   void                    * data;
} db2ACS_Options;
```

size Datentyp: db2Uint32

Die Größe des Parameters **data** (in Byte).

data Datentyp: void *

Ein Verweis auf einen Speicherblock, der die Optionen enthält.

db2ACS_PartitionEntry - Datenstruktur der DB2 ACS-API

db2ACS_PartitionEntry ist ein Element von db2ACS_PartitionList.

```
typedef struct db2ACS_PartitionEntry
{
   SQL_PDB_NODE_TYPE          num;
   char                       host[SQL_HOSTNAME_SZ + 1];
} db2ACS_PartitionEntry;
```

num Datentyp: SQL_PDB_NODE_TYPE

Eine Kennung für diesen Datenbankpartitionseintrag.

host Datentyp: char[]

Eine Zeichenfolge mit der Länge SQL_HOSTNAME_SZ + 1.

db2ACS_PartitionList - Datenstruktur der DB2 ACS-API

db2ACS_PartitionList enthält Informationen zu den Datenbankpartitionen in der Datenbank, die an einer DB2 ACS-Operation beteiligt sind.

```
typedef struct db2ACS_PartitionList
{
   db2Uint64                  numPartsInDB;
   db2Uint64                  numPartsInOperation;
   db2ACS_PartitionEntry    * partition;
} db2ACS_PartitionList;
```

numPartsInDB

Datentyp: db2Uint64

Die Anzahl der Datenbankpartitionen in der Datenbank.

numPartsInOperation

Datentyp: db2Uint64

Die Anzahl der Datenbankpartitionen, die an der DB2 ACS-Operation beteiligt sind.

partition

Datentyp: db2ACS_PartitionEntry *

db2ACS_PartitionEntry ist ein Element von db2ACS_PartitionList.

db2ACS_PathEntry - Datenstruktur der DB2 ACS-API

db2ACS_PathEntry enthält Informationen zu einem Datenbankpfad.

```
typedef struct db2ACS_PathEntry
{
   /* INPUT: The path and type will be provided by the database server, as well
    *        as a flag indicating if the path is to be excluded from the backup.
    * ------------------------------------------------------------------------ */
   char                       path[DB2ACS_MAX_PATH_SZ + 1];
   db2ACS_PathType            type;
   db2Uint32                  toBeExcluded;
```

```
   /* OUTPUT: The group ID is to be provided by the backup adapter for use by
    *         the DB2 server.  The group ID will be used during with snapshot
    *         operations as an indication of which paths are dependent and must
    *         be included together in any snapshot operation.  Unique group IDs
    *         indicate that the paths in those groups are independent for the
    *         purposes of snapshot operations.
    * ------------------------------------------------------------------------ */
   db2Uint32                 groupID;
} db2ACS_PathEntry;
```

path Datentyp: `char[]`

Eine Zeichenfolge mit der Länge `DB2ACS_MAX_PATH_SZ + 1`.

type Datentyp: `db2ACS_PathType`

Der Pfadtyp. Werte:

DB2ACS_PATH_TYPE_UNKNOWN
DB2ACS_PATH_TYPE_LOCAL_DB_DIRECTORY
DB2ACS_PATH_TYPE_DBPATH
DB2ACS_PATH_TYPE_DB_STORAGE_PATH
DB2ACS_PATH_TYPE_TBSP_CONTAINER
DB2ACS_PATH_TYPE_TBSP_DIRECTORY
DB2ACS_PATH_TYPE_TBSP_DEVICE
DB2ACS_PATH_TYPE_LOGPATH
DB2ACS_PATH_TYPE_MIRRORLOGPATH

toBeExcluded
Datentyp: `db2Uint32`

Eine Markierung, die angibt, ob der angegebene Pfad in das Momentaufnahmebackup eingefügt werden soll. Werte:

- 0 - Pfad in Momentaufnahmebackup einfügen
- 1 - Pfad nicht in Momentaufnahmebackup einfügen

groupID
Datentyp: `db2Uint32`

Eine Gruppen-ID.

db2ACS_PathList - Datenstruktur der DB2 ACS-API

`db2ACS_PathList` enthält eine Liste mit Datenbankpfaden sowie einige zusätzliche Informationen zu den einzelnen Pfaden, die sich auf DB2 ACS-Operationen beziehen.

```
/* ============================================================================
 * Snapshot File List
 *
 * This is an array of 'numEntries' db2ACS_PathEntry's, where each path entry is
 * a path to some storage on the DB2 server which is in use by the current
 * database.
 * ============================================================================ */

typedef struct db2ACS_PathList
{
   db2Uint32                 numEntries;
   db2ACS_PathEntry        * entry;
} db2ACS_PathList;
```

numEntries
Datentyp: `db2Uint32`

Die Anzahl der Pfadeinträge im Array **entry**.

entry Datentyp: db2ACS_PathEntry

db2ACS_PathEntry enthält Informationen zu einem Datenbankpfad.

db2ACS_PostObjectInfo - Datenstruktur der DB2 ACS-API

db2ACS_DB2ID besteht aus einer Datengruppe, die zum Zeitpunkt der Erstellung eines Momentaufnahmebackup-Objekts nicht bekannt sein kann, aber im Objektrepository verwaltet werden muss.

```
/* ============================================================================
 * The PostObjectInfo is a set of data that can not be known at object
 * creation time, but which must be maintained in the object repository.  This
 * is an optional field on the Verify() call, which may be NULL if there are
 * no post-operation updates to be made.
 * ============================================================================ */
typedef struct db2ACS_PostObjectInfo
{
   /* The first active log will only be valid when creating a backup or
    * snapshot object.  It will indicate the file number and chain id of the
    * first log required for recovery using this object.
    * ------------------------------------------------------------------------ */
   db2ACS_LogDetails          firstActiveLog;
} db2ACS_PostObjectInfo;
```

firstActiveLog

Datentyp: db2ACS_LogDetails

db2ACS_LogDetails enthält Informationen, die eine bestimmte Datenbankprotokolldatei identifizieren.

db2ACS_QueryInput und db2ACS_QueryOutput - Datenstrukturen der DB2 ACS-API

db2ACS_QueryInput enthält Informationen, die ein Objekt identifizieren, das von Ihnen abgefragt wird. db2ACS_QueryOutput enthält Abfrageergebnisse für Momentaufnahmebackup-Objekte.

```
/* ============================================================================
 * Unique Querying.
 *
 * When using this structure as query input, to indicate the
 * intention to supply a 'wildcard' search criteria, DB2 will supply:
 *
 *    -- character strings as "*".
 *    -- numeric values as (-1), cast as the appropriate signed or unsigned
 *       type.
 * ============================================================================ */

typedef struct db2ACS_ObjectInfo db2ACS_QueryInput;

typedef struct db2ACS_QueryOutput
{
   db2ACS_ObjectID            objectID;
   db2ACS_ObjectInfo          object;
   db2ACS_PostObjectInfo      postInfo;
   db2ACS_DB2ID               db2ID;
   db2ACS_ObjectStatus        status;

   /* Size of the object in bytes.
    * ------------------------------------------------------------------------  */
   db2Uint64                  objectSize;

   /* Size of the metadata associated with the object, if any, in bytes.
    * ------------------------------------------------------------------------  */
   db2Uint64                  metaDataSize;

   /* The creation time of the object is a 64bit value with a definition
```

```
    * equivalent to an ANSI C time_t value (seconds since the epoch, GMT).
    *
    * This field is equivalent to the file creation or modification time in
    * a traditional filesystem.  This should be created and stored
    * automatically by the BA subsystem, and a valid time value should be
    * returned with object query results, for all object types.
    * ------------------------------------------------------------------  */
   db2Uint64                createTime;
} db2ACS_QueryOutput;
```

objectID
Datentyp: `db2ACS_ObjectID`

`db2ACS_ObjectID` gibt eine eindeutige ID für jedes gespeicherte Objekt an, die von einer Abfrage an das Speicherrepository zurückgegeben wird. Der Wert für `db2ACS_ObjectID` ist in jedem Falle eindeutig und nur für den Zeitraum einer DB2 ACS-Sitzung vorhanden.

object Datentyp: `db2ACS_ObjectInfo`

`db2ACS_ObjectInfo` enthält Informationen zu einem Objekt, das mit der DB2 ACS-API erstellt wurde.

postInfo
Datentyp: `db2ACS_PostObjectInfo`

`db2ACS_DB2ID` besteht aus einer Datengruppe, die zum Zeitpunkt der Erstellung eines Momentaufnahmebackup-Objekts nicht bekannt sein kann, aber im Objektrepository verwaltet werden muss.

db2ID Datentyp: `db2ACS_DB2ID`

`db2ACS_DB2ID` identifiziert IBM Data Server.

status Datentyp: `db2ACS_ObjectStatus`

`db2ACS_ObjectStatus` enthält Informationen zum Status oder Fortschritt einer Momentaufnahmebackup-Operation bzw. zum Status oder zur Benutzerfreundlichkeit eines Momentaufnahmebackup-Objekts.

objectSize
Datentyp: `db2Uint64`

Die Größe des Objekts (in Byte).

metaDataSize
Datentyp: `db2Uint64`

Die Größe der Metadaten, die gegebenenfalls dem Objekt zugeordnet sind (in Byte).

createTime
Datentyp: `db2Uint64`

Der Zeitpunkt der Erstellung eines Objekts. Der Wert von **`createTime`** entspricht dem ANSI-C-Wert `time_t`.

db2ACS_ReadList - Datenstruktur der DB2 ACS-API

`db2ACS_ReadList` enthält eine Liste mit Gruppen.

```
/* The ReadList will only be used for snapshots where the action is READ, and
 * where one of the granularity modifiers other than BY_OBJ has been specified.
 * In the typical usage scenario of ( READ | BY_OBJ ) the ReadList parameter
 * should be ignored.
 *
 * When the action is DB2ACS_ACTION_BY_GROUP the union is to be interpreted
 * as a group list.
```

```
 * ------------------------------------------------------------------------ */
typedef union db2ACS_ReadList
{
   db2ACS_GroupList           group;
} db2ACS_ReadList;
```

group Datentyp: `db2ACS_GroupList`

`db2ACS_GroupList` enthält eine Liste der Gruppen, die in die Momentaufnahmebackup-Operation eingefügt werden sollen.

db2ACS_ReturnCode - Datenstruktur der DB2 ACS-API

`db2ACS_ReturnCode` enthält Diagnoseinformationen, einschließlich Nachrichtentext und Fehlercodes, die sich auf die Speicherhardware beziehen. Der Inhalt des Parameters `db2ACS_ReturnCode` für einen DB2 ACS-API-Funktionsaufruf wird in den Diagnoseprotokollen des Datenbankmanagers erfasst.

```
/* ========================================================================
 * Storage Adapter Return Code and Diagnostic Data.
 *
 * These will be recorded in the DB2 diagnostic logs, but are intended to be
 * internal return and reason codes from the storage layers which can be used
 * in conjunction with the DB2ACS_RC to provide more detailed diagnostic info.
 * ======================================================================== */
typedef struct db2ACS_ReturnCode
{
   int                        returnCode;
   int                        reasonCode;
   char                       description[DB2ACS_MAX_COMMENT_SZ + 1];
} db2ACS_ReturnCode;
```

returnCode
Datentyp: `int`

Rückkehrcode, der sich auf die Speicherhardware bezieht.

reasonCode
Datentyp: `int`

Ursachencode, der sich auf die Speicherhardware bezieht.

description
Datentyp: `char[]`

Eine Zeichenfolge mit der Länge `DB2ACS_MAX_COMMENT_SZ + 1`.

db2ACS_SessionInfo - Datenstruktur der DB2 ACS-API

`db2ACS_SessionInfo` enthält alle Informationen zur DB2 ACS-Sitzung.

```
/* ========================================================================
 * Session Info
 * ======================================================================== */
typedef struct db2ACS_SessionInfo
{
   db2ACS_DB2ID               db2ID;

   /* Fields identifying the backup session originator.
    * ------------------------------------------------------------------- */
      SQL_PDB_NODE_TYPE dbPartitionNum;
   char                       db[SQL_DBNAME_SZ + 1];
   char                       instance[DB2ACS_MAX_OWNER_SZ + 1];
   char                       host[SQL_HOSTNAME_SZ + 1];
   char                       user[DB2ACS_MAX_OWNER_SZ + 1];
   char                       password[DB2ACS_MAX_PASSWORD_SZ + 1];
```

```
   /* The fully qualified ACS vendor library name to be used.
    * ------------------------------------------------------------------------ */
   char                        libraryName[DB2ACS_MAX_PATH_SZ + 1];
} db2ACS_SessionInfo;
```

db2ID Datentyp: db2ACS_DB2ID

db2ACS_DB2ID identifiziert IBM Data Server.

dbPartitionNum
Datentyp: SQL_PDB_NODE_TYPE

Die eindeutige, numerische Kennung für eine Datenbankpartition.

db Datentyp: char[]

Eine Zeichenfolge mit der Länge SQL_DBNAME_SZ + 1.

instance
Datentyp: char[]

Eine Zeichenfolge mit der Länge DB2ACS_MAX_OWNER_SZ + 1.

host Datentyp: char[]

Eine Zeichenfolge mit der Länge SQL_HOSTNAME_SZ + 1.

user Datentyp: char[]

Eine Zeichenfolge mit der Länge DB2ACS_MAX_OWNER_SZ + 1.

password
Datentyp: char[]

Eine Zeichenfolge mit der Länge DB2ACS_MAX_PASSWORD_SZ + 1.

libraryName
Datentyp: char[]

Eine Zeichenfolge mit der Länge DB2ACS_MAX_PATH_SZ + 1.

db2ACS_SnapshotDetails - Datenstruktur der DB2 ACS-API

db2ACS_SnapshotDetails enthält Informationen zu einer Momentaufnahmebackup-Operation.

```
typedef struct db2ACS_SnapshotDetails
{
   char                        imageTimestamp[SQLU_TIME_STAMP_LEN + 1];
} db2ACS_SnapshotDetails;
```

imageTimestamp
Datentyp: char[]

Eine Zeichenfolge mit der Länge SQLU_TIME_STAMP_LEN + 1.

db2ACS_MetaData - Datenstruktur der DB2 ACS-API

db2ACS_MetaData speichert Metadaten zu Momentaufnahmebackups.

```
/* ============================================================================
 * The metadata structure itself is internal to DB2 and is to be treated by
 * the storage interface as an unstructured block of data of the given size.
 * ========================================================================== */
typedef struct db2ACS_MetaData
{
   db2Uint64                   size;
   void                      * data;
} db2ACS_MetaData;
```

size Datentyp: db2Uint32

Die Größe des Parameters **data** (in Byte).

data Datentyp: `void *`

Ein Verweis auf einen Speicherblock, den der Datenbankmanager zum Speichern von Momentaufnahmebackup-Metadaten verwendet.

db2ACS_VendorInfo - Datenstruktur der DB2 ACS-API

`db2ACS_VendorInfo` enthält Informationen zum DB2 ACS-API-Treiber.

```
/* ============================================================================
 * Storage Vendor Identifier
 * ============================================================================ */
typedef struct db2ACS_VendorInfo
{
   void                    * vendorCB;             /* Vendor control block  */
   db2Uint32                 version;              /* Current version       */
   db2Uint32                 release;              /* Current release       */
   db2Uint32                 level;                /* Current level         */
   char                      signature[DB2ACS_MAX_VENDORID_SZ + 1];
} db2ACS_VendorInfo;
```

vendorCB

Datentyp: `void *`

Verweis auf einen Steuerblock, der sich auf den DB2 ACS-API-Treiber bezieht.

version

Datentyp: `db2Uint32`

Version des DB2 ACS-API-Treibers.

release

Datentyp: `db2Uint32`

Release-Level des DB2 ACS-API-Treibers.

level Datentyp: `db2Uint32`

Aktualitäts-ID des DB2 ACS-API-Treibers.

signature

Datentyp: `db2ACS_VendorSignature`

Eine Zeichenfolge mit der Länge `DB2ACS_MAX_VENDORID_SZ + 1`.

Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)

Die Funktionen der DB2 ACS-API geben eine definierte Gruppe von möglichen Rückkehrcodes zurück.

Tabelle 33. Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API)

Rückkehrcode	Beschreibung
`DB2ACS_RC_OK`	Die Operation war erfolgreich.
`DB2ACS_RC_LINK_EXIST`	Die Sitzung wurde zuvor aktiviert.
`DB2ACS_RC_COMM_ERROR`	Es ist ein Kommunikationsfehler mit einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.
`DB2ACS_RC_INV_VERSION`	Die Version der DB2 ACS-Bibliothek des Datenbankmanagers und die Version des DB2 ACS-API-Treibers sind nicht kompatibel.

Tabelle 33. Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API) (Forts.)

Rückkehrcode	Beschreibung
DB2ACS_RC_INV_ACTION	Der Datenbankmanager hat eine ungültige Aktion vom DB2 ACS-API-Treiber angefordert.
DB2ACS_RC_NO_DEV_AVAIL	Derzeit ist keine Speichereinheit, wie z. B. ein Bandlaufwerk, verfügbar.
DB2ACS_RC_OBJ_NOT_FOUND	Der DB2 ACS-API-Treiber konnte das vom Datenbankmanager angegebene Momentaufnahmebackup-Objekt nicht finden.
DB2ACS_RC_OBJS_FOUND	Der DB2 ACS-API-Treiber hat mehrere Momentaufnahmebackup-Objekte gefunden, die mit der Spezifikation des Datenbankmanagers übereinstimmen.
DB2ACS_RC_INV_USERID	Der Datenbankmanager hat eine ungültige Benutzer-ID an den DB2 ACS-API-Treiber übergeben.
DB2ACS_RC_INV_PASSWORD	Der Datenbankmanager hat ein ungültiges Kennwort an den DB2 ACS-API-Treiber übergeben.
DB2ACS_RC_INV_OPTIONS	Der Datenbankmanager hat ungültige Optionen angegeben.
DB2ACS_RC_INIT_FAILED	Der Datenbankmanager hat versucht, eine DB2 ACS-Sitzung zu initialisieren, aber die Initialisierung ist fehlgeschlagen.
DB2ACS_RC_INV_DEV_HANDLE	Der Datenbankmanager hat eine ungültige Kennung für eine Speichereinheit übergeben.
DB2ACS_RC_BUFF_SIZE	Der Datenbankmanager hat eine ungültige Puffergröße angegeben.
DB2ACS_RC_END_OF_DATA	Der DB2 ACS-API-Treiber kann keine weiteren Momentaufnahmebackup-Objekte finden.
DB2ACS_RC_END_OF_TAPE	Die Speichereinheit hat unerwarteterweise das Ende des Backup-Bandes erreicht.
DB2ACS_RC_DATA_RESEND	Eine Speichereinheit, wie z. B. ein Bandlaufwerk, hat den Datenbankmanager aufgefordert, den neuesten Datenpuffer erneut zu senden.
DB2ACS_RC_COMMIT_FAILED	Der DB2 ACS-API-Treiber konnte eine Transaktion nicht festschreiben.
DB2ACS_RC_DEV_ERROR	Es ist ein Fehler in einer Speichereinheit, wie z. B. einem Bandlaufwerk, aufgetreten.
DB2ACS_RC_WARNING	Die Speicherhardware hat eine Warnung zurückgegeben. Weitere Informationen hierzu finden Sie in den Diagnoseprotokollen des Datenbankmanagers.
DB2ACS_RC_LINK_NOT_EXIST	Die Sitzung wurde zuvor nicht aktiviert.
DB2ACS_RC_MORE_DATA	Es sind weitere Daten zur Übertragung von der Speicherposition an den Datenbankmanager vorhanden.
DB2ACS_RC_ENDOFMEDIA_NO_DATA	Die Speichereinheit hat das Ende des Speicherdatenträgers erreicht und keine Daten gefunden.
DB2ACS_RC_ENDOFMEDIA	Die Speichereinheit hat das Ende des Speicherdatenträgers erreicht.
DB2ACS_RC_MAX_LINK_GRANT	Die maximale Anzahl an Verbindungen wurde hergestellt. Der Datenbankmanager kann keine weiteren Verbindungen herstellen.

Tabelle 33. Rückkehrcodes der DB2 ACS-API (Advanced Copy Services-API) (Forts.)

Rückkehrcode	Beschreibung
DB2ACS_RC_IO_ERROR	Der DB2 ACS-API-Treiber hat einen Fehler festgestellt, der auf Eingabe- oder Ausgabeoperationen zurückzuführen ist.
DB2ACS_RC_DELETE_FAILED	Der DB2 ACS-API-Treiber konnte die vom Datenbankmanager angegebenen Momentaufnahmebackup-Objekte nicht löschen.
DB2ACS_RC_INV_BKUP_FNAME	Der Datenbankmanager hat einen ungültigen Dateinamen für das Momentaufnahmebackup-Objekt angegeben.
DB2ACS_RC_NOT_ENOUGH_SPACE	Der DB2 ACS-API-Treiber geht davon aus, dass nicht genügend Speicherplatz vorhanden ist, um ein Momentaufnahmebackup der vom Datenbankmanager angegebenen Datenbank auszuführen.
DB2ACS_RC_ABORT_FAILED	Der Datenbankmanager hat versucht, eine DB2 ACS-Operation abzubrechen, aber der Versuch ist fehlgeschlagen.
DB2ACS_RC_UNEXPECTED_ERROR	Der DB2 ACS-API-Treiber hat einen schwer wiegenden, unbekannten Fehler festgestellt.
DB2ACS_RC_NO_DATA	Der DB2 ACS-API-Treiber hat keine Daten an den Datenbankmanager zurückgegeben.
DB2ACS_RC_OBJ_OUT_OF_SCOPE	Der Datenbankmanager hat versucht, eine DB2 ACS-Operation für ein Recoveryobjekt auszuführen, das nicht vom DB2 ACS-API-Treiber verwaltet wird.
DB2ACS_RC_INV_CALL_SEQUENCE	Der Datenbankmanager hat DB2 ACS-API-Funktionen in einer ungültigen Sequenz aufgerufen. Der Datenbankmanager muss beispielsweise db2ACSInitialize aufrufen, bevor andere DB2 ACS-API-Funktionen aufgerufen werden können (Ausnahme: db2ACSQueryAPIVersion).
DB2ACS_RC_SHARED_STORAGE_GROUP	Der Datenbankmanager hat versucht, eine Momentaufnahmenoperation für ein Speicherobjekt auszuführen, das von einer anderen Datenbank oder Anwendung verwendet wird.

Teil 3. Anhänge und Schlussteil

Anhang A. Übersicht über technische Informationen zu DB2

Technische Informationen zu DB2 liegen in verschiedenen Formaten vor, die auf unterschiedliche Weise abgerufen werden können.

Die technischen Informationen zu DB2 stehen über die folgenden Tools und Methoden zur Verfügung:

- DB2 Information Center
  - Themen (zu Tasks, Konzepten und Referenzinformationen)
  - Beispielprogramme
  - Lernprogramme
- DB2-Bücher
  - PDF-Dateien (für den Download verfügbar)
  - PDF-Dateien (auf der DB2-PDF-DVD)
  - Gedruckte Bücher
- Hilfe für Befehlszeile
  - Hilfe für Befehle
  - Hilfe für Nachrichten

Anmerkung: Die Themen des DB2 Information Center werden häufiger aktualisiert als die PDF- und Hardcopybücher. Um stets die neuesten Informationen zur Verfügung zu haben, sollten Sie die Dokumentationsaktualisierungen installieren, sobald diese verfügbar sind, oder das DB2 Information Center unter ibm.com aufrufen.

Darüber hinaus können Sie auf zusätzliche technische Informationen zu DB2, wie beispielsweise technische Hinweise (Technotes), White Papers und IBM Redbooks, online über ibm.com zugreifen. Rufen Sie dazu die Website 'DB2 Information Management - Software - Library' unter http://www.ibm.com/software/data/sw-library/ auf.

Feedback zur Dokumentation

Senden Sie uns Ihr Feedback zur DB2-Dokumentation! Wenn Sie Anregungen zur Verbesserung der DB2-Dokumentation haben, senden Sie eine E-Mail an db2docs@ca.ibm.com. Das DB2-Dokumentationsteam bearbeitet das gesamte Feedback, kann jedoch nicht im Einzelnen auf Ihre E-Mails antworten. Nennen Sie uns, wenn möglich, konkrete Beispiele, sodass wir die Problemstellung besser beurteilen können. Wenn Sie uns Feedback zu einem bestimmten Thema oder einer bestimmten Hilfedatei senden, geben Sie den entsprechenden Titel sowie die URL an.

Verwenden Sie diese E-Mail-Adresse nicht, wenn Sie sich an den DB2-Kundendienst wenden möchten. Wenn ein technisches Problem bei DB2 vorliegt, das Sie mithilfe der Dokumentation nicht beheben können, fordern Sie beim zuständigen IBM Service-Center Unterstützung an.

Bibliothek mit technischen Informationen zu DB2 im Hardcopy- oder PDF-Format

Die folgenden Tabellen enthalten eine Beschreibung der DB2-Bibliothek, die im IBM Publications Center unter www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss zur Verfügung steht. Über die folgende Adresse können Sie englische Handbücher im PDF-Format sowie übersetzte Versionen zu DB2 Version 10.1 herunterladen: www.ibm.com/support/docview.wss?rs=71&uid=swg27009474.

In den Tabellen sind die Bücher, die in gedruckter Form zur Verfügung stehen, gekennzeichnet; möglicherweise sind diese in Ihrem Land oder Ihrer Region jedoch nicht verfügbar.

Die Formnummer wird bei jeder Aktualisierung eines Handbuchs erhöht. Anhand der nachfolgenden Liste können Sie sicherstellen, dass Sie die jeweils neueste Version des Handbuchs lesen.

Anmerkung: Das *DB2 Information Center* wird häufiger aktualisiert als die PDF- und Hardcopybücher.

Tabelle 34. Technische Informationen zu DB2

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Administrative API Reference	SC27-3864-00	Ja	April 2012
Administrative Routines and Views	SC27-3865-01	Nein	Januar 2013
Call Level Interface Guide and Reference Volume 1	SC27-3866-01	Ja	Januar 2013
Call Level Interface Guide and Reference Volume 2	SC27-3867-01	Ja	Januar 2013
Command Reference	SC27-3868-01	Ja	Januar 2013
Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen	SC12-4673-01	Ja	Januar 2013
Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz	SC12-4691-01	Ja	Januar 2013
Datenbanküberwachung - Handbuch und Referenz	SC12-4674-01	Ja	Januar 2013
Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz	SC12-4692-01	Ja	Januar 2013
Datenbanksicherheit	SC12-4693-01	Ja	Januar 2013
DB2 Workload Management - Handbuch und Referenz	SC12-4683-01	Ja	Januar 2013
Developing ADO.NET and OLE DB Applications	SC27-3873-01	Ja	Januar 2013

Tabelle 34. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Developing Embedded SQL Applications	SC27-3874-01	Ja	Januar 2013
Developing Java Applications	SC27-3875-01	Ja	Januar 2013
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-3876-00	Nein	April 2012
Developing RDF Applications for IBM Data Servers	SC27-4462-00	Ja	Januar 2013
Developing User-defined Routines (SQL and External)	SC27-3877-01	Ja	Januar 2013
Getting Started with Database Application Development	GI13-2046-01	Ja	Januar 2013
Installation und Verwaltung von DB2 unter Linux und Windows - Erste Schritte	GI11-3285-00	Ja	April 2012
Globalisierung	SC12-4694-00	Ja	April 2012
DB2-Server - Installation	SC12-4677-01	Ja	Januar 2013
IBM Data Server-Clients - Installation	SC12-4678-00	Nein	April 2012
Fehlernachrichten, Band 1	SC12-4686-01	Nein	Januar 2013
Fehlernachrichten, Band 2	SC12-4687-01	Nein	Januar 2013
Net Search Extender - Verwaltung und Benutzerhandbuch	SC12-4689-01	Nein	Januar 2013
Partitionierung und Clustering	SC12-4695-01	Ja	Januar 2013
Preparation Guide for DB2 10.1 Fundamentals Exam 610	SC27-4540-00	Nein	Januar 2013
Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611	SC27-4541-00	Nein	Januar 2013
pureXML - Handbuch	SC12-4684-01	Ja	Januar 2013
Spatial Extender - Benutzer- und Referenzhandbuch	SC12-4688-00	Nein	April 2012
SQL Procedural Languages: Application Enablement and Support	SC27-3896-01	Ja	Januar 2013
SQL Reference Volume 1	SC27-3885-01	Ja	Januar 2013

Tabelle 34. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
SQL Reference Volume 2	SC27-3886-01	Ja	Januar 2013
Text Search	SC12-4690-01	Ja	Januar 2013
Fehlerbehebung und Optimieren der Datenbankleistung	SC12-4675-01	Ja	Januar 2013
Upgrade auf DB2 Version 10.1	SC12-4676-01	Ja	Januar 2013
Neuerungen in DB2 Version 10.1	SC12-4682-01	Ja	Januar 2013
XQuery - Referenz	SC12-4685-01	Nein	Januar 2013

Tabelle 35. Technische Informationen zu DB2 Connect

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
DB2 Connect - Installation und Konfiguration von DB2 Connect Personal Edition	SC12-4679-00	Ja	April 2012
DB2 Connect - Installation und Konfiguration von DB2 Connect-Servern	SC12-4680-01	Ja	Januar 2013
DB2 Connect - Benutzerhandbuch	SC12-4681-01	Ja	Januar 2013

Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor

DB2-Produkte geben für Bedingungen, die aufgrund einer SQL-Anweisung generiert werden können, einen SQLSTATE-Wert zurück. Die SQLSTATE-Hilfe erläutert die Bedeutung der SQL-Statuswerte und der SQL-Statusklassencodes.

Vorgehensweise

Zum Starten der Hilfe für SQL-Statuswerte müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? SQL-Status oder ? Klassencode
```

Hierbei steht *SQL-Status* für einen gültigen fünfstelligen SQL-Statuswert und *Klassencode* für die ersten beiden Ziffern dieses Statuswerts.
So kann beispielsweise durch die Eingabe von `? 08003` Hilfe für den SQL-Statuswert 08003 angezeigt werden, durch die Eingabe von `? 08` Hilfe für den Klassencode 08.

Zugriff auf verschiedene Versionen des DB2 Information Center

Die Dokumentation für andere Versionen der DB2-Produkte finden Sie in den jeweiligen Information Centers unter ibm.com.

Informationen zu diesem Vorgang

Für Themen aus DB2 Version 10.1 lautet die URL für das *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v10r1.

Für Themen aus DB2 Version 9.8 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/.

Für Themen aus DB2 Version 9.7 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/.

Für Themen aus DB2 Version 9.5 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Für Themen aus DB2 Version 9.1 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Für Themen aus DB2 Version 8 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center

Ein lokal installiertes DB2 Information Center muss regelmäßig aktualisiert werden.

Vorbereitende Schritte

Ein DB2 Version 10.1 Information Center muss bereits installiert sein. Einzelheiten hierzu finden Sie unter „Installation des DB2 Information Center mit dem DB2-Installationsassistenten" in *DB2-Server - Installation*. Alle für die Installation des Information Center geltenden Voraussetzungen und Einschränkungen gelten auch für die Aktualisierung des Information Center.

Informationen zu diesem Vorgang

Ein vorhandenes DB2 Information Center kann automatisch oder manuell aktualisiert werden:

- Mit automatischen Aktualisierungen werden vorhandene Komponenten und Sprachen des Information Center aktualisiert. Ein Vorteil von automatischen Aktualisierungen ist, dass das Information Center im Vergleich zu einer manuellen Aktualisierung nur für einen kurzenZeitraum nicht verfügbar ist. Darüber hinaus können automatische Aktualisierungen so konfiguriert werden, dass sie als Teil anderer, regelmäßig ausgeführter Stapeljobs ausgeführt werden.
- Mit manuellen Aktualisierungen können Sie vorhandene Komponenten und Sprachen des Information Center aktualisieren. Automatische Aktualisierungen reduzieren die Ausfallzeiten während des Aktualisierungsprozesses, Sie müssen jedoch den manuellen Prozess verwenden, wenn Sie Komponenten oder Sprachen hinzufügen möchten. Beispiel: Ein lokales Information Center wurde ursprünglich sowohl mit englischer als auch mit französischer Sprachunterstützung installiert; nun soll auch die deutsche Sprachunterstützung installiert werden. Bei einer manuellen Aktualisierung werden sowohl eine Installation der deutschen Sprachunterstützung als auch eine Aktualisierung der vorhandenen Komponenten und Sprachen des Information Center durchgeführt. Sie müssen jedoch bei einer manuellen Aktualisierung das Information Center manuell stop-

pen, aktualisieren und erneut starten. Das Information Center ist während des gesamten Aktualisierungsprozesses nicht verfügbar. Während des automatischen Aktualisierungsprozesses kommt es zu einem Ausfall des Information Center, und es wird erst wieder nach der Aktualisierung erneut gestartet.

Dieser Abschnitt enthält Details zum Prozess der automatischen Aktualisierung. Anweisungen zur manuellen Aktualisierung finden Sie im Abschnitt „Manuelles Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center".

Vorgehensweise

Gehen Sie wie folgt vor, um das auf Ihrem Computer bzw. Intranet-Server installierte DB2 Information Center automatisch zu aktualisieren:

1. Unter Linux:
   a. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das DB2 Information Center im Verzeichnis `/opt/ibm/db2ic/V10.1` installiert.
   b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc/bin`.
   c. Führen Sie das Script `update-ic` aus:
      ```
      update-ic
      ```
2. Unter Windows:
   a. Öffnen Sie ein Befehlsfenster.
   b. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das DB2 Information Center im Verzeichnis `<Programme>\IBM\DB2 Information Center\Version 10.1` installiert, wobei `<Programme>` das Verzeichnis der Programmdateien angibt.
   c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc\bin`.
   d. Führen Sie die Datei `update-ic.bat` aus:
      ```
      update-ic.bat
      ```

Ergebnisse

Das DB2 Information Center wird automatisch erneut gestartet. Standen Aktualisierungen zur Verfügung, zeigt das Information Center die neuen und aktualisierten Abschnitte an. Waren keine Aktualisierungen für das Information Center verfügbar, wird eine entsprechende Nachricht zum Protokoll hinzugefügt. Die Protokolldatei befindet sich im Verzeichnis `doc\eclipse\configuration`. Der Name der Protokolldatei ist eine Zufallszahl. Beispiel: `1239053440785.log`.

Manuelles Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center

Wenn Sie das DB2 Information Center lokal installiert haben, können Sie Dokumentationsaktualisierungen von IBM abrufen und installieren.

Informationen zu diesem Vorgang

Zur manuellen Aktualisierung des lokal installierten *DB2 Information Center* sind die folgenden Schritte erforderlich:

1. Stoppen Sie das *DB2 Information Center* auf Ihrem Computer und starten Sie das Information Center im Standalone-Modus erneut. Die Ausführung des Information Center im Standalone-Modus verhindert, dass andere Benutzer in Ihrem

Netz auf das Information Center zugreifen, und ermöglicht das Anwenden von Aktualisierungen. Die Workstationversion des DB2 Information Center wird stets im Standalone-Modus ausgeführt.

2. Verwenden Sie die Aktualisierungsfunktion, um zu prüfen, welche Aktualisierungen verfügbar sind. Falls Aktualisierungen verfügbar sind, die Sie installieren müssen, können Sie die Aktualisierungsfunktion verwenden, um diese abzurufen und zu installieren.

   Anmerkung: Wenn es in der verwendeten Umgebung erforderlich ist, die Aktualisierungen für das *DB2 Information Center* auf einer Maschine zu installieren, die nicht über ein Verbindung zum Internet verfügt, spiegeln Sie die Aktualisierungssite auf ein lokales Dateisystem und verwenden Sie dabei eine Maschine, die mit dem Internet verbunden ist und auf der das *DB2 Information Center* installiert ist. Wenn viele Benutzer Ihres Netzes die Dokumentationsaktualisierungen installieren sollen, können Sie die Zeit, die jeder einzelne Benutzer für die Aktualisierungen benötigt, reduzieren, indem Sie die Aktualisierungssite lokal spiegeln und ein Proxy dafür erstellen.
   Ist dies der Fall, verwenden Sie die Aktualisierungsfunktion, um die Pakete abzurufen. Die Aktualisierungsfunktion ist jedoch nur im Standalone-Modus verfügbar.

3. Stoppen Sie das im Standalone-Modus gestartete Information Center und starten Sie das *DB2 Information Center* auf Ihrem Computer erneut.

Anmerkung: Unter Windows 2008 und Windows Vista (und neueren Versionen) müssen die in diesem Abschnitt aufgeführten Befehle mit Administratorberechtigung ausgeführt werden. Zum Öffnen einer Eingabeaufforderung oder eines Grafiktools mit vollen Administratorberechtigungen klicken Sie mit der rechten Maustaste die Verknüpfung an und wählen Sie **Als Administrator ausführen** aus.

Vorgehensweise

Gehen Sie wie folgt vor, um das auf Ihrem Computer bzw. Intranet-Server installierte *DB2 Information Center* zu aktualisieren:

1. Stoppen Sie das *DB2 Information Center*.
   - Unter Windows: Klicken Sie **Start** > **Systemsteuerung** > **Verwaltung** > **Dienste** an. Klicken Sie mit der rechten Maustaste das **DB2 Information Center** an und wählen Sie **Beenden** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:
     ```
     /etc/init.d/db2icdv10 stop
     ```
2. Starten Sie das Information Center im Standalone-Modus.
   - Unter Windows:
     a. Öffnen Sie ein Befehlsfenster.
     b. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das *DB2 Information Center* im Verzeichnis *Programme*\IBM\DB2 Information Center\Version 10.1 installiert, wobei *Programme* das Verzeichnis der Programmdateien angibt.
     c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc\bin.
     d. Führen Sie die Datei help_start.bat aus:
        ```
        help_start.bat
        ```
   - Unter Linux:

a. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das *DB2 Information Center* im Verzeichnis `/opt/ibm/db2ic/V10.1` installiert.
b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc/bin`.
c. Führen Sie das Script `help_start` aus:

```
help_start
```

Der standardmäßig auf dem System verwendete Web-Browser wird geöffnet und zeigt die Standalone-Version des Information Center an.

3. Klicken Sie die Aktualisierungsschaltfläche () an. (JavaScript muss im verwendeten Browser aktiviert sein.) Klicken Sie im rechten Fenster des Information Center die Schaltfläche für die Suche nach Aktualisierungen an. Eine Liste der Aktualisierungen für die vorhandene Dokumentation wird angezeigt.
4. Wählen Sie zum Initiieren des Installationsprozesses die gewünschten Aktualisierungen aus und klicken Sie anschließend die Schaltfläche für die Installation der Aktualisierungen an.
5. Klicken Sie nach Abschluss des Installationsprozesses **Fertigstellen** an.
6. Stoppen Sie das im Standalone-Modus gestartete Information Center:
   - Unter Windows: Navigieren Sie innerhalb des Installationsverzeichnisses zum Verzeichnis `doc\bin`, und führen Sie die Datei `help_end.bat` aus:

     ```
     help_end.bat
     ```

     Anmerkung: Die Stapeldatei `help_end` enthält die Befehle, die erforderlich sind, um die Prozesse, die mit der Stapeldatei `help_start` gestartet wurden, ordnungsgemäß zu stoppen. Verwenden Sie nicht die Tastenkombination `Strg+C` oder eine andere Methode, um `help_start.bat` zu stoppen.
   - Unter Linux: Navigieren Sie innerhalb des Installationsverzeichnisses zum Verzeichnis `doc/bin`, und führen Sie das Script `help_end` aus:

     ```
     help_end
     ```

     Anmerkung: Das Script `help_end` enthält die Befehle, die erforderlich sind, um die Prozesse, die mit dem Script `help_start` gestartet wurden, ordnungsgemäß zu stoppen. Verwenden Sie keine andere Methode, um das Script `help_start` zu stoppen.
7. Starten Sie das *DB2 Information Center* erneut.
   - Unter Windows: Klicken Sie **Start** > **Systemsteuerung** > **Verwaltung** > **Dienste** an. Klicken Sie mit der rechten Maustaste das **DB2 Information Center** an und wählen Sie **Start** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:

     ```
     /etc/init.d/db2icdv10 start
     ```

Ergebnisse

Im aktualisierten *DB2 Information Center* werden die neuen und aktualisierten Themen angezeigt.

DB2-Lernprogramme

Die DB2-Lernprogramme unterstützen Sie dabei, sich mit den unterschiedlichen Aspekten der DB2-Produkte vertraut zu machen. Die Lerneinheiten bieten eine in einzelne Schritte unterteilte Anleitung.

Vorbereitungen

Die XHTML-Version des Lernprogramms kann über das Information Center unter http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/ angezeigt werden.

In einigen der Lerneinheiten werden Beispieldaten und Codebeispiele verwendet. Informationen zu bestimmten Voraussetzungen für die Ausführung der Tasks finden Sie in der Beschreibung des Lernprogramms.

DB2-Lernprogramme

Klicken Sie zum Anzeigen des Lernprogramms den Titel an.

„pureXML" in *pureXML - Handbuch*
Einrichten einer DB2-Datenbank, um XML-Daten zu speichern und Basisoperationen mit dem nativen XML-Datenspeicher auszuführen.

Informationen zur Fehlerbehebung in DB2

Es steht eine breite Palette verschiedener Informationen zur Fehlerbestimmung und Fehlerbehebung zur Verfügung, um Sie bei der Verwendung von DB2-Datenbankprodukten zu unterstützen.

DB2-Dokumentation
Informationen zur Fehlerbehebung stehen im Handbuch *Fehlerbehebung und Optimieren der Datenbankleistung* oder im Abschnitt mit grundlegenden Informationen zu Datenbanken im *DB2 Information Center* zur Verfügung, darunter:

- Informationen zum Eingrenzen und Aufdecken von Problemen mithilfe der Diagnosetools und -dienstprogramme von DB2.
- Lösungsvorschläge zu den am häufigsten auftretenden Problemen.
- Ratschläge zum Lösen anderer Probleme, die bei Verwendung der DB2-Datenbankprodukte auftreten können.

IBM Support Portal
Im IBM Support Portal finden Sie Informationen zu Problemen und den möglichen Ursachen und Fehlerbehebungsmaßnahmen. Die Website mit technischer Unterstützung enthält Links zu den neuesten DB2-Veröffentlichungen, technischen Hinweisen (TechNotes), APARs (Authorized Program Analysis Reports) und Fehlerkorrekturen, Fixpacks sowie weiteren Ressourcen. Sie können diese Wissensbasis nach möglichen Lösungen für aufgetretene Probleme durchsuchen.

Sie können auf das IBM Support Portal über die folgende Website zugreifen: http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows.

Bedingungen

Die Berechtigungen zur Nutzung dieser Veröffentlichungen werden Ihnen auf der Basis der folgenden Bedingungen gewährt.

Anwendbarkeit: Diese Bedingungen gelten zusätzlich zu den Nutzungsbedingungen für die IBM Website.

Persönliche Nutzung: Sie dürfen diese Veröffentlichungen für Ihre persönliche, nicht kommerzielle Nutzung unter der Voraussetzung vervielfältigen, dass alle Eigentumsvermerke erhalten bleiben. Sie dürfen diese Veröffentlichungen oder Teile dieser Veröffentlichungen ohne ausdrückliche Genehmigung von IBM nicht weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Kommerzielle Nutzung: Sie dürfen diese Veröffentlichungen nur innerhalb Ihres Unternehmens und unter der Voraussetzung, dass alle Eigentumsvermerke erhalten bleiben, vervielfältigen, weitergeben und anzeigen. Sie dürfen diese Veröffentlichungen oder Teile dieser Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens nicht vervielfältigen, weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Rechte: Abgesehen von den hier gewährten Berechtigungen erhalten Sie keine weiteren Berechtigungen, Lizenzen oder Rechte (veröffentlicht oder stillschweigend) in Bezug auf die Veröffentlichungen oder darin enthaltene Informationen, Daten, Software oder geistiges Eigentum.

IBM behält sich das Recht vor, die in diesem Dokument gewährten Berechtigungen nach eigenem Ermessen zurückzuziehen, wenn sich die Nutzung der Veröffentlichungen für IBM als nachteilig erweist oder wenn die obigen Nutzungsbestimmungen nicht genau befolgt werden.

Sie dürfen diese Informationen nur in Übereinstimmung mit allen anwendbaren Gesetzen und Vorschriften, einschließlich aller US-amerikanischen Exportgesetze und Verordnungen, herunterladen und exportieren.

IBM übernimmt keine Gewährleistung für den Inhalt dieser Informationen. Diese Veröffentlichungen werden auf der Grundlage des gegenwärtigen Zustands (auf "as-is"-Basis) und ohne eine ausdrückliche oder stillschweigende Gewährleistung für die Handelsüblichkeit, die Verwendungsfähigkeit oder die Freiheit der Rechte Dritter zur Verfügung gestellt.

IBM Marken: IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation. Weitere Produkt- oder Servicenamen können Marken von IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite www.ibm.com/legal/copytrade.shtml.

Anhang B. Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Die Informationen über Produkte anderer Hersteller als IBM basieren auf den zum Zeitpunkt der ersten Veröffentlichung dieses Dokuments verfügbaren Informationen und können geändert werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Dokument aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichen Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung kann Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes enthalten. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Beispielprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Die Beispielprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM haftet nicht für Schäden, die durch Verwendung der Beispielprogramme entstehen.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (*Name Ihrer Firma*) (*Jahr*). Teile des vorliegenden Codes wurden aus Beispielprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. *_Jahr/Jahre angeben_*. Alle Rechte vorbehalten.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- oder Servicenamen können Marken von oder anderen Herstellern sein. IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite „Copyright and trademark information" unter www.ibm.com/legal/copytrade.shtml.

Die folgenden Namen sind Marken oder eingetragene Marken anderer Unternehmen.

- Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.
- Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken von Oracle und/oder ihren verbundenen Unternehmen.
- UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.
- Intel, das Intel-Logo, Intel Inside, Intel Inside logo, Celeron, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder deren Tochtergesellschaften in den USA und anderen Ländern.
- Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

SC12-4692-01

Spine information:

IBM DB2 10.1 for Linux, UNIX and Windows

Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz