IBM DB2 Universal Database

Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz

Version 8.2

SC12-3054-01

IBM DB2 Universal Database

Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz

Version 8.2

SC12-3054-01

Anmerkung

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter *Bemerkungen* gelesen werden.

- Die IBM Homepage finden Sie im Internet unter: **ibm.com**
- IBM und das IBM Logo sind eingetragene Marken der International Business Machines Corporation.
- Das e-business-Symbol ist eine Marke der International Business Machines Corporation.
- Infoprint ist eine eingetragene Marke der IBM.
- ActionMedia, LANDesk, MMX, Pentium und ProShare sind Marken der Intel Corporation in den USA und/oder anderen Ländern.
- C-bus ist eine Marke der Corollary, Inc. in den USA und/oder anderen Ländern.
- Java und alle auf Java basierenden Marken und Logos sind Marken der Sun Microsystems, Inc. in den USA und/oder anderen Ländern.
- Microsoft Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.
- PC Direct ist eine Marke der Ziff Communications Company in den USA und/oder anderen Ländern.
- SET und das SET-Logo sind Marken der SET Secure Electronic Transaction LLC.
- UNIX ist eine eingetragene Marke der Open Group in den USA und/oder anderen Ländern.
- Marken anderer Unternehmen/Hersteller werden anerkannt.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference,
IBM Form SC09-4831-01

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:
SW TSC Germany
Kst. 2877
September 2004

Inhaltsverzeichnis

Zu diesem Handbuch

Dieses Buch bietet detaillierte Informationen zu den IBM DB2 Universal Database-Dienstprogrammen zum Sichern, Zurückschreiben und Wiederherstellen und ihrer Verwendung. In diesem Buch wird außerdem die Bedeutung einer hohen Verfügbarkeit erläutert, und es wird die DB2-Funktionsübernahmeunterstützung (Failover Support) auf einigen Plattformen beschrieben.

Zielgruppe

Dieses Handbuch richtet sich an Datenbankadministratoren, Anwendungsprogrammierer und andere DB2 UDB-Benutzer, die für die Sicherungs-, Zurückschreibe- und Wiederherstellungsoperationen auf DB2-Datenbanksystemen verantwortlich sind oder sie verstehen möchten.

Es wird vorausgesetzt, dass Sie mit DB2 Universal Database (DB2 UDB), Structured Query Language (SQL) und der Betriebssystemumgebung vertraut sind, in der DB2 UDB ausgeführt wird. Das vorliegende Handbuch enthält keine Anleitungen zur Installation von DB2, da diese vom Betriebssystem abhängen.

Aufbau des Handbuchs

Die folgenden Themen werden behandelt:

Datenwiederherstellung

Kapitel 1, „Entwickeln einer guten Sicherungs- und Wiederherstellungsstrategie"
Behandelt die Punkte, die bei der Auswahl der Methoden zur Wiederherstellung von Datenbanken und Tabellenbereichen zu beachten sind, und enthält Informationen zum Sichern und Zurückschreiben einer Datenbank oder eines Tabellenbereichs sowie zur aktualisierenden Wiederherstellung.

Kapitel 2, „Datenbanksicherung"
Beschreibt das DB2-Sicherungsdienstprogramm (BACKUP), mit dem Sicherungskopien einer Datenbank oder eines Tabellenbereichs erstellt werden.

Kapitel 3, „Datenbankwiederherstellung"
Beschreibt das DB2-Wiederherstellungsdienstprogramm (RECOVER), mit dem beschädigte Datenbanken oder Tabellenbereiche, die vorher gesichert wurden, erneut erstellt werden.

Kapitel 4, „Aktualisierende Wiederherstellung"
Beschreibt das Dienstprogramm zur aktualisierenden Wiederherstellung (ROLLFORWARD), mit dem eine Datenbank wiederhergestellt wird, indem die in den Datenbankwiederherstellungsprotokolldateien aufgezeichneten Transaktionen angewendet werden.

Kapitel 5, „Datenbankwiederherstellung"
Beschreibt das DB2-Wiederherstellungsdienstprogramm, das auf Basis der Informationen in der Datei des Wiederherstellungsprotokolls die erforderlichen Operationen zum Wiederherstellen und aktualisierenden Wiederherstellen einer Datenbank bis zu einem bestimmten Zeitpunkt ausführt.

Hohe Verfügbarkeit

Kapitel 6, „Einführung in die Unterstützung der hohen Verfügbarkeit und der Funktionsübernahme"
Bietet eine Übersicht über die von DB2 bereitgestellte Unterstützung von Funktionsübernahmen zur Implementierung hoher Verfügbarkeit.

Kapitel 7, „HADR (High Availability Disaster Recovery)"
Erläutert die Konzepte und Prozeduren, die zum Konfigurieren und Verwalten einer HADR-Umgebung (HADR - High Availability Disaster Recovery) erforderlich sind.

Unterstützung für High Availability Cluster Multi-Processing (HACMP)
Behandelt die DB2-Unterstützung für die Fehlerbehebung durch Funktionsübernahme mit hoher Verfügbarkeit unter AIX, die zurzeit durch das Feature ″Enhanced Scalability″ (ES - Erweiterte Skalierbarkeit) von High Availability Cluster Multi-Processing (HACMP) für AIX implementiert ist.

Unterstützung für Microsoft Cluster Server
Behandelt die DB2-Unterstützung für die Fehlerbehebung durch Funktionsübernahme mit hoher Verfügbarkeit unter Windows-Betriebssystemen, die zurzeit durch Microsoft Cluster Server (MSCS) implementiert ist.

Kapitel 10, „Clusterunterstützung für die Solaris-Betriebsumgebung"
Behandelt die DB2-Unterstützung für die Fehlerbehebung durch Funktionsübernahme mit hoher Verfügbarkeit in der Solaris-Betriebsumgebung, die zurzeit durch Sun Cluster 3.0 (SC3.0) oder Veritas Cluster Server (VCS) implementiert ist.

Anhänge

Anhang A, „Lesen von Syntaxdiagrammen"
Erläutert die in Syntaxdiagrammen verwendeten Konventionen.

Anhang B, „Warnungen, Fehler- und Beendigungsnachrichten"
Bietet Informationen zur Auswertung von Nachrichten, die vom Datenbankmanager generiert werden, wenn eine Warnungs- oder Fehlerbedingung erkannt wird.

Anhang C, „Weitere DB2-Befehle"
Beschreibt wiederherstellungsbezogene DB2-Befehle.

Anhang D, „Weitere APIs und zugehörige Datenstrukturen"
Beschreibt wiederherstellungsbezogene APIs und ihre Datenstrukturen.

Anhang E, „Beispielwiederherstellungsprogramme"
Bietet eine Codeliste für ein Beispielprogramm, das wiederherstellungsbezogene DB2-APIs und eingebettete SQL-Aufrufe enthält, und Informationen zu ihrer Verwendung.

Anhang F, „Tivoli Storage Manager"
Enthält Informationen zum Produkt Tivoli Storage Manager (TSM), das Sie zur Verwaltung von Datenbank- oder Tabellenbereichssicherungsoperationen verwenden können.

Anhang G, „Benutzerexit zur Datenbankwiederherstellung"
Behandelt die Verwendung von Benutzerexitprogrammen für Datenbankprotokolldateien und beschreibt einige Beispiele für Benutzerexitprogramme.

Anhang H, „APIs zum Sichern und Wiederherstellen für Produkte anderer Lieferanten"
Beschreibt die Funktion und Verwendung von APIs, über die DB2 eine Schnittstelle zu Software von anderen Lieferanten erhält.

Teil 1. Datenwiederherstellung

Kapitel 1. Entwickeln einer guten Sicherungs- und Wiederherstellungsstrategie

In diesem Abschnitt werden die Punkte behandelt, die bei der Auswahl der Methoden zur Wiederherstellung von Datenbanken und Tabellenbereichen zu beachten sind, und er enthält Informationen zum Sichern und Zurückschreiben einer Datenbank oder eines Tabellenbereichs sowie zur aktualisierenden Wiederherstellung.

Die folgenden Themen werden behandelt:

Entwickeln einer Sicherungs- und Wiederherstellungsstrategie

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 Wiederherstellungsaktionen. Schützen Sie Ihre Daten vor Verlust, indem Sie eine gut erprobte Wiederherstellungsstrategie bereithalten. Bei der Entwicklung Ihrer Wiederherstellungsstrategie sollten Sie unter anderem folgende Fragen berücksichtigen:

- Ist die Datenbank wiederherstellbar?
- Wie viel Zeit steht für die Wiederherstellung der Datenbank zur Verfügung?
- In welchen Abständen werden Sicherungsoperationen durchgeführt?
- Wie viel Speicher kann für Sicherungskopien und Archivprotokolldateien zugeordnet werden?
- Sind Sicherungen auf Tabellenbereichsebene ausreichend, oder muss die gesamte Datenbank gesichert werden?
- Soll ein Bereitschaftssystem manuell oder durch HADR (High Availability Disaster Recovery) konfiguriert werden?

Eine Strategie zur Wiederherstellung der Datenbank sollte sicherstellen, dass alle Informationen verfügbar sind, wenn sie für eine Wiederherstellung der Datenbank benötigt werden. Sie sollte einen Zeitplan für regelmäßige Sicherungen enthalten. Im Fall von partitionierten Datenbanksystemen sollten auch nach einer Skalierung

des Systems, d. h. nach dem Hinzufügen eines Datenbankpartitionsservers oder -knotens, Sicherungen durchgeführt werden. Ihre Gesamtstrategie sollte auch Prozeduren zum Wiederherstellen von Befehlsprozeduren, Anwendungen, benutzerdefinierten Funktionen, Code gespeicherter Prozeduren in Betriebssystembibliotheken sowie von Ladekopien enthalten.

In den folgenden Abschnitten werden verschiedene Wiederherstellungsmethoden behandelt. Sie können bestimmen, welche Wiederherstellungsmethode für Ihre Geschäftsumgebung am besten geeignet ist.

Das Konzept einer *Datenbanksicherung* ist mit jeder anderen Datensicherung 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 einer Sicherung bildet das Verfahren, bei dem zunächst die Datenbank gestoppt wird, um sicherzustellen, dass keine weiteren Transaktionen auftreten, und anschließend ein Sicherungsimage der Daten erstellt wird. Sie können die Datenbank dann erneut erstellen, falls sie beschädigt wird.

Das erneute Erstellen der Datenbank wird als *Wiederherstellung* bezeichnet. Eine *Versionswiederherstellung* ist die Wiederherstellung einer früheren Version der Datenbank mit Hilfe eines Images der Datenbank, das im Rahmen einer Sicherungsoperation erstellt wurde. Eine *aktualisierende Wiederherstellung* ist die erneute Anwendung von in den Datenbankprotokolldateien aufgezeichneten Transaktionen nach der Wiederherstellung eines Sicherungsimages einer Datenbank oder eines Tabellenbereichs.

Eine *Wiederherstellung nach einem Systemabsturz* ist die automatische Wiederherstellung der Datenbank, wenn ein Fehler auftritt, bevor alle Änderungen einer oder mehrerer Arbeitseinheiten (Transaktionen) beendet und festgeschrieben wurden. Dies geschieht durch Rückgängigmachen der unvollständigen Transaktionen und Beenden der festgeschriebenen Aktionen, die sich noch im Hauptspeicher befanden, als der Systemabsturz auftrat.

Die Protokolldateien für die Wiederherstellung und die Datei des Wiederherstellungsprotokolls werden automatisch erstellt, wenn eine Datenbank erstellt wird (Abb. 1 auf Seite 5). Diese Protokolldateien sind wichtig, falls Sie verlorene oder beschädigte Daten wiederherstellen müssen.

Jede Datenbank enthält *Wiederherstellungsprotokolle*, die zur Datenwiederherstellung nach Anwendungs- oder Systemfehlern verwendet werden. In Kombination mit den Datenbanksicherungen dienen sie zur Wiederherstellung der Konsistenz der Datenbank bis exakt zu dem Zeitpunkt, an dem der Fehler auftrat.

Die *Datei des Wiederherstellungsprotokolls* enthält eine Zusammenfassung der Siche rungsinformationen, die zur Bestimmung der Wiederherstellungsoptionen verwendet werden können, wenn die Datenbank insgesamt oder teilweise bis zu einem bestimmten Zeitpunkt wiederhergestellt werden muss. Sie dient zur Protokollierung wiederherstellungsrelevanter Ereignisse wie z. B. Sicherungs- und Wiederherstellungsoperationen. Diese Datei befindet sich im Datenbankverzeichnis.

Die *Protokolldatei der Tabellenbereichsänderungen*, die sich ebenfalls im Datenbankverzeichnis befindet, enthält Informationen, anhand derer bestimmt werden kann, welche Protokolldateien zur Wiederherstellung eines bestimmten Tabellenbereichs erforderlich sind.

Die Datei des Wiederherstellungsprotokolls oder die Protokolldatei der Tabellenbereichsänderungen können nicht direkt geändert werden, Sie können jedoch mit dem Befehl PRUNE HISTORY Einträge aus den Dateien löschen. Sie haben auch die Möglichkeit, mit dem Datenbankkonfigurationsparameter *rec_his_retentn* die Anzahl Tage anzugeben, die diese Protokolldateien beibehalten werden sollen.

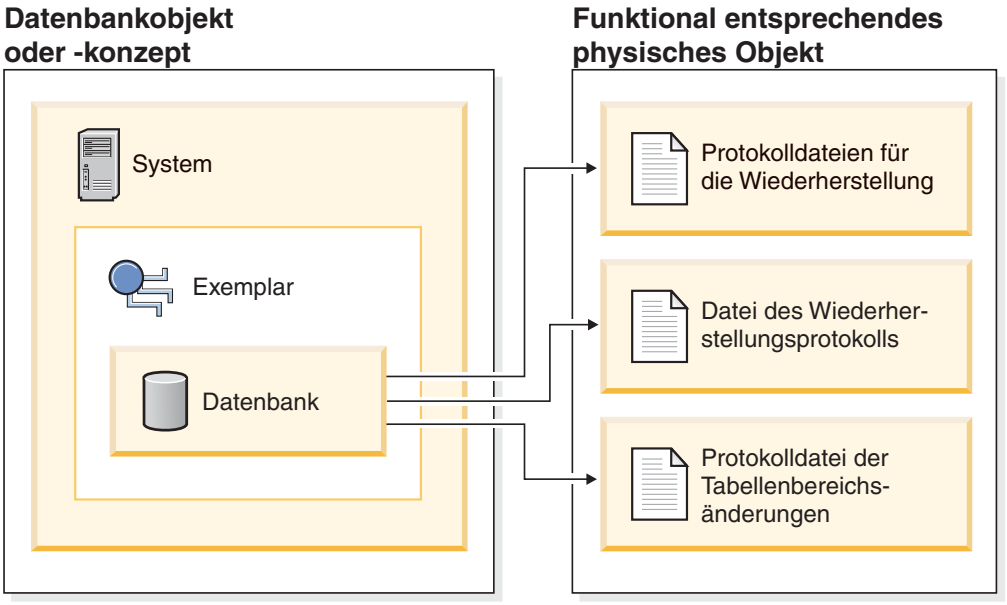


Abbildung 1. Datenbankwiederherstellungsdateien

Daten, die einfach erneut erstellt werden können, 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 Protokollierungs- b.r aufwand nicht die zusätzliche Komplexität der Verwaltung von Protokolldateien sowie die aktualisierende Wiederherstellung nach einer Wiederherstellungsoperation rechtfertigt. Bei *nicht wiederherstellbaren Datenbanken* sind die Datenbankkonfigurationsparameter *logarchmeth1* und *logarchmeth2* auf „OFF" gesetzt. Dies bedeutet, dass lediglich die für die Wiederherstellung nach einem Systemabsturz erforderlichen Protokolle beibehalten werden. Diese Protokolle werden als *aktive Protokolldateien* bezeichnet und enthalten aktuelle Transaktionsdaten. Die Versionswiederherstellung mit Hilfe von *Offlinesicherungen* ist das primäre Mittel zur Wiederherstellung einer nicht wiederherstellbaren Datenbank. (Eine Offlinesicherung bedeutet, dass während der Sicherungsoperation 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 Sicherungsimage erstellt wurde. Eine aktualisierende Wiederherstellung wird nicht unterstützt.

Daten, die *nicht* einfach erneut erstellt werden können, 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* ist einer der Datenbankkonfigurationsparameter *logarchmeth1* oder *logarchmeth2* auf einen anderen Wert als „OFF" gesetzt.

Aktive Protokolldateien sind weiterhin für die Datenbankwiederherstellung nach einem Systemabsturz verfügbar, Ihnen 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 Sicherungsimage erstellt wurde. Mit Hilfe der aktualisierenden Wiederherstellung (Rollforward Recovery) können Sie jedoch die Datenbank *aktualisierend wiederherstellen* (also über den Zeitpunkt hinaus, an dem das Sicherungsimage erstellt wurde), indem Sie die aktiven und archivierten Protokolle entweder bis zu einem bestimmten Zeitpunkt oder bis zum Ende der aktiven Protokolldateien anwenden.

Sicherungsoperationen für wiederherstellbare Datenbanken können entweder offline oder *online* durchgeführt werden (online heißt, dass andere Anwendungen während der Sicherungsoperation eine Verbindung zur Datenbank herstellen können). Operationen für die Onlinewiederherstellung eines Tabellenbereichs und für die aktualisierende Wiederherstellung werden nur für wiederherstellbare Datenbanken unterstützt. Wenn die Datenbank nicht wiederherstellbar ist, müssen Operationen zur Wiederherstellung und zur aktualisierenden Wiederherstellung der Datenbank offline ausgeführt werden. Während einer Onlinesicherung stellt die aktualisierende Wiederherstellung sicher, dass *alle* Tabellenänderungen erfasst und erneut angewendet werden, wenn diese Sicherungskopie wiederhergestellt wird.

Bei einer wiederherstellbaren Datenbank können Sie 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 Sicherungsoperationen

Mit dem Assistenten **Automatische Verwaltung konfigurieren** können Sie das zeitaufwändige Ermitteln und Planen von erforderlichen Verwaltungsaktivitäten, wie z. B. Sicherungsoperationen, vereinfachen. Bei automatischer Verwaltung geben Sie Ihre Zielsetzung für die Verwaltung sowie die Zeitpunkte an, zu denen die automatische Verwaltung ausgeführt werden kann. DB2 ermittelt 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: Manuelle Sicherungsoperationen können auch bei konfigurierter automatischer Verwaltung ausgeführt werden. DB2 führt automatische Sicherungsoperationen nur bei Bedarf aus.

Zugehörige Konzepte:

- „Wiederherstellung nach einem Systemabsturz" auf Seite 12
- „Versionswiederherstellung" auf Seite 26
- „Aktualisierende Wiederherstellung" auf Seite 27
- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201
- „Data Links server file backups" im Handbuch *DB2 Data Links Manager Administration Guide and Reference*
- „Failure and recovery overview" im Handbuch *DB2 Data Links Manager Administration Guide and Reference*

Zugehörige Referenzen:

- „rec_his_retentn - Aufbewahrungszeitraum für Wiederherstellungsprotokoll" im Handbuch *Systemverwaltung: Optimierung*
- „logarchmeth1 - Primäre Protokollarchivierungsmethode (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „DB2 Data Links Manager system setup and backup recommendations" im Handbuch *DB2 Data Links Manager Administration Guide and Reference*

Häufigkeit der Sicherung

In Ihrem Wiederherstellungsplan sollten regelmäßige Sicherungsoperationen vorgesehen sein, da das Sichern einer Datenbank Zeit und Systemressourcen in Anspruch nimmt. Ihr Plan kann eine Kombination von vollständigen Datenbanksicherungen und Teilsicherungsoperationen enthalten.

Sie sollten in regelmäßigen Abständen Gesamtsicherungen der Datenbanken erstellen, selbst wenn Sie die Protokolle archivieren (wodurch eine aktualisierende Wiederherstellung ermöglicht wird). Für die Wiederherstellung einer Datenbank benötigen Sie ein Image einer Datenbankgesamtsicherung, das alle Tabellenbereichs- b.r sicherungsimages enthält. Tabellenbereichssicherungsimages sind sinnvoll für eine Wiederherstellung nach dem Ausfall einer einzelnen Platte oder nach einem Anwendungsfehler. In Umgebungen mit partitionierten Datenbanken müssen Sie jeweils nur die fehlerhaften Tabellenbereiche wiederherstellen. Es müssen nicht jedes Mal alle Tabellenbereiche oder Partitionen wiederhergestellt werden.

Außerdem sollten Sie in Betracht ziehen, die Sicherungsimages und Protokolldateien nicht zu überschreiben, sondern mindestens zwei Gesamtsicherungsimages der Datenbank mit zugehörigen Protokollen als weitere Vorsichtsmaßnahme zu sichern.

Wenn die erforderliche Zeit für die Anwendung der archivierten Protokolldateien bei der Wiederherstellung und der aktualisierenden Wiederherstellung einer sehr aktiven Datenbank wichtig ist, sollten Sie den Aufwand häufigerer Datenbanksicherungen in Erwägung ziehen. Dadurch verringert sich die Anzahl der archivierten Protokolldateien, die Sie bei einer aktualisierenden Wiederherstellung anwenden müssen.

Sie können eine Sicherungsoperation 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 Sicherungsoperation ausgeführt wird. Wird die Sicherungsoperation 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 Onlinesicherungsoperationen in Betracht ziehen. Onlinesicherungsoperationen werden nur unterstützt, wenn die aktualisierende Wiederherstellung aktiviert ist. Wenn die aktualisierende Wiederherstellung aktiviert ist und Sie über einen kompletten Satz von Wiederherstellungsprotokollen verfügen, können Sie die Datenbank im Bedarfsfall erneut erstellen. Sie können ein Onlinesicherungsimage nur dann für die Wiederherstellung verwenden, wenn Sie über die Protokolle verfügen, die den Zeitraum für die Sicherungsoperation umfassen.

Offlinesicherungsoperationen sind schneller als Onlinesicherungsoperationen, da zwischen den Datendateien keine Konkurrenzsituation auftritt.

Mit dem Sicherungsdienstprogramm können Sie ausgewählte Tabellenbereiche sichern. Wenn Sie DMS-Tabellenbereiche verwenden, können Sie verschiedene Datentypen in speziellen Tabellenbereichen speichern, um die für Sicherungsoperationen 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 diese Maßnahmen ergreifen, wird ein auftretender Plattenfehler vermutlich nur einen der Tabellenbereiche betreffen. Zur Wiederherstellung oder aktualisierenden Wiederherstellung dieser Tabellenbereiche ist weniger Zeit erforderlich, als für die Wiederherstellung eines einzelnen Tabellenbereichs, der alle Daten enthält.

Sie können auch Zeit sparen, indem Sie zu unterschiedlichen Zeitpunkten Sicherungen 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 Wiederherstellung benötigt werden, ist eine Sicherung 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.

Anmerkung: Der folgende Hinweis gilt für den Fall, dass Sie Ihre Langfelddaten, LOB-Daten und Indizes in separaten Tabellenbereichen speichern, aber sie nicht gemeinsam sichern: Wenn Sie einen Tabellenbereich sichern, der nicht alle Tabellendaten enthält, können Sie keine aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt für diesen Tabellenbereich ausführen. Alle Tabellenbereiche, die irgendeine Art von Daten für eine Tabelle enthalten, müssen gleichzeitig bis zum selben Zeitpunkt aktualisierend wiederhergestellt werden.

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.

Die Zeit, die zur Wiederherstellung einer Datenbank benötigt wird, setzt sich aus zwei Perioden zusammen: der Zeit, die zur Wiederherstellung der Sicherungskopie benötigt wird, sowie, wenn die Datenbank für die aktualisierende Wiederherstellung aktiviert ist, der Zeit, die zur Anwendung der Protokolle während der aktualisierenden Wiederherstellung benötigt wird. Bei der Formulierung eines Wiederherstellungsplans sollten Sie diesen Wiederherstellungsaufwand und seine Auswirkung auf den Geschäftsbetrieb berücksichtigen. Durch Testen des Gesamtwiederherstellungsplans können Sie ermitteln, ob die Zeit, die zur Wiederherstellung der Datenbank benötigt wird, in Anbetracht Ihrer Geschäftserfordernisse angemessen ist. Nach jedem Test stellen Sie möglicherweise fest, dass es vorteilhaft ist, die Häufigkeit der Sicherungen zu erhöhen. Wenn Ihre Strategie eine aktualisierende Wiederherstellung vorsieht, wird dieses Vorgehen die Anzahl der zwischen Sicherungen archivierten Protokolle reduzieren und auf diese Weise die für eine aktualisierende Wiederherstellung der Datenbank benötigte Zeit nach einer Wiederherstellung von der Sicherungskopie verringern.

Zugehörige Konzepte:

- „Entwickeln einer Sicherungs- und Wiederherstellungsstrategie" auf Seite 3
- „Teilsicherung und Teilwiederherstellung" auf Seite 30

Zugehörige Referenzen:

- Anhang G, „Benutzerexit zur Datenbankwiederherstellung", auf Seite 369
- „Konfigurationsparameter für die Datenbankprotokollierung" auf Seite 42

Aspekte des Speicherbedarfs für die Wiederherstellung

Bei der Entscheidung für die zu verwendende Wiederherstellungsmethode sollten Sie auch den erforderlichen Speicherplatz in Betracht ziehen.

Bei der Versionswiederherstellung wird Speicherplatz für die Sicherungskopie der Datenbank und für die wiederhergestellte Datenbank benötigt. Bei einer aktualisierenden Wiederherstellung wird Speicherplatz für die Sicherungskopie der Datenbank bzw. der Tabellenbereiche, für die wiederhergestellte Datenbank sowie für die archivierten Datenbankprotokolle benötigt.

Enthält eine Tabelle Langfeld- oder LOB-Spalten, ist es u. U. ratsam, die betreffenden Daten in einen separaten Tabellenbereich zu stellen. Dies hat Auswirkungen auf Ihre Überlegungen zum Speicherbedarf sowie auf Ihren Wiederherstellungsplan. 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 Wiederherstellungsplan 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. Dadurch verringert sich die Größe der Protokolldateien und somit der erforderliche Protokollarchivierungsspeicher.

Um zu verhindern, dass ein Datenträgerfehler die Datenbank beschädigt und Ihnen die Wiederherstellung unmöglich macht, sollten Sie die Sicherungskopie 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 mit Hilfe des Konfigurationsparameters *newlogpath* auf eine separate Einheit umzuleiten.

Die Datenbankprotokolle können viel Speicherplatz in Anspruch nehmen. Wenn Sie beabsichtigen, eine aktualisierende Wiederherstellung auszuführen, müssen Sie entscheiden, wie die archivierten Protokolle verwaltet werden. Ihnen stehen folgende Möglichkeiten zur Auswahl:

- Geben Sie unter Verwendung des Konfigurationsparameters LOGARCHMETH1 oder LOGARCHMETH2 eine Protokollarchivierungsmethode an.
- 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 mit Hilfe eines Benutzerexitprogramms auf eine andere Speichereinheit in Ihrer Umgebung.

Zugehörige Konzepte:

- „Verwalten von Protokolldateien durch Protokollarchivierung" auf Seite 54

Zugehörige Referenzen:

- „Konfigurationsparameter für die Datenbankprotokollierung" auf Seite 42
- „logarchmeth1 - Primäre Protokollarchivierungsmethode (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „logarchmeth2 - Sekundäre Protokollarchivierungsmethode (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*

Zusammenhalten zusammengehöriger Daten

Aufgrund Ihres Datenbankentwurfs sind Sie mit den Abhängigkeiten vertraut, die zwischen Tabellen bestehen. Diese Abhängigkeiten können auf Anwendungsebene ausgedrückt werden, wenn Transaktionen mehr als eine Tabelle aktualisieren, oder auf Datenbankebene, wenn es referenzielle Integritätsbedingungen zwischen Tabellen gibt, oder wenn sich Auslöser für eine Tabelle auf eine andere Tabelle auswirken. Diese Abhängigkeiten sollten bei der Entwicklung eines Wiederherstellungsplans berücksichtigt werden. Wahrscheinlich ist es sinnvoll, voneinander abhängige Dateien gemeinsam zu sichern. Solche zusammengehörigen Dateien können entweder auf Tabellenbereichsebene oder auf Datenbankebene eingerichtet werden. Wenn die zusammengehörigen Dateien 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 Wiederherstellung bis zu einem bestimmten Zeitpunkt auszuführen.

Verwenden verschiedener Betriebssysteme

DB2® unterstützt plattformübergreifende Sicherungs- und Wiederherstellungsoperationen. Sie können Datenbanken, die mit DB2 Version 8 auf 32-Bit-Windows®-Plattformen erstellt wurden, in DB2 Version 8 auf 64-Bit-Windows-Plattformen wiederherstellen und umgekehrt. Sie können Datenbanken, die mit DB2 Version 8 auf 32-Bit-Linux-Plattformen (Intel) erstellt wurden, in DB2 Version 8 auf 64-Bit-Linux-Plattformen (Intel) wiederherstellen und umgekehrt. Sie können Datenbanken, die mit DB2 Version 8 unter AIX®, HP-UX oder in der Solaris-Betriebsumgebung als 32-Bit- oder 64-Bit-Datenbanken erstellt wurden, in DB2 Version 8 unter AIX, HP-UX oder in der Solaris-Betriebsumgebung (32 Bit oder 64 Bit) wiederherstellen.

Sie können auch Sicherungsimages wiederherstellen, die mit einer früheren Version von DB2 (maximal zwei Versionen früher) erstellt wurden, sofern die Wortgröße (32 Bit oder 64 Bit) identisch ist. Plattformübergreifende Wiederherstellungsoperationen aus Sicherungsimages, die mit einer früheren DB2-Version erstellt wurden, werden nicht unterstützt. Auf dem Zielsystem muss dieselbe (oder eine höhere) Version von DB2 installiert sein wie auf dem Quellensystem. Wiederherstellungsoperationen auf ein untergeordnetes System werden nicht unterstützt.

Wenn Sie Tabellen von einem Betriebssystem auf ein anderes versetzen müssen und in Ihrer Umgebung das plattformübergreifende Sichern und Wiederherstellen nicht unterstützt wird, verwenden Sie den Befehl **db2move**, oder verwenden Sie das Exportdienstprogramm und anschließend das Importdienstprogramm und das Dienstprogramm LOAD.

Zugehörige Referenzen:

- „db2move - Database Movement Tool Command" im Handbuch *Command Reference*
- „EXPORT Command" im Handbuch *Command Reference*
- „IMPORT Command" im Handbuch *Command Reference*
- „LOAD Command" im Handbuch *Command Reference*

Wiederherstellung nach einem Systemabsturz

Transaktionen (bzw. Arbeitseinheiten), die für eine Datenbank ausgeführt werden, können auf unerwartete Weise unterbrochen werden. Wenn eine Störung auftritt, bevor alle Änderungen, die Bestandteil der Arbeitseinheit sind, beendet und festgeschrieben wurden, verbleibt die Datenbank in einem inkonsistenten und unbrauchbaren Status. Die *Wiederherstellung nach einem Systemabsturz* versetzt die Datenbank wieder in einen konsistenten und verwendbaren Status. Dies geschieht durch Rückgängigmachen (ROLLBACK) der unvollständigen Transaktionen und Beenden der festgeschriebenen Aktionen, die sich zum Zeitpunkt des Systemabsturzes noch im Hauptspeicher befanden (Abb. 2). Wenn eine Datenbank sich in einem konsistenten und verwendbaren Status befindet, hat sie einen Punkt erreicht, der als „Konsistenzzustand" bezeichnet wird.

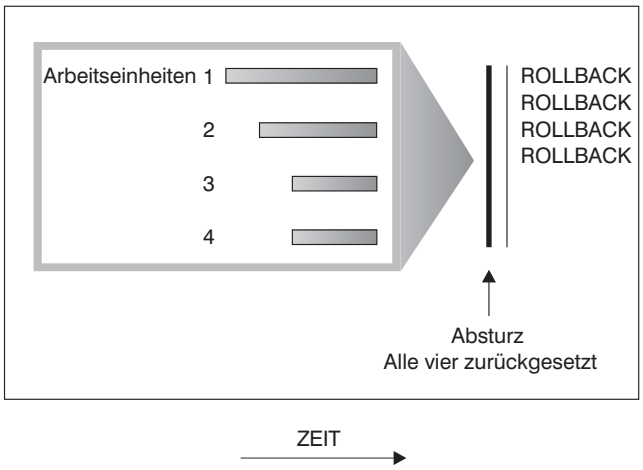


Abbildung 2. Rückgängigmachen von Arbeitseinheiten (Wiederherstellung nach einem Systemabsturz)

Ein *Transaktionsfehler* ergibt sich aus einem schwer wiegenden Fehler bzw. einer Bedingung, die zur abnormalen Beendigung der Datenbank oder des Datenbankmanagers führt. Durch nur teilweise beendete oder zum Zeitpunkt des Fehlers noch nicht auf Platte geschriebene Arbeitseinheiten wird die Datenbank inkonsistent. Daher muss die Datenbank nach einem Transaktionsfehler wiederhergestellt werden. Folgende Bedingungen können zu einem Transaktionsfehler führen:

- Ein Stromausfall auf der Maschine, durch den der Datenbankmanager und die Datenbankpartitionen abnormal beendet werden
- Ein Hardwarefehler, z. B. Datenverlust im Hauptspeicher, oder ein Platten-, CPU- oder Netzwerkfehler
- Ein schwer wiegender Betriebssystemfehler, durch den DB2® abnormal beendet wird

Wenn unvollständige Arbeitseinheiten vom Datenbankmanager automatisch rückgängig gemacht werden sollen, aktivieren Sie den Datenbankkonfigurationsparameter für automatischen Wiederanlauf (*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 Wiederherstellung

nach dem Systemabsturz fortgesetzt wird. Der Neustart der Datenbank wird im Protokoll mit den Benachrichtigungen für die Systemverwaltung aufgezeichnet.

Wenn für eine Datenbank, für die die aktualisierende Wiederherstellung aktiviert ist (d. h., der Konfigurationsparameter *logarchmeth1* ist auf einen anderen Wert als `OFF` gesetzt), eine Wiederherstellung nach einem Systemabsturz ausgeführt wird und während dieser Wiederherstellung 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 Wiederherstellung nach dem Systemabsturz wird fortgesetzt. Nach der Beendigung der Wiederherstellung 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.

Zugehörige Referenzen:

- „autorestart - Automatischer Neustart aktiviert" im Handbuch *Systemverwaltung: Optimierung*
- „logarchmeth1 - Primäre Protokollarchivierungsmethode (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*

Wiederherstellung nach einem Systemabsturz - Details

Wiederherstellen beschädigter Tabellenbereiche

Ein beschädigter Tabellenbereich enthält einen oder mehrere Behälter, 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 Behälterprobleme nicht so behoben werden können, dass die ursprünglichen Daten erhalten bleiben, sind nur die folgenden Optionen verfügbar:

- Wiederherstellen der Datenbank
- Wiederherstellen des Katalogtabellenbereichs

  Anmerkungen:

  1. Die Wiederherstellung des Tabellenbereichs ist nur für wiederherstellbare Datenbanken zulässig, da die Datenbank aktualisierend wiedergeherstellt werden muss.
  2. Wenn Sie den Katalogtabellenbereich wiederherstellen, müssen Sie eine aktualisierende Wiederherstellung bis zum Ende der Protokolle ausführen.

Wenn es sich bei dem beschädigten Tabellenbereich *nicht* um den Tabellenbereich des Systemkatalogs handelt, versucht DB2®, 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.

Zugehörige Tasks:

- „Wiederherstellen von Tabellenbereichen in wiederherstellbaren Datenbanken" auf Seite 14
- „Wiederherstellen von Tabellenbereichen in nicht wiederherstellbaren Datenbanken" auf Seite 15

Zugehörige Referenzen:

- „RESTART DATABASE Command" im Handbuch *Command Reference*
- „RESTORE DATABASE" auf Seite 102

Wiederherstellen von Tabellenbereichen in wiederherstellbaren Datenbanken

Wenn eine Wiederherstellung nach Systemabsturz erforderlich ist, wird ein beschädigter Tabellenbereich in den Offlinestatus versetzt, und es kann nicht mehr auf ihn zugegriffen werden. Er wird in den Status *Aktualisierende Wiederherstellung anstehend* versetzt. Wenn keine weiteren Probleme vorliegen, wird der Neustart erfolgreich ausgeführt.

Prozedur:

Verwenden Sie eine der folgenden Prozeduren, um den beschädigten Tabellenbereich verwendbar zu machen:

- Methode 1
  1. Beheben Sie die Fehler an den beschädigten Behältern ohne Verlust der ursprünglichen Daten.
  2. Führen Sie eine aktualisierende Wiederherstellung bis zum Ende der Protokolle für den Tabellenbereich aus.

     Anmerkung: Die Operation zur aktualisierenden Wiederherstellung versucht dabei zunächst, den Tabellenbereich wieder vom Offlinestatus in den normalen Status zurückzuversetzen.
- Methode 2
  1. Beheben Sie die Fehler an den beschädigten Behältern mit oder ohne Verlust der ursprünglichen Daten.
  2. Führen Sie eine Wiederherstellungsoperation für den Tabellenbereich aus.
  3. Führen Sie eine aktualisierende Wiederherstellung bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt für den Tabellenbereich aus.

Zugehörige Konzepte:

- „Wiederherstellen beschädigter Tabellenbereiche" auf Seite 13

Zugehörige Tasks:

- „Wiederherstellen von Tabellenbereichen in nicht wiederherstellbaren Datenbanken" auf Seite 15

Zugehörige Referenzen:

- „RESTART DATABASE Command" im Handbuch *Command Reference*
- „RESTORE DATABASE" auf Seite 102

Wiederherstellen von Tabellenbereichen in nicht wiederherstellbaren Datenbanken

Da eine Wiederherstellung nach einem Systemabsturz notwendig ist und die Protokolle nicht unendlich lange gespeichert werden, kann eine Neustartoperation nur erfolgreich sein, wenn Sie bereit sind, die beschädigten Tabellenbereiche zu löschen. Erfolgreiche Wiederherstellung bedeutet, dass die Protokollsätze, die zum Wiederherstellen eines konsistenten Status der beschädigten Tabellenbereiche erforderlich sind, nicht mehr vorhanden sind, und damit die einzige gültige Aktion für solche Tabellenbereiche das Löschen dieser Tabellenbereiche ist.

Prozedur:

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 Wiederherstellungsoperation löschen.

   Die Neustartoperation wird ohne Wiederherstellung 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. Er wird nur dann in diesen Status versetzt, wenn der Tabellenbereich während des Neustarts beschädigt ist.

3. Ist die Neustartoperation 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 zurückfordern, 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 Behältern (ohne Verlust der ursprünglichen Daten).
   - Setzen Sie den Befehl RESTART DATABASE erneut ab.
   - Führen Sie eine Operation RESTORE DATABASE aus.

Zugehörige Konzepte:

- „Wiederherstellen beschädigter Tabellenbereiche" auf Seite 13

Zugehörige Tasks:

- „Wiederherstellen von Tabellenbereichen in wiederherstellbaren Datenbanken" auf Seite 14

Zugehörige Referenzen:

- „RESTART DATABASE Command" im Handbuch *Command Reference*
- „RESTORE DATABASE" auf Seite 102

Begrenzen der Auswirkungen von Datenträgerausfällen

Zur Verringerung der Wahrscheinlichkeit eines Datenträgerfehlers und zur Vereinfachung der Wiederherstellung 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 auf dem Katalogknoten ein genau definiertes Verfahren vor. Da dieser Knoten 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 Sicherungen.
  - 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 über den Einsatz von Systemen nachdenken, 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 Plattencontroller 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 %.

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 Spiegelprotokoll-

pfad anzugeben. In Fällen, in denen Zuverlässigkeit von essenzieller Bedeutung ist (d. h., dass keine Zeit für eine Wiederherstellung 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.

Achtung:
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

Zur Verringerung der Auswirkungen von Transaktionsfehlern versuchen Sie, Folgendes sicherzustellen:

- Eine ununterbrochene Stromversorgung für jeden DB2®-Server
- Ausreichenden Plattenspeicherplatz für Datenbankprotokolle auf allen Partitionen
- Zuverlässige Kommunikationsverbindungen zwischen den Datenbankpartitionsservern in einer Umgebung mit partitionierten Datenbanken
- Synchronisation der Systemuhren in einer Umgebung mit partitionierten Datenbanken

Zugehörige Konzepte:

- „Synchronisieren der Systemuhren in einem System mit partitionierten Datenbanken" auf Seite 142

Wiederherstellen nach Transaktionsfehlern in einer Umgebung mit partitionierten Datenbanken

Tritt ein Transaktionsfehler in einer Umgebung mit partitionierten Datenbanken auf, ist in der Regel eine Datenbankwiederherstellung sowohl auf dem ausgefalle-

nen Datenbankpartitionsserver als auch auf allen anderen, an der Transaktion beteiligten Datenbankpartitionsservern erforderlich:

- Eine Wiederherstellung nach Systemabsturz wird auf dem ausgefallenen Datenbankpartitionsserver ausgeführt, nachdem die vorausgegangene Bedingung korrigiert wurde.
- Die *Wiederherstellung 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 Anwendung übergeben wird, der Koordinatorknoten, und der erste Agent, der für die Anwendung aktiv wird, 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 mit Hilfe des Protokolls zur zweiphasigen Festschreibung fest. Während der ersten Phase sendet der Koordinatorknoten 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 die Festschreibung vorbereitet.

Wenn einer der Server mit `NO` antwortet, wird die Transaktion rückgängig gemacht. Andernfalls beginnt der Koordinatorknoten mit der zweiten Phase.

Während der zweiten Phase schreibt der Koordinatorknoten einen COMMIT-Protokollsatz und sendet anschließend eine COMMIT-Anforderung an alle Server, die mit `YES` geantwortet haben. Wenn alle anderen Datenbankpartitionsserver die COMMIT-Operation ausgeführt haben, senden sie eine Bestätigung der Festschreibung an den Koordinatorknoten. Die Transaktion ist abgeschlossen, wenn der Koordinatoragent von allen beteiligten Servern die COMMIT-Bestätigungen empfangen hat. Wenn dies der Fall ist, schreibt der Koordinatoragent einen FORGET-Protokollsatz.

Wiederherstellung 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 der Koordinatorknoten für eine Anwendung ist und die Anwendung auf dem ausgefallenen Datenbankpartitionsserver ausgeführt wird (und nicht zum Festschreiben (COMMIT) bereit ist), wird der Koordinatoragent unterbrochen, um Wiederherstellungsmaßnahmen durchzuführen. Wenn der Koordinatoragent in der zweiten Phase der COMMIT-Verarbeitung ist, empfängt die Anwendung die SQL-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 SQL-Nachricht SQL1229N.
- Wenn der ausgefallene Datenbankpartitionsserver der Koordinatorknoten für die Anwendung war, werden Agenten, die noch für die Anwendung auf den aktiven Servern arbeiten, unterbrochen, um Wiederherstellungsmaßnahmen durch-

zuführen. Die aktuelle Transaktion wird lokal auf jedem Server rückgängig gemacht, sofern sie nicht durch eine PREPARE-Anforderung vorbereitet wurde und auf das Ergebnis der Transaktion wartet. In diesem Fall bleibt die Transaktion auf den aktiven Datenbankpartitionsservern unbestätigt, und der Koordinatorknoten weiß nichts davon (weil er 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 der Koordinatorknoten ist, werden Agenten, die für diese Anwendung aktiv sind, unterbrochen. Der Koordinatorknoten sendet eine Nachricht über eine ROLLBACK-Operation oder einen Trennvorgang an die anderen Datenbankpartitionsserver. Die Transaktion ist nur auf den Datenbankpartitionsservern unbestätigt, die weiterhin aktiv sind, wenn der Koordinatorknoten die SQL-Nachricht SQL0279 zurückgibt.

Jeder Prozess (wie z. B. ein Agent oder ein Detektor für gegenseitiges Sperren), der versucht, eine Anforderung an den ausgefallenen Server zu senden, wird informiert, dass er die Anforderung nicht senden kann.

Wiederherstellung 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 auf einer anderen Partition 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 Wiederherstellung 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 Wiederherstellung 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 der Koordinatorknoten ist, gilt eine Transaktion als unbestätigt, wenn sie zwar vorbereitet (PREPARE), aber noch nicht festgeschrieben (COMMIT) wurde.
- Auf dem Koordinatorknoten 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 COMMIT-Bestätigungen von allen Servern empfangen hat, die für die Anwendung aktiv waren.

Bei der Wiederherstellung nach einem Systemabsturz wird versucht, alle unbestätigten Transaktionen durch eine der folgenden Aktionen aufzulösen. Die durchge-

führte Aktion ist davon abhängig, ob der Datenbankpartitionsserver der Koordinatorknoten für eine Anwendung war:

- Wenn der Server, der erneut gestartet wird, nicht der Koordinatorknoten für die Anwendung ist, sendet er eine Abfragenachricht an den Koordinatoragenten, um das Ergebnis der Transaktion festzustellen.
- Wenn der erneut gestartete Server der Koordinatorknoten für die Anwendung *ist*, sendet er eine Nachricht an alle anderen Agenten (untergeordneten Agenten), von denen der Koordinatoragent immer noch COMMIT-Bestätigungen erwartet.

Es ist möglich, dass durch eine Wiederherstellung nach einem Systemabsturz nicht alle unbestätigten Transaktionen aufgelöst werden können (z. B., wenn einige der Datenbankpartitionsserver nicht verfügbar sind). 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 Wiederherstellung 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 zu machen.

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. Mit Hilfe 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 COMMIT-Verarbeitung beendet wird.

SQL1224N
Dieser SQLCODE-Wert wird empfangen, wenn der ausgefallene Datenbankpartitionsserver der Koordinatorknoten für die Transaktion ist.

SQL1229N
Dieser SQLCODE-Wert wird empfangen, wenn der ausgefallene Datenbankpartitionsserver nicht der Koordinatorknoten für die Transaktion ist.

Welcher Datenbankpartitionsserver ausgefallen ist, kann in zwei Schritten festgestellt werden. 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`.) Auf dem Datenbankpartitionsserver, der den Fehler feststellt, wird eine Nachricht mit der Knotennummer des ausgefallenen Servers in das Protokoll mit den Benachrichtigungen für die Systemverwaltung geschrieben.

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.

Zugehörige Konzepte:

- „Zweiphasige Festschreibung" im Handbuch *Systemverwaltung: Konzept*
- „Fehlerbehebung während einer zweiphasigen Festschreibung" im Handbuch *Systemverwaltung: Konzept*

Zugehörige Tasks:

- „Manuelles Auflösen unbestätigter Transaktionen" im Handbuch *Systemverwaltung: Konzept*

Zugehörige Referenzen:

- „db2start - Start DB2 Command" im Handbuch *Command Reference*
- „LIST INDOUBT TRANSACTIONS Command" im Handbuch *Command Reference*

Wiederherstellen nach dem Ausfall eines Datenbankpartitionsservers

Prozedur:

Gehen Sie wie folgt vor, um einen Datenbankpartitionsserver nach einem Ausfall wieder verfügbar zu machen:

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.

Zugehörige Konzepte:

- „Wiederherstellen nach Transaktionsfehlern in einer Umgebung mit partitionierten Datenbanken" auf Seite 18

Zugehörige Referenzen:

- „db2start - Start DB2 Command" im Handbuch *Command Reference*
- „RESTART DATABASE Command" im Handbuch *Command Reference*

Wiederherstellen von unbestätigten Transaktionen auf dem Host, wenn bei DB2 Connect der DB2-Synchronisationspunktmanager konfiguriert ist

Wenn Ihre Anwendung während einer Transaktion auf einen Host- oder iSeries-Datenbankserver zugegriffen hat, gibt es bei der Wiederherstellung unbestätigter Transaktionen einige Unterschiede.

Für den Zugriff auf Host- oder iSeries-Datenbankserver wird DB2 Connect verwendet. Die Schritte zur Wiederherstellung unterscheiden sich, wenn bei DB2 Connect der DB2-Synchronisationspunktmanager konfiguriert ist.

Prozeduren:

Die Wiederherstellung unbestätigter Transaktionen auf Host- oder iSeries-Servern wird normalerweise automatisch vom Transaktionsmanager (TM) und dem DB2-Synchronisationspunktmanager (SPM) durchgeführt. Eine unbestätigte Transaktion auf einem Host- oder iSeries-Server belegt keine Ressourcen auf der lokalen DB2-Station, belegt jedoch Ressourcen auf dem Host- oder iSeries-Server, solange die Transaktion auf dem betreffenden Server unbestätigt bleibt. Wenn der Administrator des Host- oder iSeries-Servers bestimmt, dass eine heuristische Maßnahme durchzuführen ist, kann er mit dem lokalen DB2-Datenbankadministrator (z. B. per Telefon) in Kontakt treten, um festzustellen, ob die Transaktion auf dem Host- oder iSeries-Server festzuschreiben (COMMIT) oder rückgängig zu machen (ROLLBACK) ist. Wenn dies geschieht, kann der Befehl LIST DRDA INDOUBT TRANSACTIONS verwendet werden, um den Status der Transaktion auf dem lokalen DB2 Connect-Exemplar zu ermitteln. In den meisten Fällen können Sie in einer SNA-Kommunikationsumgebung folgendermaßen vorgehen:

1. Stellen Sie wie nachstehend gezeigt eine Verbindung zum SPM her:

```
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 iSeries-Server. Die Partner-LU (`partner_lu`) ist der vollständig qualifizierte LU-Name des Host- oder iSeries-Servers. Dies stellt die beste Identifikation des Host- oder iSeries-Servers zur Verfügung und sollte vom Anrufer vom Standort des Host- oder iSeries-Servers bereitgestellt werden. Die Angabe `luwid` ist eine eindeutige Kennung für eine Transaktion und steht auf allen Host- und iSeries-Servern zur Verfügung. 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.
   ```

   dann wurde die Transaktion rückgängig gemacht.

   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 diesem Fall sollten Sie den Parameter WITH PROMPTING verwenden, um die Transaktion auf der DB2 Connect-Workstation festzuschreiben oder rückgängig zu machen. Erlauben Sie DB2 Connect anschließend die Resynchronisation mit dem Host- oder iSeries-Server auf der Basis der heuristischen Entscheidung.

Zugehörige Tasks:

- „Wiederherstellen von unbestätigten Transaktionen auf dem Host, wenn DB2 Connect nicht den DB2-Synchronisationspunktmanager verwendet" auf Seite 24

Zugehörige Referenzen:

- „db2start - Start DB2 Command" im Handbuch *Command Reference*
- „LIST INDOUBT TRANSACTIONS Command" im Handbuch *Command Reference*
- „RESTART DATABASE Command" im Handbuch *Command Reference*

Wiederherstellen von unbestätigten Transaktionen auf dem Host, wenn DB2 Connect nicht den DB2-Synchronisationspunktmanager verwendet

Wenn Ihre Anwendung während einer Transaktion auf einen Host- oder iSeries-Datenbankserver zugegriffen hat, gibt es bei der Wiederherstellung unbestätigter Transaktionen einige Unterschiede.

Für den Zugriff auf Host- oder iSeries-Datenbankserver wird DB2 Connect verwendet. Die Schritte zur Wiederherstellung unterscheiden sich, wenn bei DB2 Connect der DB2-Synchronisationspunktmanager konfiguriert ist.

Prozedur:

Verwenden Sie die Informationen in diesem Abschnitt, wenn die TCP/IP-Konnektivität verwendet wird, um DB2 für z/OS im Rahmen einer Aktualisierung auf mehreren Systemen über DB2 Connect Personal Edition oder DB2 Connect Enterprise Server Edition zu aktualisieren, und der DB2-Synchronisationspunktmanager nicht verwendet wird. Die Wiederherstellung 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.

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 (abhängig vom gewünschten Zweck) den Befehl RECOVER THREAD(<LUWID>) ACTION(ABORT|COMMIT) ab.

   Weitere Informationen zum Befehl RECOVER THREAD finden Sie im Handbuch *DB2 for z/OS Command Reference*.

Zugehörige Tasks:

- „Wiederherstellen von unbestätigten Transaktionen auf dem Host, wenn bei DB2 Connect der DB2-Synchronisationspunktmanager konfiguriert ist" auf Seite 23

Zugehörige Referenzen:

- „LIST INDOUBT TRANSACTIONS Command" im Handbuch *Command Reference*

Wiederherstellen nach einem Katastrophenfall

Unter dem Begriff *Wiederherstellung nach einem Katastrophenfall* werden die Aktivitäten zusammengefasst, die zur Wiederherstellung der Datenbank nach einem Brand, einem Erdbeben, nach Vandalismus oder anderen zerstörerischen Ereignissen erforderlich sind. Ein Plan zur Wiederherstellung nach einem Katastrophenfall kann Folgendes vorsehen:

- Einen Zweitstandort, der im Notfall zur Verfügung steht
- Eine andere Maschine, auf der die Datenbank wiederhergestellt werden kann
- Aufbewahrung der Datenbanksicherungen und archivierten Protokolle an einem anderen Standort

Wenn Ihr Plan zur Wiederherstellung nach einem Katastrophenfall vorsieht, die gesamte Datenbank auf einer anderen Maschine wiederherzustellen, benötigen Sie zumindest eine vollständige Datenbanksicherung und alle archivierten Protokolldateien für die Datenbank. Es kann sinnvoll sein, eine Bereitschaftsdatenbank ebenfalls auf dem aktuellen Stand zu halten, indem die Protokolle, die archiviert werden, auf sie angewendet werden. Oder Sie könnten die Datenbanksicherung und Protokollarchive auf dem Bereitschaftssystem speichern und eine Wiederherstellung bzw. eine aktualisierende Wiederherstellung nur ausführen, wenn ein Katastrophenfall eingetreten ist. (In diesem Fall ist eine möglichst aktuelle Sicherung

von Vorteil.) Bei Eintritt eines Katastrophenfalls ist es gewöhnlich jedoch nicht möglich, alle Transaktionen bis zum Zeitpunkt des Katastrophenfalls wiederherzustellen.

Der Nutzen einer Tabellenbereichssicherung zur Wiederherstellung nach einem Katastrophenfall hängt vom Ausmaß der Beschädigung ab. In der Regel ist für die Fehlerbehebung die Wiederherstellung der gesamten Datenbank erforderlich, das heißt, dass eine Gesamtsicherung der Datenbank am Bereitschaftsstandort zur Hand sein sollte. Selbst wenn Sie ein separates Sicherungsimage jedes Tabellenbereichs haben, können Sie diese Images nicht zur Wiederherstellung der Datenbank verwenden. Im Fall einer beschädigten Platte kann mit Hilfe einer Tabellenbereichssicherung jedes Tabellenbereichs auf dieser Platte die Wiederherstellung durchgeführt werden. Wenn Sie aufgrund eines Plattenfehlers (oder aus einem anderen Grund) keinen Zugriff auf einen Behälter mehr haben, können Sie den Behälter 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.

DB2 stellt Ihnen zur Planung der Wiederherstellung nach einem Katastrophenfall mehrere Optionen zur Auswahl. Abhängig von Ihren Geschäftsanforderungen können Sie entweder Tabellenbereichssicherungen oder Datenbankgesamtsicherungen als Schutz vor Datenverlust verwenden oder überlegen, ob sich Ihre Umgebung für eine Lösung wie HADR eignet. Für welche Lösung Sie sich auch entscheiden, Sie sollten Ihre Wiederherstellungsprozeduren in jedem Fall in einer Testumgebung testen, bevor Sie sie in Ihrer Produktionsumgebung implementieren.

Zugehörige Konzepte:

- „Umdefinieren von Tabellenbereichsbehältern während einer Wiederherstellungsoperation (umgeleitete Wiederherstellung)" auf Seite 100
- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201

Versionswiederherstellung

Eine *Versionswiederherstellung* ist die Wiederherstellung einer früheren Version der Datenbank mit Hilfe eines Images der Datenbank, das im Rahmen einer Sicherungsoperation erstellt wurde. Sie können diese Wiederherstellungsmethode 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 Datenbankwiederherstellungsoperation wird die gesamte Datenbank mit Hilfe eines zu einem früheren Zeitpunkt erstellten Sicherungsimages erneut erstellt. Eine Datenbanksicherung ermöglicht es Ihnen, eine Datenbank in dem Status wiederherzustellen, in dem sie sich zum Zeitpunkt der Sicherung befand. Es gehen jedoch alle Arbeitseinheiten verloren, die vom Zeitpunkt der Sicherung ausgehend bis zum Eintreten des Fehlers ausgeführt wurden (siehe Abb. 3 auf Seite 27).

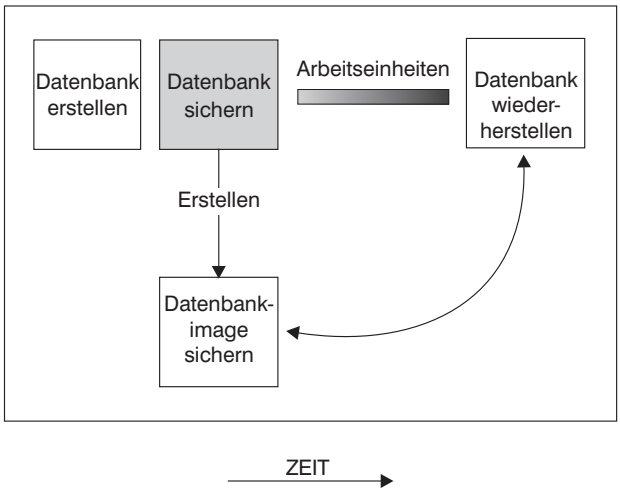


Abbildung 3. Versionswiederherstellung. Die Datenbank wird aus dem zuletzt erstellten Sicherungsimage wiederhergestellt, es gehen jedoch alle Arbeitseinheiten verloren, die vom Zeitpunkt der Sicherung ausgehend bis zum Eintreten des Fehlers ausgeführt wurden.

Bei Einsatz der Methode der Versionswiederherstellung müssen Sie regelmäßige Gesamtsicherungen der Datenbank planen und ausführen.

In einer Umgebung mit partitionierten Datenbanken ist eine Datenbank auf viele Datenbankpartitionsserver (oder Knoten) verteilt. Sie müssen alle Partitionen wiederherstellen, und die Sicherungsimages, die Sie zur Wiederherstellung der Datenbank verwenden, müssen alle zum gleichen Zeitpunkt erstellt worden sein. (Jede Datenbankpartition wird separat gesichert und wiederhergestellt.) Eine Sicherung, bei der alle Sicherungskopien der Datenbankpartitionen zur gleichen Zeit erstellt werden, wird als *Versionssicherung* bezeichnet.

Aktualisierende Wiederherstellung

Sie können die Methode der *aktualisierenden Wiederherstellung* nur dann einsetzen, wenn Sie eine Sicherung der Datenbank erstellt und die Protokolldateien archiviert haben (dazu muss einer der Datenbankkonfigurationsparameter *logarchmeth1* oder *logarchmeth2* auf einen anderen Wert als `OFF` gesetzt sein). Das Wiederherstellen der Datenbank unter Angabe der Option WITHOUT ROLLING FORWARD (d. h. ohne aktualisierende Wiederherstellung) entspricht der Methode der Versionswiederherstellung. Die Datenbank wird in einem Status wiederhergestellt, der dem Zeitpunkt entspricht, zu dem das Offlinesicherungsimage erstellt wurde. Wenn Sie die Datenbank wiederherstellen und bei der Wiederherstellung der Datenbank die Option WITHOUT ROLLING FORWARD *nicht* angeben, befindet sich die Datenbank nach Abschluss der Wiederherstellung im Status *Aktualisierende Wiederherstellung anstehend*. Dadurch kann die aktualisierende Wiederherstellung ausgeführt werden.

Anmerkung: Die Option WITHOUT ROLLING FORWARD kann nicht verwendet werden, wenn die Sicherung der Datenbank erstellt wurde, als die Datenbank online war.

Es können zwei Typen der aktualisierenden Wiederherstellung in Erwägung gezogen werden:

- *Aktualisierende Wiederherstellung der Datenbank*. Bei diesem Typ der aktualisierenden Wiederherstellung werden im Anschluss an die Wiederherstellung der

Datenbank Transaktionen angewendet, die in den Datenbankprotokollen aufgezeichnet sind (siehe Abb. 4). In den Datenbankprotokollen werden alle Änderungen aufgezeichnet, die an der Datenbank vorgenommen werden. Diese Methode vervollständigt die Wiederherstellung 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 auf der Partition abgesetzt werden, auf der sich die Katalogtabellen für die Datenbank befinden (Katalogtabellenpartition). Wenn Sie eine aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt ausführen, müssen alle Datenbankpartitionen aktualisierend wiederhergestellt werden, um sicherzustellen, dass alle Partitionen den gleichen Stand haben. Wenn Sie eine einzelne Datenbankpartition wiederherstellen müssen, können Sie eine aktualisierende Wiederherstellung bis zum Ende der Protokolle durchführen, um die Partition auf den gleichen Stand wie die anderen Partitionen in der Datenbank zu bringen. Bei der aktualisierenden Wiederherstellung einer einzelnen Datenbankpartition kann nur die Wiederherstellung bis zum Ende der Protokolle verwendet werden. Die Wiederherstellung bis zu einem bestimmten Zeitpunkt wird immer auf *alle* Datenbankpartitionen angewendet.

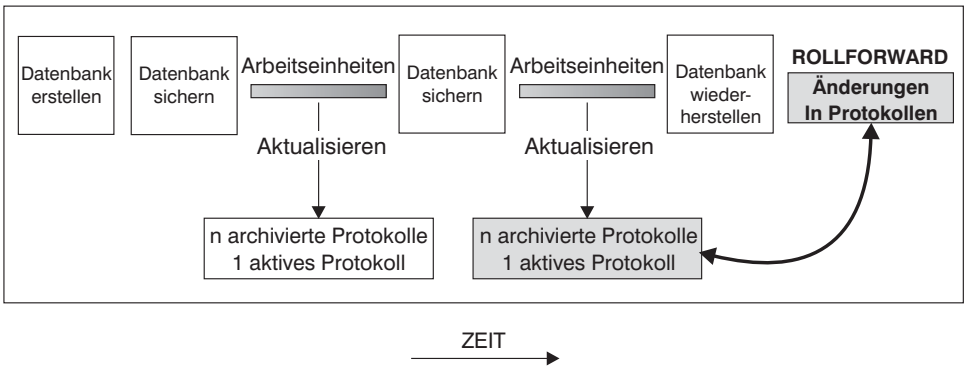


Abbildung 4. Aktualisierende Wiederherstellung einer Datenbank. Bei einer zeitintensiven Transaktion kann mehr als eine Protokolldatei aktiv sein.

- *Aktualisierende Wiederherstellung eines Tabellenbereichs*. Wenn für eine Datenbank die aktualisierende Wiederherstellung aktiviert ist, können auch Tabellenbereiche gesichert, wiederhergestellt und aktualisierend wiederhergestellt werden (siehe Abb. 5 auf Seite 29). Zum Wiederherstellen und aktualisierenden Wiederherstellen eines Tabellenbereichs benötigen Sie ein Sicherungsimage 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:
  - Dem Ende der Protokolldateien
  - Einem bestimmten Zeitpunkt (als Wiederherstellung *bis zu einem bestimmten Zeitpunkt* bezeichnet)

Eine aktualisierende Wiederherstellung von Tabellenbereichen kann in den beiden folgenden Situationen ausgeführt werden:

- Ein Tabellenbereich befindet sich nach seiner Wiederherstellung immer im Status *Aktualisierende Wiederherstellung anstehend* und muss aktualisierend wiederhergestellt werden. Rufen Sie den Befehl ROLLFORWARD DATABASE auf, um die

Protokolle auf die Tabellenbereiche entweder bis zu einem Zeitpunkt oder bis zum Ende der Protokolldateien anzuwenden.

- Wenn sich mindestens ein Tabellenbereich nach einer Wiederherstellung infolge eines Systemabsturzes im Status *Aktualisierende Wiederherstellung anstehend* befindet, beheben Sie zuerst das Problem mit dem Tabellenbereich. In einigen Fällen kann ein Fehler am Tabellenbereich ohne Wiederherstellung der Datenbank behoben werden. Beispielsweise kann ein Spannungsverlust den Tabellenbereich in den Status *Aktualisierende Wiederherstellung anstehend* versetzen. Eine Wiederherstellung 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 Wiederherstellung nach einem Systemabsturz behoben wird, reicht diese Wiederherstellung 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 Wiederherstellung bis zum Ende der Protokolldateien ausführen.

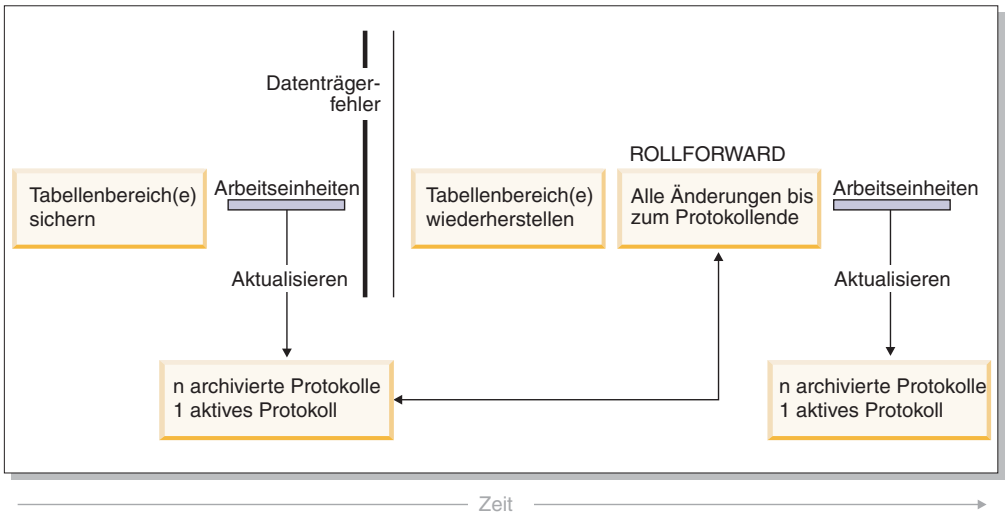


Abbildung 5. Aktualisierende Wiederherstellung von Tabellenbereichen. Bei einer zeitintensiven Transaktion kann mehr als eine Protokolldatei aktiv sein.

Wenn Sie in einer Umgebung mit partitionierten Datenbanken einen Tabellenbereich bis *zu einem bestimmten Zeitpunkt* aktualisierend wiederherstellen, müssen Sie die Liste der Knoten (Datenbankpartitionen) nicht angeben, auf die sich der Tabellenbereich verteilt. DB2® übergibt die Anforderung zur aktualisierenden Wiederherstellung an alle Partitionen. Dies bedeutet, dass der Tabellenbereich auf 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 Partitionen aktualisierend wiederherstellen wollen. Wenn Sie alle Tabellenbereiche in allen Partitionen, die sich im Status *Aktualisierende Wiederherstellung*

anstehend befinden, bis zum Ende der Protokolle aktualisierend wiederherstellen wollen, müssen Sie die Liste der Datenbankpartitionen nicht angeben. Standardmäßig wird die Anforderung zur aktualisierenden Wiederherstellung der Datenbank an alle Partitionen gesendet.

Zugehörige Konzepte:

- „Wiederherstellungsprotokolle" auf Seite 37

Zugehörige Referenzen:

- „ROLLFORWARD DATABASE" auf Seite 144

Zugehörige Beispiele:

- „dbrecov.out -- HOW TO RECOVER A DATABASE (C)"
- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.out -- HOW TO RECOVER A DATABASE (C++)"
- „dbrecov.sqC -- How to recover a database (C++)"

Teilsicherung und Teilwiederherstellung

Da die Größe von Datenbanken, speziell die von Warehouses, in zunehmenden Maße Terabyte- und Petabyte-Bereiche erreicht, steigen die für die Sicherung und Wiederherstellung 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 Tabellenbereichssicherungen keine weiteren Änderungen an anderen, bei der Sicherung nicht berücksichtigten Tabellenbereichen vorgenommen wurden.

Um dieser Problematik gerecht zu werden, bietet DB2® die Möglichkeit der Teilsicherung und Teilwiederherstellung. Eine *Teilsicherung* ist ein Sicherungsimage, das nur die Seiten enthält, die seit der letzten Sicherung aktualisiert wurden. Neben aktualisierten Daten und Indexseiten enthält jedes Teilsicherungsimage auch die vollständigen ursprünglichen Metadaten der Datenbank (wie Datenbankkonfiguration, Tabellenbereichsdefinitionen, Datenbankprotokolle usw.), die normalerweise in Gesamtsicherungsimages enthalten sind.

Anmerkung: Enthält ein Tabellenbereich Daten von Langfeldern oder großen Objekten und wird eine Teilsicherung erstellt, werden alle Daten von Langfeldern oder großen Objekten in das Sicherungsimage kopiert, wenn irgendwelche Seiten im betreffenden Tabellenbereich seit der letzten Sicherung geändert wurden.

Es werden zwei Teilsicherungstypen unterstützt:

- *Inkrementell*. Bei einer solchen Teilsicherung enthält das Image alle Datenbankdaten, die seit der zuletzt erfolgreich ausgeführten Gesamtsicherung geändert wurden. Dies wird auch als ein kumulatives Sicherungsimage bezeichnet, da die über einen gewissen Zeitraum hinweg erstellten Teilsicherungen jeweils den

Inhalt des vorherigen Teilsicherungsimages enthalten. Der Vorgänger eines Teilsicherungsimages ist jeweils die neueste erfolgreiche Gesamtsicherung desselben Objekts.

- *Delta*. Ein Deltasicherungsimage, oder eine Deltateilsicherung, ist eine Kopie aller Datenbankdaten, die seit der zuletzt erfolgreich ausgeführten Sicherung (gesamt, inkrementell oder delta) des betreffenden Tabellenbereichs geändert wurden. Dies wird auch als differenzielles oder nicht kumulatives Sicherungsimage bezeichnet. Der Vorgänger eines Deltasicherungsimages ist die neueste erfolgreiche Sicherung, die eine Kopie aller im Deltasicherungsimage enthaltenen Tabellenbereiche enthält.

Der wesentliche Unterschied zwischen inkrementellen und Deltasicherungsimages wird deutlich beim Erstellen von aufeinander folgenden Sicherungen eines Objekts, das kontinuierlichen Änderungen unterworfen ist. Jedes weitere erstellte inkrementelle Image enthält den vollständigen Inhalt des zuvor erstellten inkrementellen Images sowie alle seit der letzten Gesamtsicherung geänderten oder neu hinzugekommenen Daten. Deltasicherungsimages enthalten hingegen nur die seit der letzten Sicherung (gesamt oder inkrementell) geänderten oder hinzugekommenen Daten.

Kombinationen aus Datenbank- und Tabellenbereichsteilsicherungen sind zulässig, sowohl im Online- als auch im Offlinebetrieb. Gehen Sie bei der Planung Ihrer Sicherungsstrategie sorgfältig vor, da bei der Kombination von Datenbank- und Tabellenbereichsteilsicherungen nicht ausgeschlossen werden kann, dass es sich bei dem Vorgänger eines Datenbanksicherungsimages (oder eines Images mehrerer Tabellenbereiche) anstatt um ein Einzelimage um eine eindeutige Reihe von zu unterschiedlichen Zeitpunkten erstellten Sicherungsimages von Datenbanken und Tabellenbereichen handelt.

Um die Datenbank oder den Tabellenbereich in einem konsistenten Status wiederherzustellen, müssen Sie zur Wiederherstellung ein konsistentes Image des gesamten Objekts (Datenbank oder Tabellenbereich) verwenden und anschließend alle erforderlichen Teilsicherungen in der nachfolgend beschriebenen Reihenfolge anwenden.

Zur Aktivierung der Überwachung von Datenbankaktualisierungen unterstützt DB2 einen neuen Datenbankkonfigurationsparameter, *trackmod*. Dieser Parameter kann einen der beiden folgenden Werte haben:

- NO. Teilsicherungen sind bei dieser Konfiguration nicht zulässig. Aktualisierungen von Datenbankseiten werden weder verfolgt noch aufgezeichnet. Dies ist der Standardwert.
- YES. Teilsicherungen 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 eine Teilsicherung durchgeführt werden kann, muss eine Gesamtsicherung vorgenommen 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 Sicherungsoperation 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.

Zugehörige Tasks:

- „Wiederherstellen von Teilsicherungsimages" auf Seite 32

Teilsicherung und Teilwiederherstellung - Details

Wiederherstellen von Teilsicherungsimages

Prozedur:

Eine Wiederherstellungsoperation, bei der Teilsicherungsimages verwendet werden, umfasst grundsätzlich die folgenden Schritte:

1. Identifizieren des Zielimages der Teilwiederherstellung.

   Ermitteln Sie das endgültig wiederherzustellende Image, und fordern Sie eine Teilwiederherstellungsoperation vom DB2-Wiederherstellungsdienstprogramm an. Dieses Image wird als Zielimage der Teilwiederherstellung bezeichnet, da es das letzte Image ist, das wiederhergestellt wird. Das Zielimage der Teilwiederherstellung wird im Befehl RESTORE DATABASE mit dem Parameter TAKEN AT angegeben.
2. Wiederherstellen der neuesten Gesamtsicherung der Datenbank oder des Tabellenbereichs, um die Basis für die nachfolgend anzuwendenden Teilsicherungsimages zu erstellen.
3. Wiederherstellen aller erforderlichen Gesamt- oder Teilsicherungsimages 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 vollständigen Teilwiederherstellungsoperation 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 Teilwiederherstellungsoperation 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 Wiederherstellungsoperation erstellt wird. In Fällen, in denen ein Tabellenbereich seit der Erstellung des ersten Gesamtsicherungsimages der Datenbank gelöscht wurde, werden die Tabellenbereichsdaten für dieses Image zwar aus den Sicherungsimages gelesen, aber bei der Verarbeitung der Teilwiederherstellung ignoriert.

Teilsicherungsimages können auf zwei verschiedene Weisen wiederhergestellt werden.

- Bei einer automatischen Teilwiederherstellung wird der Befehl RESTORE nur einmal abgesetzt, um das gewünschte Zielimage anzugeben. DB2 ermittelt die übrigen erforderlichen Sicherungsimages anhand des Datenbankprotokolls und stellt sie wieder her.
- Bei einer manuellen Teilwiederherstellung muss der Befehl RESTORE einmal für jedes einzelne Sicherungsimage abgesetzt werden, das wiederhergestellt werden muss (wie in den vorstehenden Schritten erläutert).

Automatische Teilwiederherstellung - Beispiel

Geben Sie zur Wiederherstellung einer Gruppe von Teilsicherungsimages mit dem Befehl RESTORE DATABASE die Option TAKEN AT *zeitmarke* an. Geben Sie die Zeitmarke des Images an, das Sie zuletzt wiederherstellen wollen. Beispiel:

```
db2 restore db sample incremental automatic taken at 20031228152133
```

Dieser Befehl veranlasst, dass das DB2-Wiederherstellungsdienstprogramm automatisch die am Anfang dieses Abschnitts aufgeführten Schritte ausführt. Während der Anfangsphase der Verarbeitung wird das Sicherungsimage mit der Zeitmarke 20001228152133 gelesen, und das Wiederherstellungsdienstprogramm stellt sicher, dass die Datenbank, ihr Protokoll und die Tabellenbereichsdefinitionen vorhanden und gültig sind.

Während der zweiten Verarbeitungsphase wird das Datenbankprotokoll abgefragt, um eine Kette von Sicherungsimages aufzubauen, die zum Ausführen der angeforderten Wiederherstellungsoperation erforderlich sind. Wenn dies aus einem bestimmten Grund nicht möglich ist und DB2 keine vollständige Kette der erforderlichen Images erstellen kann, wird die Wiederherstellungsoperation beendet und eine Fehlernachricht zurückgegeben. In diesem Fall ist eine automatische Teilwiederherstellung nicht möglich, so dass Sie den Befehl RESTORE DATABASE mit der Option INCREMENTAL ABORT absetzen müssen. Dadurch werden alle übrigen Ressourcen bereinigt, so dass Sie mit einer manuellen Teilwiederherstellung fortfahren können.

Anmerkung: Es wird dringend davon abgeraten, die Option FORCE mit dem Befehl PRUNE HISTORY zu verwenden. Die Standardoperation dieses Befehls verhindert, dass Sie Protokolleinträge löschen, die möglicherweise für die Wiederherstellung unter Verwendung des zuletzt erstellten Gesamtsicherungsimages der Datenbank erforderlich sind. Mit der Option FORCE ist es jedoch möglich, Einträge zu löschen, die für eine automatische Wiederherstellungsoperation benötigt werden.

Während der dritten Verarbeitungsphase stellt DB2 alle übrigen Sicherungsimages 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 erneut absetzen bzw. erneut eine manuelle Teilwiederherstellung versuchen.

Manuelle Teilwiederherstellung - Beispiel

Geben Sie zur manuellen Wiederherstellung einer Gruppe von Teilsicherungsimages mit dem Befehl RESTORE DATABASE die Option TAKEN AT *zeitmarke* an, und führen Sie die vorstehend aufgeführten Schritte aus. Beispiel:

```
1. db2 restore database sample incremental taken at <zm>

   Dabei gilt Folgendes:
     <zm> zeigt auf das zuletzt wiederherzustellende Teilsicherungsimage
     (das Zielimage)

2. db2 restore database sample incremental taken at <zm1>

   Dabei gilt Folgendes:
     <zm1> zeigt auf das erste Image der gesamten Datenbank bzw. eines
     Tabellenbereichs.
```

```
3. db2 restore database sample incremental taken at <zmX>

   Dabei gilt Folgendes:
     <zmX> zeigt auf alle Teilsicherungsimages in der Reihenfolge
     ihrer Erstellung.

4. Wiederholen Sie Schritt 3, und stellen Sie alle Teilsicherungsimages bis
   einschließlich Image <zm> wieder her.
```

Wenn Sie eine Wiederherstellungsoperation für eine Datenbank ausführen und zuvor Sicherungsimages von Tabellenbereichen erstellt wurden, müssen die Tabellenbereichsimages in der chronologischen Reihenfolge der Zeitmarken ihrer Sicherung wiederhergestellt werden.

Mit dem Dienstprogramm **db2ckrst** können Sie das Datenbankprotokoll abfragen und eine Liste der Zeitmarken der für eine Teilwiederherstellung erforderlichen Sicherungsimages generieren. Außerdem wird eine vereinfachte Wiederherstellungssyntax für eine manuelle Teilwiederherstellung generiert. Es empfiehlt sich, alle Sicherungsoperationen zu notieren und dieses Dienstprogramm nur zur Orientierung zu verwenden.

Zugehörige Konzepte:

- „Teilsicherung und Teilwiederherstellung" auf Seite 30

Zugehörige Referenzen:

- „RESTORE DATABASE" auf Seite 102
- „db2ckrst - Reihenfolge der Images der Teilwiederherstellung überprüfen" auf Seite 281

Automatische Teilwiederherstellung - Einschränkungen

1. Wenn Sie seit der letzten Sicherung einen Tabellenbereich umbenannt haben, der jetzt für die Wiederherstellung verwendet werden soll, und bei der Wiederherstellung auf Tabellenbereichsebene den neuen Bereichsnamen verwenden, wird die erforderliche Kette der Sicherungsimages nicht korrekt aus dem Datenbankprotokoll generiert, so dass 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 Die automatische Teilrückschreibung kann nicht fortgesetzt werden.
   Ursachencode: "3".
   ```

   Mögliche Lösung: Nehmen Sie eine manuelle Teilwiederherstellung 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 Sicherungsimages hatte. Alle nach diesem Zeitpunkt erstellten Protokolleinträge gehen verloren. Wenn Sie versuchen, eine automatische Teilwiederherstellung auszuführen, für die einige dieser verloren gegangenen Protokolleinträge benötigt würden, versucht das Dienstprogramm RESTORE, eine falsche Sicherungsimagekette 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 Lösungen:

- Nehmen Sie eine manuelle Teilwiederherstellung vor.
- Stellen Sie zuerst die Protokolldatei unter Verwendung von Image <zm4> wieder her, bevor Sie eine automatische Teilwiederherstellung starten.

3. Wenn Sie ein Sicherungsimage aus einer Datenbank in eine andere Datenbank wiederherstellen und anschließend eine Teilsicherung oder Deltasicherung ausführen, können Sie zur Wiederherstellung dieses Sicherungsimages keine automatische Teilwiederherstellung 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 Sicherungsimage der Datenbankdatei
   anhand des Aliasnamens der Quellendatenbank "datenbankalias" und der angegebenen
   Zeitmarke "zeitmarke" gefunden.
   ```

   Mögliche Lösung:

   - Nehmen Sie eine manuelle Teilwiederherstellung vor, und zwar wie folgt:

     ```
     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 Wiederherstellungsoperation in Datenbank B setzen Sie eine Datenbankgesamtsicherung ab, um eine neue Kette von Teilsicherungen zu starten.

Zugehörige Konzepte:

• „Teilsicherung und Teilwiederherstellung” auf Seite 30

Zugehörige Tasks:

• „Wiederherstellen von Teilsicherungsimages” auf Seite 32

Zugehörige Referenzen:

• „RESTORE DATABASE” auf Seite 102

Überwachen des Fortschritts von Operationen zur Sicherung, Wiederherstellung und aktualisierenden Wiederherstellung

Sie können den Befehl LIST UTILITIES verwenden, um Operationen zur Sicherung, Wiederherstellung und aktualisierenden Wiederherstellung für eine Datenbank zu überwachen.

Prozedur:

Gehen Sie wie folgt vor, um den Fortschritt von Operationen zur Sicherung, Wiederherstellung und aktualisierenden Wiederherstellung zu überwachen:

- Setzen Sie den Befehl LIST UTILITIES ab, und geben Sie die Option SHOW DETAIL an.

```
list utilities show detail
```

Es folgt eine Beispielausgabe der Leistungsüberwachung einer Offlinesicherungsoperation für eine Datenbank:

```
LIST UTILITIES SHOW DETAIL

ID                          = 2
Typ                         = BACKUP
Datenbankname               = SAMPLE
Beschreibung                = offline db
Startzeit                   = 2003-10-30 12.55.31.786115
Drosselung:
Priorität                   = Ungedrosselt
Fortschrittsüberwachung:
Geschätzte Fertigstellung (%)    = 41
      Gesamte Arbeit        = 20232453 Byte
      Abgeschlossene Arbeit = 230637 Byte
      Startzeit             = 2003-10-30 12.55.31.786115
```

Für Sicherungsoperationen wird eine erste geschätzte Anzahl zu verarbeitender Byte angegeben. Während der Sicherungsoperation 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 Sicherungsimages verwendet werden. Das tatsächliche Image kann wesentlich kleiner sein, abhängig davon, ob es sich um eine Teilsicherung oder eine komprimierte Sicherung handelt.

Für Wiederherstellungsoperationen wird kein geschätzter Anfangswert angegeben. Stattdessen wird UNBEKANNT angegeben. Während die einzelnen Puffer aus dem Image gelesen werden, wird die tatsächliche Menge gelesener Byte aktualisiert. Für automatische Teilwiederherstellungen, 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 Teilsicherungen wiederhergestellt werden soll. Zu Beginn wird nur eine Phase angegeben. Nachdem das erste Image wiederhergestellt wurde, wird die Gesamtzahl der Phasen angezeigt. Beim Wiederherstellen der einzelnen Images werden die Anzahl verarbeiteter Phasen sowie die Anzahl verarbeiteter Byte aktualisiert.

Für die Wiederherstellung nach einem Systemabsturz und die aktualisierende Wiederherstellung sind zwei Phasen der Fortschrittsüberwachung vorhanden: FORWARD und BACKWARD. In der Phase FORWARD werden Protokolldateien gelesen und die Protokollsätze auf die Datenbank angewendet.

Für die Wiederherstellung nach einem Systemabsturz wird unter Verwendung der Protokollstartfolgenummer bis zum Ende der letzten Protokolldatei der Gesamtaufwand geschätzt. Für die aktualisierende Wiederherstellung wird zu Beginn dieser Phase für den geschätzten Gesamtaufwand UNBEKANNT angegeben. Die Menge der verarbeiteten Byte wird während der 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 Ausführung des Prozesses aktualisiert.

Zugehörige Konzepte:

- „Wiederherstellung nach einem Systemabsturz" auf Seite 12
- „Sicherung - Übersicht" auf Seite 71
- „Wiederherstellung - Übersicht" auf Seite 95
- „Aktualisierende Wiederherstellung - Übersicht" auf Seite 129

Zugehörige Referenzen:

- „LIST UTILITIES Command" im Handbuch *Command Reference*

Wiederherstellungsprotokolle

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

DB2® stellt zwei Protokollierungstypen zur Verfügung: *Umlaufprotokollierung* und *Archivprotokollierung*. Jeder Typ stellt eine andere Stufe der Wiederherstellbarkeit bereit:

- *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 Offlinesicherungen der Datenbank zulässig. Die Datenbank muss offline sein (d. h. für Benutzer nicht zugänglich), wenn eine Gesamtsicherung erstellt wird. Wie der Name bereits andeutet, verwendet die Umlaufprotokollierung einen „Ring" von Onlineprotokollen, die eine Wiederherstellung 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 der letzten Gesamtsicherung durchgeführt wurden, aktualisierend wiederherzustellen. Alle Änderungen, die seit der letzten Sicherung vorgenommen wurden, gehen verloren. Da diese Art der Wiederherstellung die Daten auf dem Stand zum Zeitpunkt der Gesamtsicherung wiederherstellt, wird sie als *Versionswiederherstellung* bezeichnet.

  Abb. 6 auf Seite 38 zeigt, dass die aktive Protokolldatei mit einem Ring aus Protokolldateien arbeitet, wenn die Umlaufprotokollierung aktiv ist.

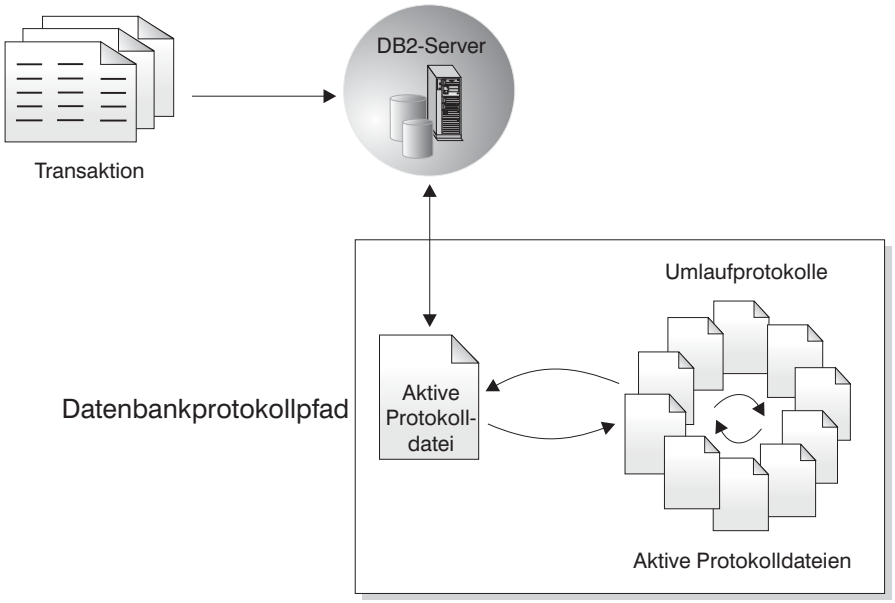


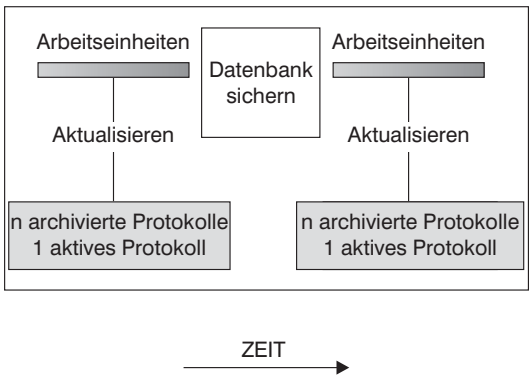
Abbildung 6. Umlaufprotokollierung

Aktive Protokolldateien werden bei der Wiederherstellung 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* wird besonders für die aktualisierende Wiederherstellung verwendet. Archivierte Protokolle sind Protokolle, die aktiv waren, aber nicht mehr zur Wiederherstellung nach einem Systemabsturz erforderlich sind. Verwenden Sie zum Aktivieren der Archivprotokollierung den Datenbankkonfigurationsparameter *logarchmeth1*.

Der Vorteil der Archivprotokollierung liegt darin, dass von der aktualisierenden Wiederherstellung 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 Wiederherstellung von Änderungen verwendet werden, die nach der Sicherung vorgenommen wurden. Bei der Umlaufprotokollierung hingegen ist eine Wiederherstellung nur bis zum Zeitpunkt der letzten Sicherung möglich, alle später vorgenommenen Änderungen gehen verloren.

Das Erstellen von Onlinesicherungen wird nur dann unterstützt, wenn die Datenbank für Archivprotokollierung konfiguriert ist. Während einer Onlinesicherung werden alle Aktivitäten an der Datenbank protokolliert. Wenn eine Onlinesicherung wiederhergestellt wird, muss mit Hilfe der Protokolle die Datenbank mindestens bis zu dem Zeitpunkt aktualisierend wiederhergestellt werden, zu dem die Sicherung abgeschlossen wurde. Dazu müssen die Protokolle archiviert worden sein und bei der Wiederherstellung der Datenbank zur Verfügung stehen. Nach Abschluss einer Onlinesicherung erzwingt DB2 das Schließen des aktuell geöffneten Protokolls, wodurch es archiviert wird. Hierdurch wird sichergestellt, dass Ihrer Onlinesicherung alle zur Wiederherstellung benötigten archivierten Protokolldateien zur Verfügung stehen.



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

Abbildung 7. Aktive und archivierte Datenbankprotokolle bei aktualisierender Wiederherstellung. Bei einer zeitintensiven Transaktion kann mehr als eine Protokolldatei aktiv sein.

Mit den folgenden Datenbankkonfigurationsparametern können Sie die Position ändern, an der Archivprotokolldateien gespeichert werden: *newlogpath*, *logarchmeth1* und *logarchmeth2*. Änderungen am Parameter *newlogpath* wirken sich außerdem auf die Speicherposition aus, an der die aktiven Protokolle gespeichert werden.

Welche der *Protokollspeicherbereiche* im Verzeichnispfad der Datenbankprotokolle 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 *loghead* sind, handelt es sich um archivierte Protokolldateien, die versetzt werden können. Die Werte dieser Parameter können Sie überprüfen, indem Sie die Steuerzentrale verwenden oder mit Hilfe des Befehlszeilenprozessors den Befehl GET DATABASE CONFIGURATION ausführen, um die erste aktive Protokolldatei anzuzeigen. Weitere Informationen zu diesem Konfigurationsparameter finden Sie im Handbuch *Systemverwaltung: Optimierung*.

Zugehörige Konzepte:

- „Spiegeln von Protokollen" auf Seite 39

Zugehörige Referenzen:

- Anhang G, „Benutzerexit zur Datenbankwiederherstellung", auf Seite 369
- „loghead - Erste aktive Protokolldatei" im Handbuch *Systemverwaltung: Optimierung*
- „Konfigurationsparameter für die Datenbankprotokollierung" auf Seite 42
- „logarchmeth1 - Primäre Protokollarchivierungsmethode (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*

Wiederherstellungsprotokolle - Details

Spiegeln von Protokollen

DB2® bietet Unterstützung für das Spiegeln von Protokollen auf Datenbankebene. Durch das Spiegeln von Protokolldateien kann in der Datenbank Folgendes verhindert werden:

- Versehentliches Löschen einer aktiven Protokolldatei
- Datenverlust aufgrund von Hardwarefehlern

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, dass Sie den sekundären Protokollpfad auf einer physisch getrennten Platte einrichten (bevorzugt auf einer Platte, die sich zudem auf einem anderen Plattencontroller befindet). Auf diese Weise wird vermieden, dass der Plattencontroller einen einzelnen Fehlerpunkt darstellt.

Wird MIRRORLOGPATH zum ersten Mal aktiviert, wird der Parameter erst beim nächsten Start der Datenbank verwendet. Gleiches gilt auch für den 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 entsprechenden 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 wird nicht versuchen, den „unbrauchbaren" Pfad erneut zu verwenden, bis die aktuelle Protokolldatei voll ist. Wenn DB2 die nächste Protokolldatei öffnen muss, wird sichergestellt, dass dieser Pfad gültig ist. Ist dies der Fall, wird der Pfad verwendet. Andernfalls wird DB2 erst versuchen, den Pfad erneut zu verwenden, wenn zum ersten Mal auf die folgende Protokolldatei zugegriffen wird. Es wird nicht versucht, die Protokollpfade zu synchronisieren, DB2 erfasst jedoch Informationen zu den aufgetretenen Zugriffsfehlern, so dass beim Archivieren von Protokolldateien die korrekten Pfade verwendet werden. Wenn beim Schreiben in den verbleibenden „brauchbaren" Pfad ebenfalls ein Fehler auftritt, wird die Datenbank abnormal beendet.

Zugehörige Referenzen:

- „mirrorlogpath - Pfad für Protokollspiegelung" im Handbuch *Systemverwaltung: Optimierung*

Verringern der Protokollieraktivitäten mit dem Parameter NOT LOGGED INITIALLY

Wenn Ihre Anwendung Arbeitstabellen aus Originaltabellen erstellt und mit Werten füllt und Sie sich um die Wiederherstellbarkeit dieser Arbeitstabellen keine Gedanken machen müssen, weil sie leicht mit Hilfe der Originaltabellen wieder erstellt werden können, können Sie die Arbeitstabellen unter Angabe der Klausel NOT LOGGED INITIALLY in der Anweisung CREATE TABLE erstellen. Die Verwendung des Parameters NOT LOGGED INITIALLY hat den Vorteil, dass alle Änderungen an der Tabelle (einschließlich Operationen zum Einfügen, Löschen, Aktualisieren sowie Erstellen von Indizes) in der Arbeitseinheit, in der die Tabelle erstellt wird, nicht protokolliert werden. Dadurch werden nicht nur die Protokollieraktivitäten 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.

Anmerkungen:

1. Sie können mehrere Tabellen mit dem Parameter NOT LOGGED INITIALLY in derselben Arbeitseinheit 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 COMMIT-Zeitpunkt auf Platte geschrieben. Das bedeutet, dass die COMMIT-Operation 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 Arbeitseinheit rückgängig gemacht oder eine Operation ROLLBACK TO SAVEPOINT ausgeführt (SQL1476N).
- Diese Tabellen können Sie bei einer aktualisierenden Wiederherstellung nicht wiederherstellen. Wenn die aktualisierende Wiederherstellung auf eine Tabelle stößt, die mit der Option NOT LOGGED INITIALLY erstellt wurde, wird diese Tabelle als nicht verfügbar markiert. Nach der Wiederherstellung 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 eine COMMIT-Operation ausgeführt wird. Die Inaktivierung der Protokollfunktion ist nur dann von Vorteil, wenn Sie die Tabelle in derselben Arbeitseinheit 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 COMMIT-Operation.
- Geben Sie die Option ON COMMIT PRESERVE an, um die Zeilen in der Tabelle nach einer COMMIT-Operation zu behalten. Ansonsten werden alle Zeilen gelöscht.
- Nur die Anwendung, die die deklarierte temporäre Tabelle erstellt, kann auf dieses Exemplar der Tabelle zugreifen.
- Die Tabelle wird implizit gelöscht, wenn die Verbindung der Anwendung zur Datenbank beendet wird.
- Betriebsfehler in einer Arbeitseinheit mit einer deklarierten temporären Tabelle bewirken nicht, dass die Arbeitseinheit 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 Arbeitseinheit (bzw.

eines Sicherungspunkts) löscht alle Zeilen in deklarierten temporären Tabellen, die innerhalb dieser Arbeitseinheit (bzw. dieses Sicherungspunkts) geändert wurden.

Zugehörige Konzepte:

- „Application processes, concurrency, and recovery" im Handbuch *SQL Reference, Volume 1*

Zugehörige Tasks:

- „Erstellen eines Tabellenbereichs" im Handbuch *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „DECLARE GLOBAL TEMPORARY TABLE statement" im Handbuch *SQL Reference, Volume 2*

Konfigurationsparameter für die Datenbankprotokollierung

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 definiert werden, um zu vermeiden, dass Fehlernachrichten wegen einer vollen Protokollplatte generiert werden, wenn DB2 keine neue Protokolldatei im Pfad der aktiven Protokolle erstellen kann. DB2 wird stattdessen alle fünf Minuten erneut versuchen, die Protokolldatei zu erstellen, bis dies erfolgreich ist. Nach jedem Versuch schreibt DB2 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 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 blockiert sind, oder auf eine Datenseite, die von der aktualisierenden Anwendung im Pufferpool fixiert wurde, erscheinen Abfragen mit Lesezugriff ebenfalls blockiert.

Wenn Sie *blk_log_dsk_ful* auf `YES` setzen, blockieren Anwendungen, sobald DB2 einen Fehler wegen einer vollen Protokollplatte feststellt. Sie können diesen Fehler anschließend beheben, und die Transaktion kann fortgesetzt werden. Sie können auf einem vollen Datenträger Speicher freigeben, indem Sie alte Protokolldateien auf ein anderes Dateisystem verschieben, oder indem Sie die Größe des Dateisystems erhöhen, so dass blockierte Anwendungen beendet werden können.

Wenn *blk_log_dsk_ful* auf `NO` gesetzt ist und eine Transaktion einen Fehler wegen einer vollen Protokollplatte erhält, schlägt die Transaktion fehl und wird rückgängig gemacht. In einigen Fällen wird die Datenbank abnormal beendet, wenn eine Transaktion einen Fehler wegen einer vollen Protokollplatte verursacht.

Alternativpfad für Archivprotokolldatei (failarchpath)
Gibt einen Alternativpfad für die Archivprotokolldateien an, wenn die angegebene Protokollarchivierungsmethode fehlschlägt. 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 das Verzeichnis für die Protokollarchivierungsmethode versetzt werden. 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.

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 doppelt gespeichert. Dies bedeutet, dass Sie zwei Kopien der Archivprotokolldateien in zwei verschiedenen Verzeichnissen erhalten.

Zu den gültigen Werten für diese Parameter gehören ein Datenträgertyp und, in einigen Fällen, ein Zielfeld. Verwenden Sie als Trennzeichen zwischen den Werten einen Doppelpunkt (:). Gültige Werte:

OFF Gibt an, dass die Protokollarchivierungsmethode nicht verwendet werden soll. Wenn sowohl *logarchmeth1* als auch *logarchmeth2* auf `OFF` gesetzt ist, wird davon ausgegangen, dass die Datenbank Umlaufprotokolle verwendet; sie kann in diesem Fall nicht aktualisierend wiederhergestellt werden. Dies ist der Standardwert.

LOGRETAIN
Dieser Wert kann nur für *logarchmeth1* verwendet werden und ist äquivalent zur Einstellung `RECOVERY` für den Konfigurationsparameter *logretain*. Wenn Sie diesen Wert angeben, werden die Konfigurationsparameter *logretain* automatisch aktualisiert.

USEREXIT
Dieser Wert kann nur für *logarchmeth1* verwendet werden und ist äquivalent zur Einstellung `ON` für den Konfigurationsparameter *userexit*. Wenn Sie diesen Wert angeben, wird der Konfigurationsparameter *userexit* automatisch aktualisiert.

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 *logarchmeth1* auf `DISK:/u/dbuser/archived_logs` setzen, werden die Archivprotokolldateien in ein Verzeichnis mit dem Namen `/u/dbuser/archived_logs` 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 TSM-Server 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 Sicherungs- und Wiederherstellungs-APIs für Produkte anderer Lieferanten verwenden.

Anmerkungen:

1. Wenn *logarchmeth1* oder *logarchmeth2* auf einen anderen Wert als `OFF` gesetzt ist, ist die Datenbank für die aktualisierende Wiederherstellung konfiguriert.
2. Wenn Sie die Konfigurationsparameter *userexit* oder *logretain* aktualisieren, wird *logarchmeth1* automatisch aktualisiert und umgekehrt. Wenn Sie *userexit* oder *logretain* verwenden, muss *logarchmeth2* auf `OFF` gesetzt werden.

Optionen für logarchmeth1 (logarchopt1), Optionen für logarchmeth2 (logarchopt2)
Gibt eine Zeichenfolge an, die an den TSM-Server oder an APIs anderer Lieferanten übergeben wird. Für TSM wird dieses Feld verwendet, um der Datenbank das Abrufen von Protokollen zu ermöglichen, die auf einem anderen TSM-Knoten oder von einem anderen TSM-Benutzer erstellt wurden. Die Zeichenfolge muss im folgenden Format angegeben werden:

```
"-fromnode=knotenname -fromowner=benutzername"
```

Dabei ist `knotenname` der Name des TSM-Knotens, auf dem die Protokolldateien ursprünglich archiviert wurden, und `benutzername` ist der Name des TSM-Benutzers, der die Protokolldateien ursprünglich archiviert hat. Jedes Protokollarchivierungsoptionsfeld entspricht einer der Protokollarchivierungsmethoden: *logarchopt1* wird mit *logarchmeth1* verwendet und *logarchopt2* mit *logarchmeth2*.

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 führt zu einer effizienteren E/A-Aktivität in Bezug auf die Protokollierung, da die Protokollsätze nun in größeren Abständen und dabei in größeren Mengen auf Platte geschrieben werden.

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

Es besteht eine logische Begrenzung von 256 GB für den konfigurierbaren Gesamtspeicherplatz für aktive Protokolldateien. Diese Begrenzung ergibt sich aus dem oberen Grenzwert für *logfilsiz*, der bei 262144 liegt, und dem oberen Grenzwert für (*logprimary* + *logsecond*), der 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 reinen Leistungsgründen würde daher gelten daher: je größer das Protokoll, desto besser. Dieser Parameter gibt ebenfalls die Protokolldateigröße für die Archivierung an. Hier ist eine große Protokolldatei nicht unbedingt leistungsfördernd, da sie das Fehlerrisiko erhöht bzw. bei der Protokollübertragung zu Verzögerungen führen kann. Für den Speicherbereich für aktive Protokolldateien empfiehlt es sich eher, eine größere Anzahl kleinerer Protokolldateien zu verwenden. Wenn beispielsweise 2 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 DB2 andernfalls unnötigerweise eine große Datei generieren würde, die ohnehin abgeschnitten wird. Sie können diesen Aufwand vermeiden, indem Sie den Befehl ACTIVATE DATABASE verwenden. Wenn Sie den Pufferpool vorbereiten, wirkt sich dies ebenfalls positiv auf die Leistung aus.

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. Von der Archivierung ist stets das gesamte Protokoll betroffen. Wenn Sie ein einzelnes, umfangreiches Protokoll verwenden, 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 Protokollgröße erhöht zwar die Häufigkeit der Archivierungen, kann aber den Informationsverlust bei einem Datenträgerfehler verringern, da die kleineren Protokolle, die vor dem verloren gegangenen Protokoll erstellt wurden, weiterhin verwendet werden können.

Beibehalten von Protokollen (logretain)
Dieser Konfigurationsparameter wurde durch *logarchmeth1* ersetzt. Er wird aus Gründen der Abwärtskompatibilität weiter unterstützt.

Wenn *logretain* auf `RECOVERY` gesetzt ist, werden Archivprotokolldateien im Verzeichnispfad der Datenbankprotokolle beibehalten, und die Datenbank wird als wiederherstellbar betrachtet, d. h., die aktualisierende Wiederherstellung wird aktiviert.

Anmerkung: Die Standardeinstellung für den Datenbankkonfigurationsparameter *logretain* unterstützt keine aktualisierende Wiederherstellung. Wenn Sie die aktualisierende Wiederherstellung verwenden wollen, müssen Sie den Wert dieses Parameters ändern.

Maximales Protokoll pro Transaktion (max_log)
Dieser Parameter zeigt den Prozentsatz des primären Protokollspeicherbereichs an, der von einer Transaktion verbraucht werden kann.

Wird der Wert auf `0` gesetzt, besteht keine Begrenzung des Prozentsatzes des gesamten primären Protokollbereichs, den eine Transaktion verbrau-

chen kann. Verletzt eine Anwendung die Konfiguration *max_log*, wird die Anwendung gezwungen, die Verbindung zur Datenbank zu beenden. Die Transaktion wird rückgängig gemacht und der Fehler SQL1224N wird zurückgegeben.

Sie können dieses Verhalten außer Kraft setzen, indem Sie die Registriervariable DB2_FORCE_APP_ON_MAX_LOG auf *FALSE* setzen. Dies bewirkt, dass Transaktionen, die die Konfiguration *max_log* verletzen, fehlschlagen und den Fehler SQL0964N zurückgeben. Die Anwendung kann weiterhin von vorherigen Anweisungen in der Arbeitseinheit fertiggestellte Arbeit festschreiben oder die fertiggestellte Arbeit wird rückgängig gemacht, um die Arbeitseinheit zu widerrufen.

Anmerkung: Die folgenden DB2-Befehle werden von der Begrenzung ausgenommen, die vom Konfigurationsparameter *max_log* festgelegt wird: ARCHIVE LOG, BACKUP DATABASE, LOAD, REORG TABLE (online), 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, so dass 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 Wiederherstellung konfiguriert ist.

Da Sie den Protokollpfad ändern können, befinden sich die für die aktualisierende Wiederherstellung erforderlichen Protokolle möglicherweise in verschiedenen Verzeichnissen. Sie können diesen Konfigurationsparameter während der aktualisierenden Wiederherstellung 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.

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

Anmerkungen:

1. Dieser Konfigurationsparameter wird nicht unterstützt, wenn es sich bei dem primären Protokollpfad um eine unformatierte Einheit handelt.
2. Der für diesen Parameter angegebene Wert darf keine unformatierte Einheit sein.

Neuer Protokollpfad (newlogpath)

Anfangs werden die Datenbankprotokolle im Unterverzeichnis SQLOGDIR des Datenbankverzeichnisses erstellt. Sie können die Position, an der die aktiven Protokolle sowie die künftigen Archivprotokolldateien gespeichert werden, ändern, indem Sie den Wert für diesen Konfigurationsparameter so ändern, dass er entweder auf ein anderes Verzeichnis oder auf eine andere Einheit zeigt. Aktive Protokolldateien, die momentan im Verzeichnispfad der Datenbankprotokolle gespeichert werden, werden nicht an die neue Position versetzt, wenn die Datenbank für die aktualisierende Wiederherstellung konfiguriert ist.

Da Sie den Protokollpfad ändern können, befinden sich die für die aktualisierende Wiederherstellung erforderlichen Protokolle möglicherweise in verschiedenen Verzeichnissen bzw. auf verschiedenen Einheiten. Sie können diesen Konfigurationsparameter während der aktualisierenden Wiederherstellung 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.

Anzahl der Gruppenfestschreibungen (mincommit)
Mit diesem Parameter können Sie das Schreiben von Protokollsätzen auf die Platte verzögern, bis eine Mindestanzahl von COMMIT-Operationen durchgeführt wurde. 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 Festschreibungen von den Anwendungen innerhalb kurzer Zeit angefordert werden.

Diese Gruppierung von Festschreibungen wird nur dann ausgeführt, wenn der Wert dieses Parameters größer als eins und die Anzahl der Anwendungen, die mit der Datenbank verbunden sind, größer als der Wert dieses Parameters ist. Wenn die Gruppierung von Festschreibungen durchgeführt wird, werden Festschreibungsanforderungen von Anwendungen solange zurückgehalten, bis entweder eine Sekunde vergangen oder die Anzahl der Festschreibungsanforderungen gleich dem Wert dieses Parameters ist.

Anzahl Wiederholungen der Archivierung nach Fehler (numarchretry)
Gibt an, wie oft versucht wird, die Protokolldateien mit der angegebenen 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 aktiver Protokolldateien für UOW (num_log_span)
Dieser Parameter zeigt die Anzahl aktiver Protokolldateien an, die eine aktive Transaktion umfassen kann. Wird der Wert auf `0` gesetzt, besteht keine Begrenzung für die Anzahl der Protokolldateien, die eine einzelne Transaktion umfassen kann. Verletzt eine Anwendung die Konfiguration *num_log_span*, wird die Anwendung gezwungen, die Verbindung zur Datenbank zu beenden und der Fehler SQL1224N wird zurückgegeben.

Anmerkung: Die folgenden DB2-Befehle werden von der Begrenzung ausgenommen, die vom Konfigurationsparameter *num_log_span* festgelegt wird: ARCHIVE LOG, BACKUP DATABASE, LOAD, REORG TABLE (online), 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 DB2 nach Protokolldateien suchen soll, die für eine aktualisierende Wiederherstellung 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 die-

sen 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 DB2 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 DB2 die Protokolldateien in den Pfad für aktive Protokolldateien ab. Wenn Sie für diesen Parameter einen Wert angeben, können Sie DB2 dadurch zusätzliche Ressourcen zum Speichern der abgerufenen Protokolldateien bereitstellen. Dies bietet beispielsweise den Vorteil, dass der Ein-/Ausgabeaufwand auf mehrere Platten verteilt und mehr Protokolldateien im Pfad für aktive Protokolldateien gespeichert werden können.

Wenn Sie beispielsweise die Anwendungsprogrammierschnittstelle **db2ReadLog** zur Replikation verwenden, können Sie mit dem Parameter *overflowlogpath* eine Position angeben, an der DB2 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 in der Datenbankkonfiguration der Parameter *userexit* aktiviert ist, ruft DB2 das Benutzerexitprogramm auf, um die Protokolldatei abzurufen. Sie können mit diesem Parameter auch ein Verzeichnis angeben, in dem DB2 die abgerufenen Protokolldateien speichern soll. Dies bietet beispielsweise den Vorteil, dass der Ein-/Ausgabeaufwand für den Pfad für aktive Protokolldateien reduziert und mehr Protokolldateien im Pfad für aktive Protokolldateien gespeichert werden können.

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

Zum Definieren von *overflowlogpath* geben Sie eine Zeichenfolge mit maximal 242 Byte an. Die Zeichenfolge muss auf einen Pfadnamen zeigen und muss ein vollständig qualifizierter Pfadname sein, kein relativer Pfad. Der Pfadname muss ein Verzeichnis sein, keine unformatierte Einheit.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken wird die Knotennummer automatisch an den Pfad angehängt. Dadurch wird in Konfigurationen mit mehreren logischen Knoten die Eindeutigkeit des Pfadnamens sichergestellt.

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

Ein primäres Protokoll belegt im leeren sowie im vollen Zustand denselben Plattenspeicherplatz. Wenn Sie also mehr Protokolle als erforderlich konfigurieren, wird unnötigerweise Plattenspeicherplatz belegt. Konfigurieren Sie dagegen zuwenig 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 Anwen-

dung 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 Wiederherstellung aktivieren, müssen Sie die Anzahl der primären Protokolle auf den Wert der Summe der primären und sekundären Protokolle plus 1 ändern. Zusätzliche Informationen werden für Felder der Datentypen LONG VARCHAR und LOB in einer Datenbank aufgezeichnet, die für die aktualisierende Wiederherstellung aktiviert ist.

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

Anmerkungen:
1. Die Protokollarchivierung muss aktiviert sein, damit *logsecond* auf -1 gesetzt werden kann.
2. Wenn dieser Parameter auf -1 gesetzt ist, wird die Zeit für die Wiederherstellung nach einem Systemabsturz möglicherweise erhöht, da DB2 Archivprotokolldateien abrufen muss.

Benutzerexit (userexit)
Dieser Konfigurationsparameter wurde durch *logarchmeth1* ersetzt. Er wird aus Gründen der Abwärtskompatibilität weiter unterstützt.

Der Konfigurationsparameter *userexit* weist den Datenbankmanager an, ein Benutzerexitprogramm zum Archivieren und Abrufen von Protokollen aufzurufen. Die Protokolldateien werden an einer anderen Position als dem aktiven Protokollpfad gespeichert. Wenn *userexit* auf `ON` gesetzt ist, ist die aktualisierende Wiederherstellung aktiviert.

Die Datenübertragungsgeschwindigkeit der Einheit, die Sie zum Speichern von Offlinearchivprotokolldateien verwenden, und die Software, mit der Sie die Kopien anlegen, müssen mindestens der Durchschnittsgeschwindigkeit entsprechen, mit der der Datenbankmanager Daten in die Protokolle schreibt. Wenn die Übertragungsgeschwindigkeit für die neuen Protokolldaten, die generiert werden, nicht ausreicht, kann der Plattenspeicherplatz knapp werden. Dieser Fall kann eintreten, wenn die Protokollierungsaktivitäten über einen genügend langen Zeitraum hinweg fortgesetzt werden. Die Zeit bis zum Knappwerden des Plattenspeicherplatzes ist abhängig von der Größe des freien Plattenspeicherplatzes. Reicht der verfügbare Speicherplatz nicht aus, wird die Datenbankverarbeitung gestoppt.

Die Datenübertragungsgeschwindigkeit spielt insbesondere bei der Verwendung von Bändern oder optischen Datenträgern eine bedeutende Rolle. Einige Bandeinheiten benötigen stets dieselbe Zeit zum Kopieren einer Datei, unabhängig davon, wie groß die betreffende Datei ist. Sie müssen die Leistungsfähigkeit Ihrer Archivierungseinheit bestimmen.

Für Bandeinheiten gelten andere Faktoren. Die Häufigkeit der Archivierungsanforderungen ist entscheidend. Wenn z. B. eine Kopieroperation

fünf Minuten in Anspruch nimmt, muss das Protokoll groß genug sein, um bei einer hohen Arbeitsauslastung die Protokolldaten von fünf Minuten aufnehmen zu können. Bedingt durch die Eigenschaften Ihrer Bandeinheit kann zudem die Anzahl der pro Tag ausführbaren Operationen begrenzt sein. Sie müssen diese Faktoren berücksichtigen, wenn Sie die Protokollgröße definieren.

Anmerkungen:

1. Dieser Parameter muss auf `ON` gesetzt werden, um unbegrenzten Speicherbereich für aktive Protokolldateien zu aktivieren.
2. Die Standardeinstellung für den Datenbankkonfigurationsparameter *userexit* unterstützt keine aktualisierende Wiederherstellung und muss geändert werden, falls Sie ihn verwenden wollen.

Zugehörige Konzepte:

- „Verwalten von Protokolldateien" auf Seite 51
- „Verbessern der Wiederherstellungsleistung" auf Seite 67

Zugehörige Referenzen:

- Anhang G, „Benutzerexit zur Datenbankwiederherstellung", auf Seite 369
- „logfilsiz - Protokolldateigröße" im Handbuch *Systemverwaltung: Optimierung*
- „logprimary - Anzahl primärer Protokolldateien" im Handbuch *Systemverwaltung: Optimierung*
- „logsecond - Anzahl sekundärer Protokolldateien" im Handbuch *Systemverwaltung: Optimierung*
- „logbufsz - Protokollpuffergröße" im Handbuch *Systemverwaltung: Optimierung*
- „mincommit - Anzahl der Gruppenfestschreibungen" im Handbuch *Systemverwaltung: Optimierung*
- „newlogpath - Datenbankprotokollpfad ändern" im Handbuch *Systemverwaltung: Optimierung*
- „logretain - Beibehalten von Protokollen aktivieren" im Handbuch *Systemverwaltung: Optimierung*
- „userexit - Benutzerexit aktivieren" im Handbuch *Systemverwaltung: Optimierung*
- „APIs für das Sichern und Wiederherstellen in Speichermanagern" auf Seite 373
- „UPDATE DATABASE CONFIGURATION Command" im Handbuch *Command Reference*
- „mirrorlogpath - Pfad für Protokollspiegelung" im Handbuch *Systemverwaltung: Optimierung*
- „blk_log_dsk_ful - Bei voller Protokollplatte blockieren" im Handbuch *Systemverwaltung: Optimierung*
- „overflowlogpath - Überlaufprotokollpfad" im Handbuch *Systemverwaltung: Optimierung*
- „max_log - Maximale Protokollgröße pro Transaktion (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „num_log_span - Anzahl verwendeter Protokolldateien (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „db2tapemgr - Manage Log Files on Tape Command" im Handbuch *Command Reference*

Verwalten von Protokolldateien

Bei der Verwaltung von Datenbankprotokollen sind folgende Faktoren zu beachten:

- Das Nummerierungsschema für Archivprotokolldateien beginnt mit S0000000.LOG und reicht bis S9999999.LOG und bietet somit Platz für bis zu 10 Millionen Protokolldateien. Der Datenbankmanager beginnt in folgenden Situationen erneut bei S0000000.LOG:
  - Die Konfigurationsdatei der Datenbank wurde geändert, um die aktualisierende Wiederherstellung zu aktivieren.
  - Die Konfigurationsdatei der Datenbank wurde geändert, um die aktualisierende Wiederherstellung zu *inaktivieren*.
  - S9999999.LOG wurde verwendet

  DB2® verwendet die Protokolldateinamen nach dem Wiederherstellen einer Datenbank (mit und ohne aktualisierende Wiederherstellung) erneut. Der Datenbankmanager stellt sicher, dass bei der aktualisierenden Wiederherstellung kein falsches Protokoll verwendet wird. Wenn DB2 nach einer Wiederherstellungsoperation einen Protokolldateinamen erneut verwendet, werden die neuen Protokolldateien in separaten Verzeichnissen archiviert, sodass mehrere Protokolldateien desselben Namens archiviert werden können. Die Position der Protokolldateien wird in der Datei des Wiederherstellungsprotokolls aufgezeichnet, sodass die Dateien während einer aktualisierenden Wiederherstellung angewendet werden können. Sie müssen sicherstellen, dass die richtigen Protokolle für die aktualisierende Wiederherstellung zur Verfügung stehen.

  Wenn die aktualisierende Wiederherstellung erfolgreich ausgeführt wurde, wird das letzte bei der aktualisierenden Wiederherstellung 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 Wiederherstellung 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 Wiederherstellung 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 (mit dem Befehl ACTIVATE DATABASE), schneidet DB2 die aktuelle Protokolldatei ab, sobald alle Anwendungen ihre Verbindung zur Datenbank getrennt haben. Bei der nächsten Verbindungsherstellung einer Anwendung zur Datenbank beginnt DB2 mit der Protokollierung in einer neuen Datei. Wenn auf Ihrem System viele kleine Protokolldateien erstellt werden, ziehen Sie 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. 8 auf Seite 52). Wenn Sie beispielsweise Sicherung 2 wiederherstellen wollen, könnten hierfür zwei verschiedene Protokollfolgen verwendet werden. Wenn Sie während einer vollständigen Wiederherstellung der Datenbank die aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt ausführen 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 einer Onlinesicherung verfü-

gen, die sich über die erste Protokollfolge erstreckt, müssen Sie die aktualisierende Wiederherstellung mit der ersten Protokollfolge beenden.

Wenn Sie nach der Wiederherstellung eine neue Protokollfolge erstellt haben, sind Tabellenbereichssicherungen in den alten Protokollfolgen ungültig. Dies wird normalerweise bei der Wiederherstellung erkannt, das Wiederherstellungsdienstprogramm kann jedoch eine Tabellenbereichssicherung in einer alten Protokollfolge nicht erkennen, wenn die Wiederherstellungsoperation für den Tabellenbereich unmittelbar auf eine Wiederherstellungsoperation für die Datenbank folgt. Bis zur tatsächlichen aktualisierenden Wiederherstellung der Datenbank ist die zu verwendende Protokollfolge nicht bekannt. Wenn der Tabellenbereich zu einer alten Protokollfolge gehört, muss er von der Operation zur aktualisierenden Wiederherstellung des Tabellenbereichs „erfasst" werden. Eine Wiederherstellungsoperation, bei der ein ungültiges Sicherungsimage verwendet wird, kann erfolgreich beendet werden, eine anschließende aktualisierende Wiederherstellung dieses Tabellenbereichs wird jedoch fehlschlagen, und der Tabellenbereich befindet sich weiter im Status *Wiederherstellung anstehend*.

Nehmen Sie beispielsweise an, dass eine Sicherungsoperation auf Tabellenbereichsebene, Sicherung 3, zwischen `S0000013.LOG` und `S0000014.LOG` in der oberen Protokollfolge beendet wird (siehe Abb. 8). Wenn Sie unter Verwendung des Sicherungsimages auf Datenbankebene, Sicherung 2, eine Wiederherstellung und eine aktualisierende Wiederherstellung ausführen wollen, müssen Sie die aktualisierende Wiederherstellung bis `S0000012.LOG` ausführen. Anschließend könnten Sie die aktualisierende Wiederherstellung 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 Sicherungsimage auf Tabellenbereichsebene, Sicherung 3, nicht zum Wiederherstellen und aktualisierenden Wiederherstellen verwenden.

Um unter Verwendung des Sicherungsimages auf Tabellenbereichsebene, Sicherung 3, für einen Tabellenbereich eine aktualisierende Wiederherstellung bis zum Protokollende auszuführen, müssen Sie das Sicherungsimage auf Datenbankebene, Sicherung 2, wiederherstellen und anschließend die obere Protokollfolge für eine aktualisierende Wiederherstellung verwenden. Nach der Wiederherstellung des Sicherungsimages auf Tabellenbereichsebene, Sicherung 3, können Sie eine aktualisierende Wiederherstellung bis zum Ende der Protokolle ausführen.

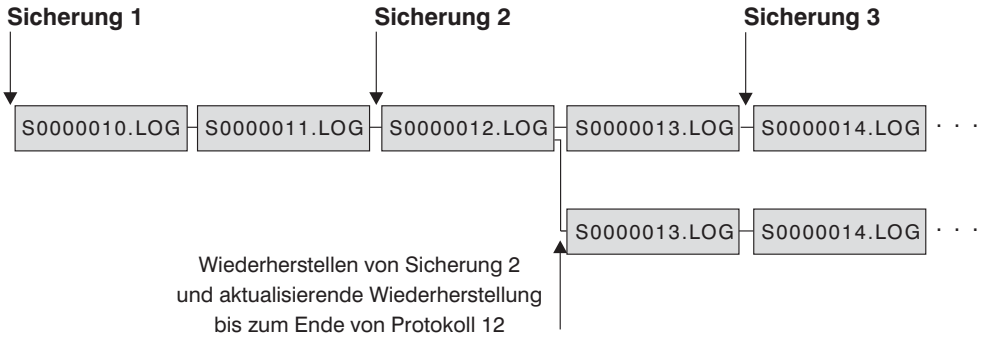


Abbildung 8. Erneutes Verwenden der Namen von Protokolldateien

Zugehörige Referenzen:

- Anhang G, „Benutzerexit zur Datenbankwiederherstellung", auf Seite 369

Zuordnen und Entfernen von Protokolldateien

Die im Datenbankprotokollverzeichnis vorhandenen Protokolldateien werden nicht entfernt, wenn sie eventuell zur Wiederherstellung nach einem Systemabsturz benötigt werden. Wenn Sie jedoch Endlosprotokollierung aktiviert haben, werden die Protokolldateien nach erfolgreicher Archivierung gelöscht. Wenn der Datenbankkonfigurationsparameter *logarchmeth1* nicht auf OFF gesetzt ist, wird eine volle Protokolldatei nur dann entfernt, wenn sie nicht länger zur Wiederherstellung nach einem Systemabsturz benötigt wird. Eine für die Wiederherstellung nach einem Systemabsturz benötigte Protokolldatei wird als aktive Protokolldatei bezeichnet. Eine nicht für die Wiederherstellung nach einem Systemabsturz benötigte Protokolldatei wird als Archivprotokolldatei bezeichnet.

Der Prozess der Zuordnung neuer Protokolldateien und des Entfernens alter Protokolldateien ist von den Einstellungen des Datenbankkonfigurationsparameters *logarchmeth1* abhängig:

logarchmeth1 **und** *logarchmeth2* **sind auf OFF gesetzt**
: Es wird Umlaufprotokollierung verwendet. Bei der Umlaufprotokollierung wird die Wiederherstellung nach einem Systemabsturz unterstützt, die aktualisierende Wiederherstellung 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 DB2® 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 **ist auf LOGRETAIN gesetzt**
: Es wird Archivprotokollierung verwendet. Die Datenbank ist eine wiederherstellbare Datenbank. Sowohl die Wiederherstellung nach einem Systemabsturz als auch die aktualisierende Wiederherstellung sind aktiviert. 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. Jedes Mal, wenn eine Protokolldatei voll ist, schreibt DB2 in eine andere Protokolldatei und erstellt dabei eine neue Protokolldatei, sofern die maximale Anzahl primärer und sekundärer Protokolle noch nicht erreicht ist.

logarchmeth1 **ist auf einen anderen Wert als OFF oder LOGRETAIN gesetzt**
: Es wird Archivprotokollierung verwendet. Die Datenbank ist eine wiederherstellbare Datenbank. Sowohl die Wiederherstellung nach einem Systemabsturz als auch die aktualisierende Wiederherstellung sind aktiviert. Wenn eine Protokolldatei voll ist, wird sie automatisch unter Verwendung des vom Benutzer angegebenen Benutzerexitprogramms archiviert.

  Protokolldateien werden in der Regel 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 Archivprotokolldateien kopiert wurde. DB2 wartet, bis eine neue Protokolldatei benötigt wird und benennt dann die

älteste Archivprotokolldatei um. Eine Protokolldatei, die während der Wiederherstellung in das Datenbankverzeichnis versetzt wurde, wird während des Wiederherstellungsprozesses entfernt, sobald sie nicht mehr benötigt wird. Die alten Protokolldateien bleiben so lange im Datenbankverzeichnis, bis DB2 den Speicherplatz benötigt.

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 DB2 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 gesetzt ist. Wenn NUMARCHRETRY auf 0 gesetzt ist, wird DB2 kontinuierlich versuchen, die Protokolldateien im primären oder sekundären Verzeichnispfad 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 Benutzerexitprogramm 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. Dies kann auftreten, wenn zum Zeitpunkt des Abschaltens 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 Abschalten 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 Abschalten der Datenbank verbleiben die erstellten, sekundären Protokolldateien bei einem erneuten Start der Datenbank im Pfad für aktive Protokolldateien.

Verwalten von Protokolldateien durch Protokollarchivierung

Folgende Hinweise und Informationen sind für das Aufrufen eines Benutzerexitprogramms zum Archivieren und Abrufen von Protokolldateien zu beachten:

- Wenn der Datenbankkonfigurationsparameter *logarchmeth1* auf `USEREXIT` gesetzt wird, ruft der Datenbankmanager während der aktualisierenden Wiederherstellung von Datenbanken ein Benutzerexitprogramm zum Archivieren von Dateien oder zum Abrufen von Protokolldateien auf. Es erfolgt eine Anforderung zum

Abrufen einer Protokolldatei, wenn das Dienstprogramm zur aktualisierenden Wiederherstellung eine Protokolldatei benötigt, die nicht im Verzeichnis für den Protokollpfad zu finden ist.

Anmerkung: Unter Windows® können Sie keinen REXX-Benutzerexit zum Archivieren von Protokollen verwenden.

- Beim Archivieren wird eine Protokolldatei an den Benutzerexit 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 Benutzerexit übergebene Protokolldatei wird so lange im Verzeichnis für den Protokollpfad behalten, bis sie für die normale Verarbeitung nicht mehr benötigt wird. Dann wird der Plattenspeicherplatz wiederverwendet.
- DB2® öffnet eine Datei im Lesemodus, wenn DB2 ein Benutzerexitprogramm zum Archivieren einer Protokolldatei startet. Auf einigen Plattformen nimmt dies dem Benutzerexitprogramm die Fähigkeit, die Protokolldatei zu löschen. Andere Plattformen wie AIX® ermöglichen es Prozessen, einschließlich des Benutzerexitprogramms, Protokolldateien zu löschen. Ein Benutzerexitprogramm sollte eine Protokolldatei nie nach ihrer Archivierung löschen können, da die Datei weiterhin aktiv sein könnte und für die Wiederherstellung nach einem Systemabsturz benötigt werden kann. DB2 verwaltet bei der Archivierung von Protokolldateien die erneute Verwendung von Plattenspeicherplatz.
- Wenn eine Protokolldatei archiviert wurde und keine offenen Transaktionen enthält, löscht DB2 die Datei nicht, sondern benennt sie in den Namen der nächsten Protokolldatei um, wenn eine solche Datei benötigt wird. Dies führt zu einem Leistungsvorteil, da bei der Erstellung einer neuen Protokolldatei (statt der Umbenennung) alle Seiten neu geschrieben werden müssen, um den Plattenspeicherplatz zu reservieren. Es ist effizienter, die erforderlichen Seiten auf der Platte wiederzuverwenden, als sie freizugeben und neu zuzuordnen.
- DB2 ruft während einer Wiederherstellung nach einem Systemabsturz *nur dann* Protokolldateien ab, wenn der Datenbankkonfigurationsparameter *logsecond* auf -1 gesetzt ist.
- Das Konfigurieren der Protokollarchivierung garantiert nicht die aktualisierende Wiederherstellung bis zum Fehlerpunkt, sondern ist lediglich ein Versuch, das Fehlerfenster zu verkleinern. Wenn die Protokolldateien voll werden, werden sie für die Benutzerexitroutine in eine Warteschlange eingereiht. 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.
- 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 verlorengehen, wenn ein Fehler auf der Platte auftritt. Bei einer Datenbankkonfiguration mit kleinen Protokolldateien werden häufiger Daten an die Benutzerexitroutine übergeben.

  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 *logarchmeth1* auf USEREXIT gesetzt ist, erfolgt jedes Mal, wenn eine aktive Protokolldatei voll wird, eine Anforderung zum Archivieren an das Benutzerexitprogramm. Es ist möglich, dass eine aktive Protokolldatei nicht voll ist, wenn

die letzte Verbindung von der Datenbank getrennt wird, und dass das Benutzerexitprogramm auch für eine teilweise gefüllte aktive Protokolldatei aufgerufen wird.

Anmerkung: Zur Freigabe nicht genutzten Protokollspeichers wird die Protokolldatei abgeschnitten, bevor sie archiviert wird.

- Es sollte eine Kopie des Protokolls auf einer anderen physischen Einheit erstellt werden, so dass die offline gespeicherte Protokolldatei zur aktualisierenden Wiederherstellung verwendet werden kann, wenn auf der Einheit, auf der die Protokolldatei gespeichert ist, ein Datenträgerfehler auftritt. Bei dieser Einheit sollte es sich nicht um diejenige handeln, auf der die Datendateien für die Datenbank gespeichert werden.
- Wenn Sie Benutzerexitprogramme aktiviert haben und als Speichereinheit für Protokolle und Sicherungsimages ein Bandlaufwerk verwenden, müssen Sie sicherstellen, dass das Ziel für die Sicherungsimages und die Archivprotokolldateien nicht identisch ist. Da während einer Sicherungsoperation Protokollarchivierungsaktivitäten stattfinden können, würde es zu einem Fehler kommen, wenn beide Prozesse versuchen, gleichzeitig auf dasselbe Bandlaufwerk zu schreiben.
- Lokal angeschlossene Bandlaufwerke sollten nicht zur Verwaltung von Protokolldateien mit einem Benutzerexitprogramm verwendet werden, wenn Sie eine der folgenden Funktionen verwenden:
  - Endlosprotokollierung
  - Onlinewiederherstellung auf Tabellenbereichsebene
  - Replikation
  - Die API zum asynchronen Lesen von Protokolldaten (db2ReadLog)
  - HADR (High Availability Disaster Recovery)
- Wenn die Datenbank geschlossen wird, bevor eine positive Antwort für eine Archivierungsanforderung von einem Benutzerexitprogramm empfangen wurde, sendet der Datenbankmanager in einigen Fällen beim Öffnen der Datenbank eine weitere Anforderung. Auf diese Weise kann es vorkommen, dass eine Protokolldatei mehrmals archiviert wird.
- Wenn ein Benutzerexitprogramm eine Anforderung zum Archivieren einer nicht existierenden Datei empfängt (weil mehrere Anforderungen zum Archivieren ausgeführt wurden und die Datei nach der ersten erfolgreichen Archivierung gelöscht wurde) oder zum Abrufen einer nicht existierenden Datei (weil sie sich in einem anderen Verzeichnis befindet oder das Ende der Protokolle erreicht wurde), sollte es diese Anforderung ignorieren und einen erfolgreichen Rückkehrcode übergeben.
- Das Benutzerexitprogramm sollte das Vorhandensein verschiedener gleichnamiger Protokolldateien nach einer Wiederherstellung bis zu einem bestimmten Zeitpunkt zulassen. Es sollte so konzipiert sein, dass beide Protokolldateien beibehalten werden und diese Protokolldateien dem korrekten Wiederherstellungspfad zugeordnet werden.
- Wenn ein Benutzerexitprogramm für mindestens zwei Datenbanken aktiviert ist, die zur Archivierung von Protokolldateien dieselbe Bandeinheit verwenden, und es wird für eine der Datenbanken eine aktualisierende Wiederherstellung vorgenommen, sollte(n) die andere(n) Datenbank(en) nicht aktiv sein. Wenn eine der anderen Datenbanken während der Operation zur aktualisierenden Wiederherstellung versucht, eine Protokolldatei zu archivieren, ist es möglich, dass die zur aktualisierenden Wiederherstellung erforderlichen Protokolle nicht gefunden werden oder dass die neue, auf der Bandeinheit archivierte Protokolldatei die zuvor auf dieser Bandeinheit gespeicherten Protokolldateien überschreibt.

Damit keine der beiden Situationen auftritt, müssen Sie entweder sicherstellen, dass keine anderen Datenbanken auf dem Knoten, der das Benutzerexitprogramm aufruft, während der aktualisierenden Wiederherstellung geöffnet sind, oder Sie müssen ein Benutzerexitprogramm zur Behandlung dieser Situation schreiben.

Zugehörige Konzepte:

- „Verwalten von Protokolldateien" auf Seite 51

Blockieren von Transaktionen bei voller Protokollverzeichnisdatei

Der Datenbankkonfigurationsparameter *blk_log_dsk_ful* kann so eingestellt werden, dass keine Fehlermeldungen wegen vollen Datenträgers generiert werden, wenn DB2® im aktiven Protokollpfad keine neue Protokolldatei erstellen kann.

Stattdessen versucht DB2 alle fünf Minuten erneut, die Protokolldatei zu erstellen, bis dies erfolgreich ist. Wenn eine Protokollarchivierungsmethode angegeben ist, überprüft DB2 außerdem den Beendigungsstatus der Protokolldateiarchivierung. Wenn eine Archivprotokolldatei erfolgreich archiviert wurde, kann DB2 der inaktiven Protokolldatei den neuen Protokolldateinamen zuweisen und fortfahren. Nach jedem Versuch schreibt DB2 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 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 blockiert sind, oder auf eine Datenseite, die von der aktualisierenden Anwendung im Pufferpool fixiert wurde, erscheinen Abfragen mit Lesezugriff ebenfalls blockiert.

Zugehörige Konzepte:

- „Wiederherstellungsprotokolle" auf Seite 37
- „Verwalten von Protokolldateien durch Protokollarchivierung" auf Seite 54

Bedarfsgesteuerte Protokollarchivierung

DB2® unterstützt das Schließen (und, wenn aktiviert, auch das Archivieren) der aktiven Protokolldatei einer wiederherstellbaren Datenbank zu einem 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.

Zugehörige Konzepte:

- „Verwalten von Protokolldateien durch Protokollarchivierung" auf Seite 54

Zugehörige Referenzen:

- „ARCHIVE LOG" auf Seite 291
- „Konfigurationsparameter für die Datenbankprotokollierung" auf Seite 42
- „db2ArchiveLog - Aktive Protokolldatei archivieren" auf Seite 301

Verwenden von Protokollen auf unformatierten Einheiten

Sie können für das Datenbankprotokoll eine unformatierte Einheit verwenden. Dies hat Vor- und Nachteile.

- Es gibt die folgenden Vorteile:
  - Sie können mehr als 26 physische Laufwerke an das System anschließen.
  - Die Länge des Datei-E/A-Pfads ist kürzer. Dies kann zu einer verbesserten Systemleistung führen. Sie sollten Vergleichstests durchführen, um festzustellen, ob es messbare Vorteile für Ihre Systemauslastung gibt.
- Es gibt die folgenden Nachteile:
  - Eine Einheit kann nicht von anderen Anwendungen mitverwendet werden. Die gesamte Einheit *muss* DB2 zugeordnet werden.
  - Die Einheit kann von keinen Dienstprogramm des Betriebssystems oder einem Tool anderer Hersteller verwendet werden, die auf die Einheit sichern oder von der Einheit kopieren würden.
  - Sie können leicht das Dateisystem auf einem vorhandenen Laufwerk löschen, wenn Sie die falsche Nummer für das physische Laufwerk angeben.

Sie können ein Protokoll auf einer unformatierten Einheit mit dem Datenbankkonfigurationsparameter *newlogpath* konfigurieren. Vorher sollten Sie die oben aufgeführten Vor- und Nachteile abwägen. Außerdem sollten Sie die folgenden Aspekte berücksichtigen:

- Nur eine Einheit ist zulässig. Sie können die Einheit über mehrere Platten hinweg auf Betriebssystemebene definieren. DB2 führt einen Betriebssystemaufruf aus, um die Größe der Einheit in 4-KB-Seiten zu ermitteln. Wenn Sie mehrere Platten verwenden, wird dadurch eine größere Einheit bereitgestellt, und das resultierende plattenübergreifende Lesen und Schreiben von Daten kann aufgrund des schnelleren E/A-Durchsatzes zu einer Leistungssteigerung führen.
- DB2 versucht, auf die letzte 4-KB-Seite der Einheit zu schreiben. Wenn die Einheit größer als 2 GB ist, schlägt der Versuch, auf die letzte Seite zu schreiben, bei Betriebssystemen fehl, die keine Unterstützung für Einheiten bereitstellen, die größer als 2 GB sind. In diesem Fall versucht DB2, alle Seiten bis zur unterstützten Begrenzung zu verwenden. Mit Hilfe der Informationen zur Größe der Einheit wird die Einheitengröße (in 4-KB-Seiten) angegeben, die für DB2 bei Unterstützung durch das Betriebssystem zur Verfügung steht. Die Menge des Plattenspeicherplatzes, auf den DB2 schreiben kann, wird als *verfügbare Einheitengröße* bezeichnet.

  Die erste 4-KB-Seite der Einheit wird von DB2 nicht verwendet (dieser Speicherbereich wird in der Regel vom Betriebssystem für andere Zwecke verwendet). Dies bedeutet, dass der für DB2 verfügbare Gesamtspeicherbereich *Einheitengröße = verfügbare Einheitengröße - 1* ist.
- Der Parameter *logsecond* wird nicht verwendet. DB2 ordnet keine sekundären Protokolle zu. Die Größe des Speicherbereichs für aktive Protokolle ist die Anzahl von 4-KB-Seiten, die sich aus *logprimary* x *logfilsiz* ergibt.
- Protokollsätze werden weiterhin in Protokollspeicherbereichen gruppiert, wobei die Protokolldateigröße (*logfilsiz*) für jeden in 4-KB-Seiten festgelegt ist.

Protokollspeicherbereiche werden nacheinander in die unformatierte Einheit platziert. Jeder Speicherbereich besteht zudem aus zwei zusätzlichen Seiten für den Speicherbereichsheader. Dies bedeutet, dass sich die *Anzahl verfügbarer Protokollspeicherbereiche*, die die Einheit unterstützen kann, aus *Einheitengröße* / (*logfilsiz* + 2) berechnen lässt.

- Die Einheit muss groß genug sein, um den Speicherbereich für aktive Protokolldateien unterstützen zu können. Das heißt, dass die *Anzahl verfügbarer Protokollspeicherbereiche* größer-gleich dem Wert sein muss, der für den Konfigurationsparameter *logprimary* angegeben ist. Wenn die Protokollarchivierung über die Konfigurationsparameter *logarchmeth1* und *logarchmeth2* archiviert ist, stellen Sie sicher, dass die unformatierte Einheit mehr Protokolle enthalten kann, als mit dem Wert des Konfigurationsparameters *logprimary* angegeben. Dadurch wird die Verzögerung kompensiert, die beim Archivieren einer Protokolldatei auftritt.
- Bei Verwendung der Umlaufprotokollierung legt der Konfigurationsparameter *logprimary* die Anzahl von Protokollspeicherbereichen fest, die auf die Einheit geschrieben werden. Dies kann zu freiem Speicherplatz auf der Einheit führen.
- Wenn *logarchmeth1* auf `LOGRETAIN` gesetzt wird, nachdem die *Anzahl verfügbarer Protokollspeicherbereiche* erschöpft ist, empfangen alle Operationen, die zu einer Aktualisierung führen, die Fehlernachricht, dass das Protokoll voll ist. Zu diesem Zeitpunkt müssen Sie die Datenbank schließen und eine Offlinesicherung erstellen, um die Wiederherstellbarkeit sicherzustellen. Nach der Datenbanksicherung gehen die auf die Einheit geschriebenen Protokollsätze verloren. Dies bedeutet, dass Sie eine frühere Datenbanksicherung nicht zur Wiederherstellung der Datenbank gefolgt von einer aktualisierenden Wiederherstellung verwenden können. Wenn Sie eine Datenbanksicherung vor der Ausschöpfung der *Anzahl verfügbarer Protokollspeicherbereiche* erstellen, können Sie die Datenbank wiederherstellen und aktualisierend wiederherstellen.
- Wenn *logarchmeth1* auf `USEREXIT` gesetzt ist, muss das Benutzerexitprogramm die Einheit lesen und das archivierte Protokoll als Datei speichern können.
- DB2 ruft keine Protokolldateien von einer unformatierten Einheit ab. Wenn eine Protokolldatei benötigt wird, liest DB2 stattdessen den Speicherbereichsheader, um zu ermitteln, ob die unformatierte Einheit die zu verwendende Protokolldatei enthält. Wenn die erforderliche Protokolldatei nicht in der unformatierten Einheit gefunden wird, sucht DB2 im Überlaufprotokollpfad. Wenn die Protokolldatei auch dort nicht gefunden wird, ruft DB2 die Protokolldatei ab und stellt sie in den Überlaufprotokollpfad. Wurde kein Überlaufprotokollpfad angegeben, wird DB2 nicht versuchen, die Protokolldatei abzurufen.
- Wenn Sie für die Protokollierung eine unformatierte Einheit konfiguriert haben und DataPropagator™ (DPROP) verwenden bzw. eine andere Anwendung, die die Anwendungsprogrammierschnittstelle **db2ReadLog** aufruft, muss der Konfigurationsparameter *overflowlogpath* konfiguriert werden. DB2 kann ein Benutzerexitprogramm aufrufen, um die Protokolldatei abzurufen und die von der Anwendungsprogrammierschnittstelle **db2ReadLog** angeforderten Protokolldateien zurückzugeben. Die abgerufene Protokolldatei wird in den Pfad gestellt, der vom Datenbankkonfigurationsparameter *overflowlogpath* angegeben ist.

Zugehörige Tasks:

- „Angeben eines direkten E/A-Zugriffs (Raw I/O)" im Handbuch *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „db2ReadLog - Protokolldaten asynchron lesen" auf Seite 323
- Anhang F, „Tivoli Storage Manager", auf Seite 365

Einschließen von Protokolldateien in ein Sicherungsimage

Wenn Sie eine Onlinesicherungsoperation ausführen, können Sie angeben, ob die zum Wiederherstellen und aktualisierenden Wiederherstellen erforderlichen Protokolldateien in das Sicherungsimage aufgenommen werden. Dies bedeutet, dass Sie die Protokolldateien nicht separat senden oder selbst packen müssen, wenn Sie Sicherungsimages an einen Standort senden, an dem eine Wiederherstellung nach einem Katastrophenfall ausgeführt werden muss. Außerdem müssen sie so nicht selbst entscheiden, welche Protokolldateien erforderlich sind, um die Konsistenz einer Onlinesicherung zu gewährleisten. Dies bietet einen gewissen Schutz vor dem Löschen von Protokolldateien, die für eine erfolgreiche Wiederherstellung erforderlich sind.

Wenn Sie diese Funktion verwenden wollen, geben Sie die Option INCLUDE LOGS des Befehls BACKUP DATABASE an. Wenn Sie diese Option angeben, schneidet das Sicherungsdienstprogramm die aktive Protokolldatei ab und kopiert die erforderlichen Protokollspeicherbereiche in das Sicherungsimage.

Verwenden Sie zum Wiederherstellen der Protokolldateien aus einem Sicherungsimage die Option LOGTARGET des Befehls RESTORE DATABASE, und geben Sie einen vollständig qualifizierten, auf dem DB2®-Server vorhandenen Pfad an. Das Dienstprogramm zur Datenbankwiederherstellung schreibt anschließend die Protokolldateien aus dem Image in den Zielpfad. Wenn eine Protokolldatei desselben Namens bereits im Zielpfad vorhanden ist, schlägt die Wiederherstellungsoperation fehl, und es wird ein Fehler zurückgegeben. Wird die Option LOGTARGET nicht angegeben, werden keine Protokolldateien aus dem Sicherungsimage wiederhergestellt.

Wird die Option LOGTARGET angegeben und das Sicherungsimage enthält keine Protokolldateien, wird ein Fehler zurückgegeben, bevor versucht wird, Tabellenbereichsdaten wiederherzustellen. Die Wiederherstellungsoperation schlägt ebenfalls fehl, wenn ein ungültiger oder schreibgeschützter Pfad angegeben wird. Wenn während der Wiederherstellung einer Datenbank oder eines Tabellenbereichs, für die bzw. den die Option LOGTARGET angegeben wurde, mindestens eine Protokolldatei nicht extrahiert werden kann, schlägt die Wiederherstellungsoperation fehl, und es wird ein Fehler zurückgegeben.

Sie können auch nur die in einem Sicherungsimage gespeicherten Protokolldateien wiederherstellen. Geben Sie dazu die Option LOGS mit der Option LOGTARGET des Befehls RESTORE DATABASE an. Wenn bei der Wiederherstellung von Protokolldateien in diesem Modus Fehler auftreten, schlägt die Wiederherstellungsoperation fehl, und es wird ein Fehler zurückgegeben.

Bei einer automatischen Teilwiederherstellungsoperation werden nur die Protokolle aus dem Sicherungsimage abgerufen, die sich im Zielimage der Wiederherstellungsoperation befinden. Protokolle, die sich in temporären Images befinden, auf die während des Teilwiederherstellungsprozesses verwiesen wird, werden nicht aus diesen Sicherungsimages extrahiert. Wenn Sie bei der manuellen Teilwiederherstellung eines Sicherungsimages, das Protokolldateien enthält, ein Protokollzielverzeichnis angeben, werden die in diesem Sicherungsimage enthaltenen Protokolldateien wiederhergestellt.

Anmerkungen:

1. Diese Funktion steht nur für Datenbanken mit einer einzelnen Partition zur Verfügung.
2. Diese Funktion wird für Offlinesicherungen nicht unterstützt.
3. Wenn in ein Onlinesicherungsimage Protokolldateien eingeschlossen werden, kann das daraus resultierende Image nicht in Releases von DB2 Universal Database™ (UDB) wiederhergestellt werden, die älter als Version 8.2 sind.

Zugehörige Tasks:

- „Wiederherstellen von Teilsicherungsimages" auf Seite 32

Zugehörige Referenzen:

- „BACKUP DATABASE" auf Seite 79
- „RESTORE DATABASE" auf Seite 102

Verhindern von Protokolldateiverlusten

In Situationen, in denen Sie eine Datenbank löschen oder eine aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt durchführen müssen, verlieren Sie möglicherweise Protokolldateien, die für zukünftige Wiederherstellungsoperationen erforderlich sind. In solchen Fällen ist es wichtig, Kopien aller im aktuellen Protokollverzeichnispfad der Datenbank vorhandenen Protokolle zu erstellen. Betrachten Sie die folgenden Szenarios:

- Wenn Sie vor einer Wiederherstellungsoperation eine Datenbank löschen wollen, müssen Sie die im Pfad für aktive Protokolldateien vorhandenen Protokolldateien speichern, bevor Sie den Befehl DROP DATABASE absetzen. Nach der Wiederherstellung der Datenbank werden diese Protokolldateien möglicherweise für die aktualisierende Wiederherstellung 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 auf einer Partition löschen müssen, indem Sie die Option AT NODE mit dem Befehl DROP DATABASE angeben), da sie beschädigt wurde, so dass der Befehl RESTORE fehlschlägt. Sie können eine Datenbank auch löschen, um vor der Wiederherstellungsoperation 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 Wiederherstellungsoperation bis zu einem bestimmten Zeitpunkt und der erneuten Verbindungsherstellung zur Datenbank feststellen, dass Sie die Datenbank eigentlich bist zu einem späteren Zeitpunkt hätten wiederherstellen müssen, können Sie dies nicht mehr nachholen, da die Protokolle überschrieben wurden. Zwar ist es möglich, dass die ursprünglichen Protokolldateien archiviert wurden, DB2® könnte jedoch ein Benutzerexitprogramm aufrufen, um die neu generierten Protokolldateien zu archivieren. Je nach dem, wie das Benutzerexitprogramm geschrieben ist, könnten hierbei die ursprünglichen Protokolldateien im Verzeichnis der Archivprotokolldateien überschrieben werden. Auch wenn sich die ursprünglichen Protokolldateien zusammen mit den neuen Protokolldateien im Verzeichnis der Archivprotokolldateien befinden (als verschiedene Versionen derselben Dateien), müssen Sie bestimmen, welche Protokolle nun für spätere Wiederherstellungsoperationen verwendet werden sollen.

Zugehörige Konzepte:

- „Wiederherstellungsprotokolle" auf Seite 37

Datei des Wiederherstellungsprotokolls

Mit jeder Datenbank wird eine Datei des Wiederherstellungsprotokolls (Recovery History File) erstellt. Diese Datei wird automatisch aktualisiert, wenn eine der folgenden Aktionen stattfindet:

- Sichern einer Datenbank oder von Tabellenbereichen
- Wiederherstellen einer Datenbank oder von Tabellenbereichen
- Aktualisierendes Wiederherstellen einer Datenbank oder von Tabellenbereichen
- Erstellen eines Tabellenbereichs
- Ändern eines Tabellenbereichs
- Versetzen eines Tabellenbereichs in den Wartemodus
- Umbenennen eines Tabellenbereichs
- Löschen eines Tabellenbereichs
- Laden einer Tabelle
- Löschen einer Tabelle (wenn das Wiederherstellen gelöschter Tabellen aktiviert ist)
- Reorganisieren einer Tabelle
- Aufrufen der bedarfsgesteuerten Archivierung von Protokolldateien
- Schreiben in eine neue Protokolldatei (wenn die Protokollierung für Wiederherstellung verwendet wird)
- Archivieren einer Protokolldatei (wenn die Protokollierung für Wiederherstellung verwendet wird)
- Wiederherstellen einer Datenbank

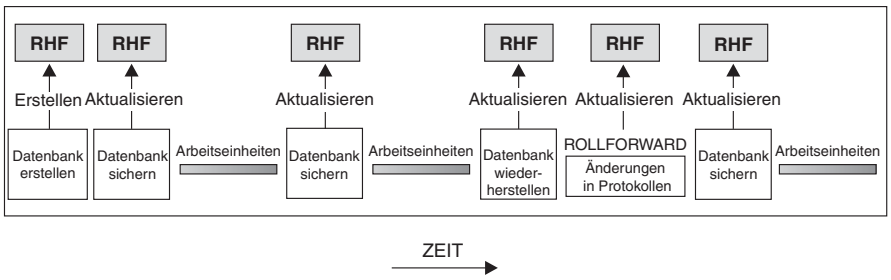


Abbildung 9. Erstellen und Aktualisieren der Datei des Wiederherstellungsprotokolls

Sie können die zusammengefassten Sicherungsinformationen in dieser Datei zur Wiederherstellung 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 der letzten Wiederherstellung
- Den Zeitpunkt, zu dem ein Tabellenbereich umbenannt wurde, sowie den vorherigen und den aktuellen Name des Tabellenbereichs
- Den Status der Sicherung: aktiv, inaktiv, abgelaufen oder gelöscht

- Die letzte Protokollfolgenummer, die von der Datenbanksicherung gespeichert oder von einer aktualisierenden Wiederherstellung verarbeitet wurde

Verwenden Sie den Befehl LIST HISTORY, wenn Sie die Einträge in der Datei des Wiederherstellungsprotokolls anzeigen wollen.

Jede Sicherungsoperation (sowohl für einen Tabellenbereich als auch für die gesamte Datenbank bzw. für eine Teilsicherung) schließt eine Kopie der Datei des Wiederherstellungsprotokolls mit ein. Die Datei des Wiederherstellungsprotokolls ist der Datenbank zugeordnet. Beim Löschen einer Datenbank wird auch die Datei des Wiederherstellungsprotokolls gelöscht. Beim Wiederherstellen einer Datenbank an einer neuen Position wird auch die Datei des Wiederherstellungsprotokolls wiederhergestellt. Beim Wiederherstellen wird die vorhandene Datei des Wiederherstellungsprotokolls nur dann überschrieben, wenn die auf dem Datenträger vorhandene Datei keine Einträge enthält. Ist dies der Fall, wird das Datenbankprotokoll aus dem Sicherungsimage wiederhergestellt.

Wenn die aktuelle Datenbank unbrauchbar oder nicht verfügbar ist und die zugehörige Datei des Wiederherstellungsprotokolls beschädigt oder gelöscht wurde, steht eine Option des Befehls RESTORE zur Verfügung, mit der lediglich die Datei des Wiederherstellungsprotokolls wiederhergestellt werden kann. Im Anschluss daran kann anhand der Informationen in der Datei des Wiederherstellungsprotokolls festgestellt werden, welche Sicherung zur Wiederherstellung 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 zur aktuellen vollständigen Datenbanksicherung und der zugehörigen Wiederherstellungsgruppe beibehalten. (Diese Kopie kann nur über den Befehl PRUNE 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.

Zugehörige Referenzen:

- „rec_his_retentn - Aufbewahrungszeitraum für Wiederherstellungsprotokoll" im Handbuch *Systemverwaltung: Optimierung*
- „LIST HISTORY" auf Seite 294

Datei des Wiederherstellungsprotokolls - Garbage Collection

Garbage Collection

Auch wenn Sie den Befehl PRUNE HISTORY zu einem beliebigen Zeitpunkt verwenden können, um Einträge aus der Protokolldatei zu entfernen, wird empfohlen, diese Aktion von DB2® ausführen zu lassen. Die Anzahl der DB2-Datenbanksicherungskopien, die in der Datei des Wiederherstellungsprotokolls dokumentiert sind, wird von *DB2 Garbage Collection* automatisch überwacht. DB2 Garbage Collection wird in folgenden Fällen aufgerufen:

- Nach der erfolgreichen Beendigung einer Operation zur Gesamtsicherung der Datenbank.

- Nach der erfolgreichen Beendigung einer Operation zur Datenbankwiederherstellung, bei der keine aktualisierende Wiederherstellung erforderlich ist.
- Nach der erfolgreichen Beendigung einer Operation zur aktualisierenden Wiederherstellung einer Datenbank.

Mit dem Konfigurationsparameter *num_db_backups* wird definiert, wie viele Images von aktiven Gesamtsicherungen (keine Teilsicherungen) aufbewahrt werden. Der Wert dieses Parameters wird verwendet, um die Protokolldatei ab dem letzten Eintrag zu durchsuchen.

Nach jeder Datenbankgesamtsicherung (keine Teilsicherung) wird der Konfigurationsparameter *rec_his_retentn* verwendet, um die abgelaufenen Einträge aus der Protokolldatei zu löschen.

Eine *aktive Datenbanksicherung* ist eine Sicherung, die wiederhergestellt und mit aktuellen Protokollen aktualisierend wiederhergestellt werden kann, um wieder den aktuellen Status der Datenbank zu erhalten. Eine *inaktive Datenbanksicherung* ist eine Sicherung, bei deren Wiederherstellung die Datenbank in einen früheren Zustand zurückversetzt wird.

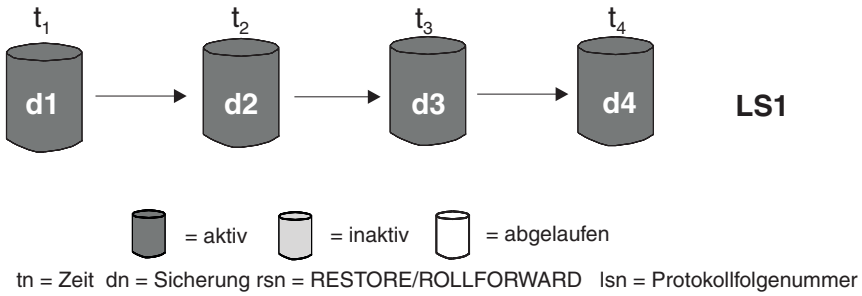


Abbildung 10. Aktive Datenbanksicherungen. Der Wert von *num_db_backups* wurde auf vier gesetzt.

Alle aktiven Datenbanksicherungen, die nicht mehr benötigt werden, sind als „abgelaufen" markiert. Diese Images werden als unnötig betrachtet, da neuere Sicherungsimages zur Verfügung stehen. Alle Sicherungsimages von Tabellenbereichen und alle Ladesicherungskopien, die vor der abgelaufenen Datenbanksicherung erstellt worden sind, werden ebenfalls als „abgelaufen" markiert.

Alle Datenbanksicherungen, die als „inaktiv" markiert sind und vor dem Zeitpunkt der abgelaufenen Datenbanksicherung gespeichert wurden, werden ebenfalls als „abgelaufen" markiert. Alle zugeordneten „inaktiven" Sicherungsimages von Tabellenbereichen und Ladesicherungskopien werden ebenfalls als „abgelaufen" markiert.

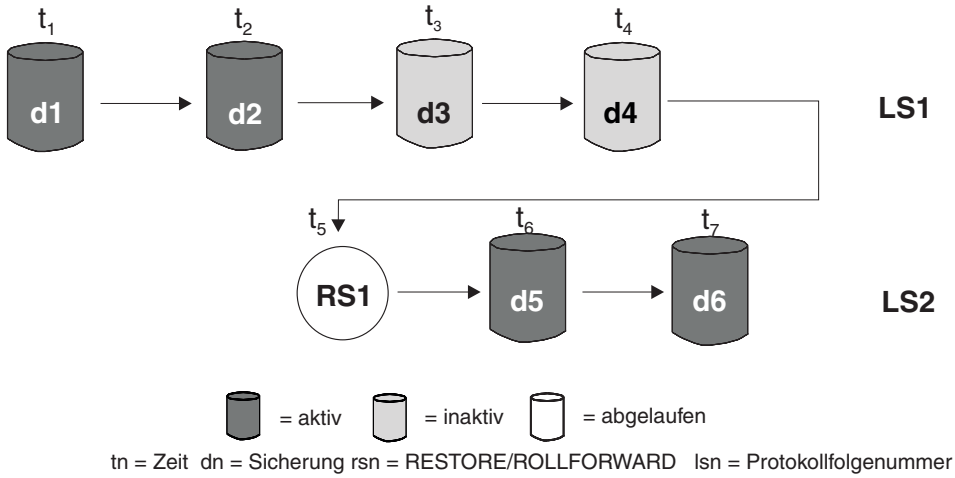


Abbildung 11. Inaktive Datenbanksicherungen

Wenn ein aktives Datenbanksicherungsimage wiederhergestellt wird, es aber nicht die aktuelle in der Protokolldatei aufgezeichnete Datenbanksicherung ist, werden alle nachfolgenden Datenbanksicherungsimages, die zur selben Protokollfolge gehören, als „inaktiv" markiert.

Wenn ein inaktives Datenbanksicherungsimage wiederhergestellt wird, werden alle inaktiven Datenbanksicherungen, die zur aktuellen Protokollfolge gehören, erneut als „aktiv" markiert. Alle aktiven Datenbanksicherungsimages, die nicht länger zur aktuellen Protokollfolge gehören, werden als „inaktiv" markiert.

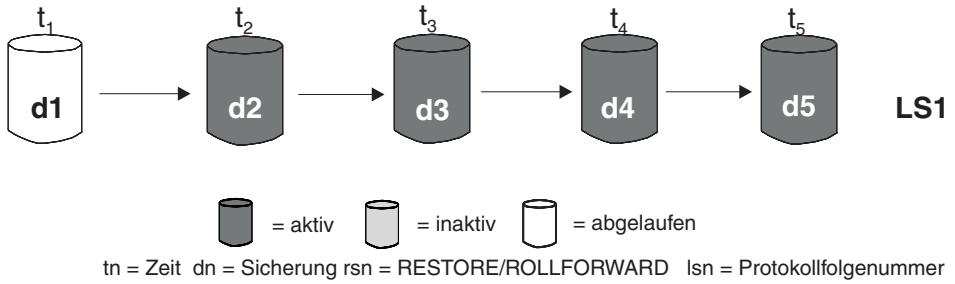


Abbildung 12. Abgelaufene Datenbanksicherungen

Infolge von DB2 Garbage Collection werden auch die Protokolldateieinträge für das Sicherungsimage einer DB2-Datenbank oder eines DB2-Tabellenbereichs als „inaktiv" markiert, wenn diese Sicherung nicht der aktuellen *Protokollfolge* entspricht, die auch als aktuelle *Protokollkette* bezeichnet wird. Die aktuelle Protokollfolge wird durch das DB2-Datenbanksicherungsimage, das wiederhergestellt wurde, und durch die verarbeiteten Protokolldateien bestimmt. Nach der Wiederherstellung eines Datenbanksicherungsimages werden alle zu einem späteren Zeitpunkt erstellten Datenbanksicherungsimages als „inaktiv" markiert, da mit dem wiederhergestellten Image eine neue Protokollkette beginnt. (Dies trifft zu, wenn das Sicherungsimage ohne aktualisierende Wiederherstellung wiederhergestellt wurde. Wenn eine aktualisierende Wiederherstellung ausgeführt wurde, werden alle nach der Unterbrechung der Protokollkette erstellten Datenbanksicherungen als „inaktiv" markiert. Es ist vorstellbar, das ein älteres Datenbanksicherungsimage wiederhergestellt werden muss, da das Dienstprogramm zur aktualisierenden Wiederherstellung die Protokollfolge verwendet hat, die ein aktuelles beschädigtes Sicherungsimage enthält.)

Das Sicherungsimage eines Tabellenbereichs wird als „inaktiv" markiert, wenn nach seiner Wiederherstellung der aktuelle Status der Datenbank nicht durch Anwenden der aktuellen Protokollfolge erreicht werden kann.

Wenn ein Sicherungsimage DATALINK-Spalten enthält, werden alle Data Links-Server, auf denen DB2 Data Links Manager ausgeführt wird, dazu aufgefordert, Garbage Collection anzufordern. DB2 Garbage Collection löscht daraufhin Sicherungen der zugehörigen Data Links-Serverdateien, die in der abgelaufenen Sicherung vorhanden waren, zu denen jedoch vor der nächsten Datenbanksicherung die Verbindung aufgehoben wurde.

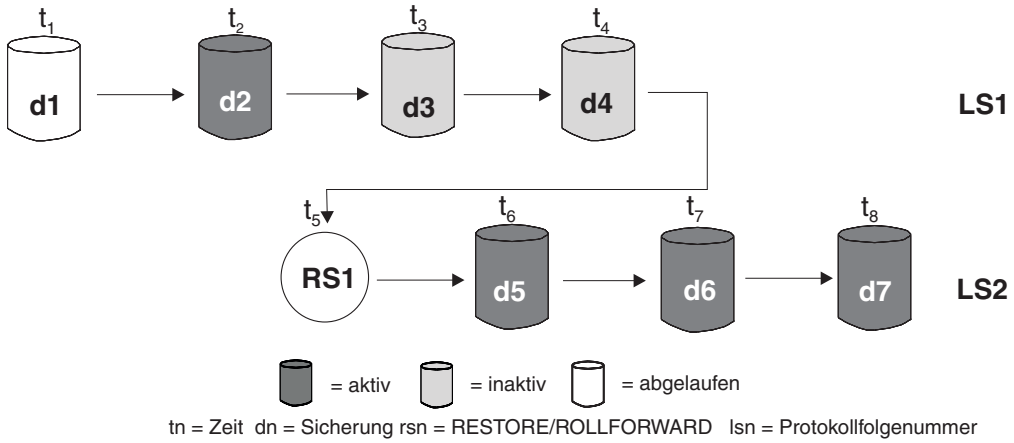


Abbildung 13. Gemischte aktive, inaktive und abgelaufene Datenbanksicherungen

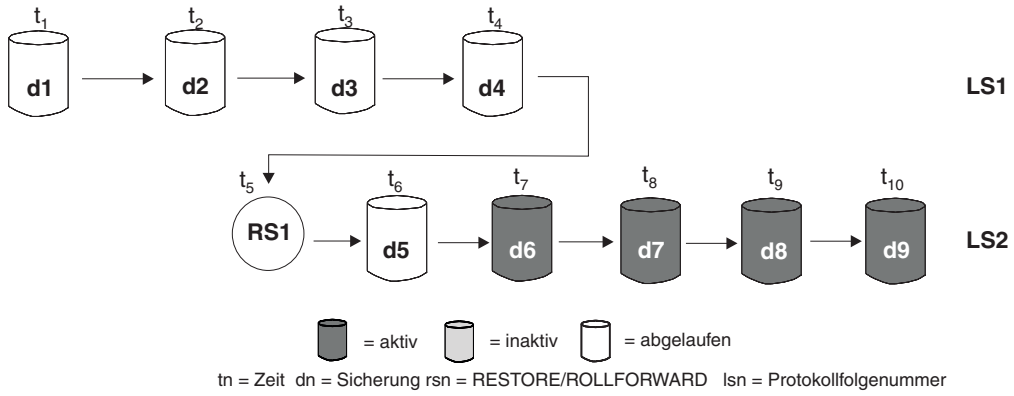


Abbildung 14. Abgelaufene Protokollfolge

Zugehörige Konzepte:

- „Datei des Wiederherstellungsprotokolls" auf Seite 62

Zugehörige Referenzen:

- „PRUNE HISTORY/LOGFILE" auf Seite 296

Tabellenbereichsstatus

Der aktuelle Zustand eines Tabellenbereichs wird von seinem *Status* wiedergegeben. Zu den häufigsten bei der Wiederherstellung auftretenden Status gehören folgende:

- *Sicherung anstehend*. Ein Tabellenbereich wird in diesen Status versetzt, wenn eine aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt ausgeführt wurde, oder im Anschluss an eine LOAD-Operation, die mit der Option NO COPY ausgeführt wurde. Der Tabellenbereich muss gesichert werden, bevor er verwendet werden kann. (Wird keine Sicherung erstellt, kann der Tabellenbereich nicht aktualisiert werden, es sind jedoch Operationen mit Lesezugriff möglich.)
- *Wiederherstellung anstehend*. Ein Tabellenbereich wird in diesen Status versetzt, wenn eine für den Tabellenbereich ausgeführte Operation zur aktualisierenden Wiederherstellung abgebrochen wird oder einen Fehler feststellt, der auf Nichtwiederherstellbarkeit hinweist. Ist Letzeres der Fall, muss der Tabellenbereich wiederhergestellt und anschließend erneut aktualisierend wiederhergestellt werden.
- *Aktualisierende Wiederherstellung läuft*. Ein Tabellenbereich wird in diesen Status versetzt, während für den Tabellenbereich eine Operation zur aktualisierenden Wiederherstellung ausgeführt wird. Nach Beendigung dieser Operation zur aktualisierenden Wiederherstellung befindet sich der Tabellenbereich nicht mehr in diesem Status. Der Tabellenbereich kann diesen Status auch bei Abbruch der Operation zur aktualisierenden Wiederherstellung verlieren.
- *Aktualisierende Wiederherstellung anstehend*. Ein Tabellenbereich wird in diesen Status versetzt, nachdem seine Wiederherstellung beendet wurde oder ein E/A-Fehler aufgetreten ist. Nach seiner Wiederherstellung kann der Tabellenbereich bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt werden. Nach einem E/A-Fehler muss die aktualisierende Wiederherstellung des Tabellenbereichs bis zum Ende der Protokolle erfolgen.

Verbessern der Wiederherstellungsleistung

Folgendes ist im Hinblick auf die Wiederherstellungsleistung 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® versucht automatisch, die zum Ausführen einer Sicherung oder Wiederherstellungsoperation benötigte Zeit zu minimieren, indem es optimale Werte für die Anzahl Puffer, die Puffergröße und die Parallelitätseinstellungen auswählt.

Die Werte basieren auf der Menge des für Dienstprogramme verfügbaren Zwischenspeichers, der Anzahl verfügbarer Prozessoren und der Datenbankkonfiguration.

- Gehen Sie wie folgt vor, um die zur Durchführung einer Wiederherstellungsoperation benötigte Zeit zu verringern:
  - Passen Sie die Größe der Wiederherstellungspuffer an. Die Puffergröße muss ein Vielfaches der Puffergröße sein, die bei der Sicherungsoperation verwendet wurde.
  - Erhöhen Sie die Anzahl der Puffer.

    Wenn Sie mehrere Puffer und externe E/A-Einheiten verwenden, sollten Sie zumindest doppelt so viele Puffer als externe Einheiten verwenden, um sicherzustellen, dass die externen Einheiten nicht auf Daten warten müssen. Die Größe der Puffer wirkt sich ebenfalls auf die Leistung der Wiederherstellungsoperation aus. Die ideale Größe der Wiederherstellungspuffer ist ein Vielfaches der Größe des durch EXTENTSIZE definierten Speicherbereichs für die Tabellenbereiche.

    Wenn Sie mehrere Tabellenbereiche mit verschiedenen EXTENTSIZE-Angaben haben, geben Sie einen Wert an, der ein Vielfaches des größten Werts für EXTENTSIZE darstellt.

    Die empfohlene *Mindestanzahl* Puffer ist die Summe der Anzahl der externen Einheiten und der für die Option PARALLELISM angegebenen Zahl.
  - Verwenden Sie mehrere Quelleneinheiten.
  - Legen Sie die Option PARALLELISM für die Wiederherstellungsoperation auf mindestens eins (1) größer als die Anzahl Quelleneinheiten fest.
- 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 Wiederherstellung 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 Sicherungsoperation 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 von der Wiederherstellung 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 Wiederherstellung bis zum Ende der Protokolle durchführen, so dass 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 Wiederherstellung bis zu einem bestimmten Zeitpunkt für diesen Tabellenbereich ausführen. Alle Tabellenbereiche für eine Tabelle müssen gleichzeitig bis zum selben Zeitpunkt aktualisierend wiederhergestellt werden.

- Folgendes gilt für Sicherungs- und Wiederherstellungsoperationen:
  - Es sollten mehrere E/A-Puffer und Einheiten verwendet werden.
  - Ordnen Sie mindestens doppelt so viele Puffer zu, wie Einheiten verwendet werden.
  - Überlasten Sie die Bandbreite der E/A-Einheitencontroller nicht.
  - Verwenden Sie eher mehr Puffer kleinerer Größe als wenige große Puffer.
  - Optimieren Sie die Anzahl und die Größe der Puffer entsprechend den Systemressourcen.
  - Verwenden Sie die Option PARALLELISM.
- DB2® verwendet zum parallelen Wiederherstellen nach einem Systemabsturz und zum aktualisierenden Wiederherstellen mehrere Agenten. Sie werden während dieser Operationen eine bessere Leistung feststellen, vor allem auf Maschinen mit symmetrischen Mehrprozessoren (SMP); die Verwendung mehrerer Agenten bei der Datenbankwiederherstellung nutzt die Vorteile der zusätzlichen CPUs auf SMP-Maschinen.

  Der mit der parallelen Wiederherstellung implementierte Agententyp ist db2agnsc. DB2 wählt die bei der Datenbankwiederherstellung zu verwendende Anzahl Agenten auf Basis der vorhandenen Anzahl CPUs auf der Maschine aus.

  DB2 verteilt Protokollsätze auf diese Agenten, so dass 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.

Zugehörige Konzepte:

- „Optimieren der Leistung des Sicherungsdienstprogramms" auf Seite 93
- „Wiederherstellung - Übersicht" auf Seite 95

Kapitel 2. Datenbanksicherung

In diesem Abschnitt wird das DB2 UDB-Sicherungsdienstprogramm beschrieben, mit dem Sicherungskopien von Datenbanken und Tabellenbereichen erstellt werden können.

Die folgenden Themen werden behandelt:

Sicherung - Übersicht

In der einfachsten Form des DB2®-Befehls BACKUP DATABASE müssen Sie lediglich den Aliasnamen der Datenbank angeben, die Sie sichern wollen. Beispiel:

```
db2 backup db sample
```

Wird der Befehl erfolgreich ausgeführt, wird ein neues Sicherungsimage 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. Auf Windows®-Betriebssystemen beispielsweise erstellt dieser Befehl (wenn er im Stammverzeichnis abgesetzt wird) ein Image, das in einer Verzeichnisliste wie folgt angezeigt wird:

```
 Verzeichnis von D:\SAMPLE.0\DB2\NODE0000\CATN0000\20010320

20.03.2001  12:26       <DIR>          .
20.03.2001  12:26       <DIR>          ..
20.03.2001  12:27           12.615.680 122644.001
```

Anmerkung: Wenn sich der DB2-Client und der DB2-Server nicht auf demselben System befinden, ermittelt DB2 das aktuelle Arbeitsverzeichnis auf der Clientmaschine und verwendet dieses Verzeichnis als Zielverzeichnis für die Sicherung auf dem Server. Aus diesem Grund wird empfohlen, dass Sie für das Sicherungsimage ein Zielverzeichnis angeben.

Sicherungsimages werden an der Zielposition erstellt, die Sie beim Aufrufen des Sicherungsdienstprogramms angeben. Sie können hierbei Folgendes angeben:

- Ein Verzeichnis (für Sicherungen auf Platte oder Diskette)
- Eine Einheit (für Sicherungen auf Band)
- Einen TSM-Server (Tivoli® Storage Manager)
- Den Server eines anderen Lieferanten

Die Datei des Wiederherstellungsprotokolls wird jedes Mal automatisch mit den Ergebnisinformationen aktualisiert, wenn Sie eine Sicherungsoperation für eine Datenbank starten. Diese Datei wird im selben Verzeichnis wie die Datenbankkonfigurationsdatei erstellt.

Auf UNIX®-Systemen setzen sich die Dateinamen für auf Platte erstellte Sicherungsimages aus verschiedenen durch Punkte voneinander getrennten Elementen zusammen:

```
DB_alias.typ.exemplar.NODEnnnn.CATNnnnn.zeitmarke.folgenr
```

Beispiel:

```
STAFF.0.DB201.NODE0000.CATN0000.19950922120112.001
```

Auf Windows-Betriebssystemen wird eine Unterverzeichnisbaumstruktur mit vier Ebenen verwendet:

```
DB_alias.typ\exemplarname\NODEnnnn\CATNnnnn\jjjjmmtt\hhmmss.folgenr
```

Beispiel:

```
SAMPLE.0\DB2\NODE0000\CATN0000\20010320\122644.001
```

DB_alias
Der aus 1 bis 8 Zeichen bestehende Datenbankaliasname, der beim Aufrufen des Sicherungsdienstprogramms angegeben wurde.

typ
Der Typ der Sicherungsoperation. Dabei gilt Folgendes: `0` steht für eine Sicherung der gesamten Datenbank, `3` steht für eine Sicherung auf Tabellenbereichsebene, und `4` steht für ein Sicherungsimage, das mit dem Befehl LOAD...COPY TO generiert wird.

exemplar
Der aus 1 bis 8 Zeichen bestehende Name des aktuellen Exemplars, der der Umgebungsvariablen **DB2INSTANCE** entnommen wird.

Knotennummer
Die Knotennummer. In nicht partitionierten Datenbanksystemen lautet diese Nummer immer `NODE0000`. In partitionierten Datenbanksystemen lautet die Nummer `NODE`*xxxx*, wobei *xxxx* die Nummer angibt, die dem Knoten in der Datei `db2nodes.cfg` zugeordnet ist.

Katalogknotennummer
Die Nummer des Katalogknotens der Datenbank. In nicht partitionierten Datenbanksystemen lautet diese Nummer immer `CATN0000`. In partitionierten Datenbanksystemen lautet die Nummer `CATN`*xxxx*, wobei *xxxx* die Nummer angibt, die dem Knoten in der Datei `db2nodes.cfg` zugeordnet ist.

zeitmarke
Ein Wert aus 14 Zeichen, der das Datum und die Uhrzeit des Zeitpunkts angibt, an dem die Sicherung ausgeführt 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).

- *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.

Wenn ein Sicherungsimage auf Band geschrieben wird, gilt Folgendes:

- Dateinamen werden nicht erstellt, die obigen Informationen werden jedoch im Sicherungsheader zu späteren Prüfzwecken gespeichert.
- Es muss eine Bandeinheit über die Standardschnittstelle des Betriebssystems verfügbar sein. In einem großen partitionierten Datenbanksystem 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, so dass der Zugriff auf diese Bandeinheiten jedem Datenbankpartitionsserver ermöglicht wird.
- In einem partitionierten Datenbanksystem können Sie darüber hinaus Produkte verwenden, die Funktionen virtueller Bandeinheiten implementieren, wie zum Beispiel REELlibrarian 4.2 oder CLIO/S. Sie können diese Produkte verwenden, um auf die mit anderen Knoten (Datenbankpartitionsservern) verbundene Bandeinheit über eine Pseudobandeinheit zuzugreifen. Der Zugriff auf die ferne Bandeinheit wird transparent ermöglicht, während der Zugriff auf die Pseudobandeinheit über die Standardschnittstelle des Betriebssystems erfolgen kann.

Es ist nicht möglich, eine Datenbank zu sichern, die sich in einem unbrauchbaren Status befindet, es sei denn, für diese Datenbank steht eine Sicherung an. Wenn sich einer der Tabellenbereiche in einem abnormalen Status befindet, können Sie kein Sicherungsimage der Datenbank bzw. dieses Tabellenbereichs erstellen, außer sie bzw. er befindet sich im Status *Aktualisierende Wiederherstellung anstehend*.

Gleichzeitig ablaufende Sicherungsoperationen für denselben Tabellenbereich sind nicht zulässig. Sobald eine Sicherungsoperation für einen Tabellenbereich eingeleitet wurde, schlagen alle nachfolgenden Versuche fehl (SQL2048).

Wenn eine Datenbank oder ein Tabellenbereich nur teilweise wiederhergestellt wurde, da während der Wiederherstellung ein Systemabsturz auftrat, müssen Sie die Datenbank bzw. den Tabellenbereich erfolgreich wiederherstellen, bevor Sie ein Sicherungsimage davon erstellen können.

Die Sicherung schlägt fehl, wenn die Liste der zu sichernden Tabellenbereiche den Namen eines temporären Tabellenbereichs enthält.

Das Sicherungsdienstprogramm erlaubt die Steuerung des gemeinsamen Zugriffs für mehrere Prozesse, die Sicherungskopien verschiedener Datenbanken anlegen. Diese Steuerung des gemeinsamen Zugriffs gewährleistet, dass die Zieleinheiten für die Sicherung bis zur Beendigung aller Sicherungsoperationen geöffnet bleiben. Wenn während der Sicherungsoperation ein Fehler auftritt und ein geöffneter Behälter nicht geschlossen werden kann, erhalten andere Sicherungsoperationen mit demselben Ziellaufwerk eventuell Nachrichten über Zugriffsfehler. Zur Behebung etwaiger Zugriffsfehler müssen Sie die Sicherungsoperation, die den Fehler verursachte, beenden und die Verbindung zur Zieleinheit trennen. Wenn Sie das Sicherungsdienstprogramm für gleichzeitig ablaufende Sicherungsoperationen auf Band verwenden, müssen Sie sicherstellen, dass diese Operationen nicht dasselbe Band ansteuern.

Anzeigen von Sicherungsinformationen

Mit dem Dienstprogramm **db2ckbkp** können Sie Informationen zu vorhandenen Sicherungsimages anzeigen. Es ermöglicht Folgendes:

- Testen der Integrität des Sicherungsimages und Klärung der Frage, ob ein Wiederherstellen möglich ist.
- Anzeigen der im Sicherungsheader gespeicherten Informationen.
- Anzeigen von Informationen zu den Objekten und des Protokollheaders im Sicherungsimage.

Zugehörige Konzepte:

- „Entwickeln einer Sicherungs- und Wiederherstellungsstrategie" auf Seite 3
- „Datei des Wiederherstellungsprotokolls" auf Seite 62
- „Einschließen von Protokolldateien in ein Sicherungsimage" auf Seite 60
- „Automatische Verwaltung" im Handbuch *Systemverwaltung: Konzept*

Zugehörige Referenzen:

- Anhang F, „Tivoli Storage Manager", auf Seite 365

Erforderliche Zugriffsrechte und Berechtigungen zur Verwendung des Sicherungsdienstprogramms

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf sie zuzugreifen. Berechtigungsstufen stellen eine Methode dar, Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zu gruppieren. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte. Benutzer können nur auf solche Objekte zugreifen, für die sie die entsprechende Berechtigung besitzen, d. h., für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Sie benötigen die Berechtigung SYSADM, SYSCTRL oder SYSMAINT, um das Sicherungsdienstprogramm verwenden zu können.

Verwenden des Sicherungsdienstprogramms

Vorbedingungen:

Es sollte noch keine Verbindung zu der zu sichernden Datenbank bestehen: Das Datenbanksicherungsdienstprogramm stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der Sicherungsoperation beendet wird. Wenn bereits eine Verbindung zu der zu sichernden Datenbank besteht, wird diese Verbindung beim Absetzen des Befehls BACKUP DATABASE unterbrochen, und die Sicherungsoperation wird fortgesetzt.

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln. Das Sicherungsimage bleibt auf dem Datenbankserver, sofern Sie kein Speicherverwaltungsprodukt wie z. B. Tivoli Storage Manager (TSM) verwenden.

In einem partitionierten Datenbanksystem werden Datenbankpartitionen jeweils separat gesichert. Die Operation ist für den Datenbankpartitionsserver, auf dem Sie das Dienstprogramm aufrufen, lokal. Sie können jedoch **db2_all** von einem der Datenbankpartitionsserver im Exemplar ausführen, um das Sicherungsdienst-

programm für eine Liste von Servern aufzurufen, die Sie anhand ihrer jeweiligen Knotennummer angeben. (Verwenden Sie den Befehl LIST NODES, um festzustellen, auf welchen Knoten oder Datenbankpartitionsservern Benutzertabellen vorhanden sind.) Wenn Sie auf diese Weise vorgehen, müssen Sie zuerst den Katalogknoten und anschließend die anderen Datenbankpartitionen sichern. Zur Sicherung von Datenbankpartitionen können Sie auch den Befehlseditor verwenden. Da diese Vorgehensweise jedoch die aktualisierende Wiederherstellung nicht unterstützt, empfiehlt es sich, die auf diesen Knoten vorhandene Datenbank regelmäßig zu sichern. Für den Fall einer möglichen Beschädigung der Datei `db2nodes.cfg` ist es ratsam, mit allen Sicherungskopien, die Sie erstellen, auch eine Kopie dieser Datei zu speichern.

Auf einem System, in dem mit verteilten Anforderungen gearbeitet wird, werden die Sicherungsoperationen 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, aber nicht auf das aktuelle Release migriert wurde, müssen Sie sie migrieren. Andernfalls ist es nicht möglich, eine Sicherungskopie der Datenbank anzulegen.

Einschränkungen:

Für das Sicherungsdienstprogramm gelten die folgenden Einschränkungen:

- Es ist nicht möglich, gleichzeitig eine Sicherung eines Tabellenbereichs und eine Wiederherstellung eines Tabellenbereichs auszuführen. Dies gilt auch dann, wenn Sicherung und Wiederherstellung für unterschiedliche Tabellenbereiche erfolgen.
- Wenn Sie in der Lage sein wollen, in einer Umgebung mit partitionierten Datenbanken eine aktualisierende Wiederherstellung durchzuführen, müssen Sie die Datenbank auf den Knoten der Liste regelmäßig sichern und über mindestens ein Sicherungsimage der übrigen Knoten im System verfügen (auch von Knoten, die keine Benutzerdaten für diese Datenbank enthalten). In zwei Situationen ist ein Sicherungsimage einer Datenbankpartition auf einem Datenbankpartitionsserver erforderlich, der keine Benutzerdaten für die Datenbank enthält:
  - Sie haben dem Datenbanksystem einen Datenbankpartitionsserver hinzugefügt, nachdem die letzte Sicherung erstellt wurde, und müssen eine aktualisierende Wiederherstellung auf diesem Datenbankpartitionsserver durchführen.
  - Es wird eine Wiederherstellung bis zu einem bestimmten Zeitpunkt durchgeführt, für die erforderlich ist, dass sich alle Datenbankpartitionen im Status *Aktualisierende Wiederherstellung anstehend* befinden.
  - Onlinesicherungsoperationen 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)

Prozedur:

Das Sicherungsdienstprogramm kann über den Befehlszeilenprozessor (CLP), das Notizbuch bzw. den Assistenten **Datenbank sichern** in der Steuerzentrale oder die Anwendungsprogrammierschnittstelle **db2Backup** aufgerufen werden.

Es folgt ein Beispiel für den Befehl BACKUP DATABASE, der über den CLP abgesetzt wird:

```
db2 backup database sample to c:\DB2Backups
```

Gehen Sie wie folgt vor, um das Notizbuch bzw. den Assistenten **Datenbank sichern** zu öffnen:

1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis Sie den Ordner **Datenbanken** finden.
2. Klicken Sie den Ordner **Datenbanken** an. Alle vorhandenen Datenbanken werden auf der rechten Seite des Fensters, im Inhaltsteilfenster, angezeigt.
3. Klicken Sie im Inhaltsteilfenster die gewünschte Datenbank mit Maustaste 2 an, und wählen Sie **Datenbank sichern** oder **Sichern —> Datenbank mit Assistent** im Kontextmenü aus. Das Notizbuch bzw. der Assistent **Datenbank sichern** wird geöffnet.

Zusatzinformationen bietet die Onlinehilfefunktion der Steuerzentrale.

Zugehörige Konzepte:

- „Administrative APIs in Embedded SQL or DB2 CLI Programs" im Handbuch *Application Development Guide: Programming Client Applications*
- „Einführung der Plug-in-Architektur für die Steuerzentrale" im Handbuch *Systemverwaltung: Implementierung*

Zugehörige Tasks:

- „Migrieren von Datenbanken" im Handbuch *DB2 Universal Database für DB2-Server Einstieg*

Zugehörige Referenzen:

- „LIST DBPARTITIONNUMS Command" im Handbuch *Command Reference*
- „db2Backup - Datenbank sichern" auf Seite 85

Sichern 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 ist deswegen vorhanden, weil DB2® den Sicherungsimage-Header in 4-KB-Blöcken schreibt. Die einzigen festen Blockgrößen, die von DB2 unterstützt werden, sind 512, 1024, 2048 und 4096 Byte. Wenn Sie mit einer festen Blockgröße arbeiten, können Sie für die Sicherung eine beliebige Puffergröße angeben. Es ist jedoch mögliche, dass die Sicherung nicht

erfolgreich abgeschlossen werden kann, wenn die feste Blockgröße nicht mit einem der von DB2 unterstützten Werte übereinstimmt.

Wenn Ihre Datenbank umfangreich ist und Sie eine feste Blockgröße verwenden, wird möglicherweise sehr viel Zeit für die Sicherung benötigt. Aus diesem Grund kann es günstiger sein, variable Blockgrößen einzusetzen.

Anmerkung: Die Verwendung variabler Blockgrößen wird derzeit *nicht* unterstützt. Wenn Sie diese Option verwenden müssen, stellen Sie sicher, dass Sie über gut erprobte Verfahren verfügen, die Ihnen eine erfolgreiche Wiederherstellung unter Verwendung von Sicherungsimages ermöglichen, die mit variabler Blockgröße erstellt wurden.

Bei der Verwendung variabler Blockgrößen müssen Sie eine Sicherungspuffergröße 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:

<einheit>
Ein gültiger Bandeinheitenname. Der Standardwert unter 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 zur Wiederherstellung ein mit variabler Blockgröße erstelltes Sicherungsimage verwendet wird, kann dies zu einer Fehlermeldung führen. In diesem Fall muss das Image eventuell mit einer passenden Blockgröße erneut geschrieben werden. Das folgende Beispiel zeigt einen unter AIX ausgeführten Befehl:

```
tctl -b 0 -Bn -f /dev/rmt0 read > sicherungsdatei.datei
dd if=sicherungsdatei.datei of=/dev/rmt0 obs=4096 conv=sync
```

Durch diesen Befehl wird in einer Datei des Namens `sicherungsdatei.datei` ein Speicherauszug des Sicherungsimages 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. Ein möglicher Lösungsansatz 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 unformatierten Einheit spei-

chern und ihn anschließend von dieser Einheit als Speicherauszug auf ein Band schreiben. Die Schwierigkeiten bei dieser Methode ergeben sich dadurch, dass beim Befehl **dd** die Anzahl der auf die unformatierte Einheit geschriebenen Blöcke protokolliert werden *muss*. Diese Blockanzahl muss beim Zurückversetzen des Images auf ein Band angegeben werden. Wird der Befehl **dd** dazu verwendet, einen Imagespeicherauszug von einer unformatierten Einheit auf ein Band zu schreiben, wird der gesamte Inhalt der unformatierten Einheit auf das Band übertragen. Der Befehl **dd** kann nicht feststellen, wie viel Speicherplatz auf der unformatierten Einheit zum Speichern des Images verwendet wird.

Wenn Sie das Sicherungsdienstprogramm verwenden, müssen Sie den oberen Grenzwert für die Blockgröße der verwendeten Bandeinheit(en) kennen. Es folgen einige Beispiele:

Einheit	Anschlusseinrichtung	Blockgrößengrenzwert	Grenzwert für DB2-Puffergröße (in 4-KB-Seiten)
8 mm	scsi	131.072	32
3420	s370	65.536	16
3480	s370	65.536	16
3490	s370	65.536	16
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

Anmerkungen:

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.

Sichern in benannten Pipes

Auf UNIX-Systemen wird Unterstützung für das Sichern in (und Wiederherstellen aus) lokalen benannten Pipes geboten.

Vorbedingungen:

Das Aus- und Eingabeprogramm der benannten Pipe müssen sich auf derselben Maschine befinden. Die Pipe muss in einem lokalen Dateisystem vorhanden und lokalisierbar 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.

Prozedur:

Das folgende Beispiel gilt unter AIX:
1. Erstellen Sie eine benannte Pipe:
   ```
   mkfifo /u/dmcinnis/meinepipe
   ```
2. Wenn dieses Sicherungsimage vom Dienstprogramm RESTORE verwendet werden soll, muss die Wiederherstellungsoperation *vor* der Sicherungsoperation aufgerufen werden, so dass alle Daten erfasst werden:
   ```
   db2 restore db sample into meineneuedb from /u/dmcinnis/meinepipe
   ```
3. Verwenden Sie diese Pipe als Ziel für eine Datenbanksicherungsoperation:
   ```
   db2 backup db sample to /u/dmcinnis/meinepipe
   ```

Zugehörige Tasks:

- „Verwenden des Sicherungsdienstprogramms" auf Seite 74

Zugehörige Referenzen:
- „BACKUP DATABASE" auf Seite 79
- „RESTORE DATABASE" auf Seite 102

BACKUP DATABASE

Erstellt eine Sicherungskopie einer Datenbank bzw. eines Tabellenbereichs.

Geltungsbereich:

Dieser Befehl wirkt sich nur auf die Datenbankpartition aus, auf der er ausgeführt wird.

Berechtigung:

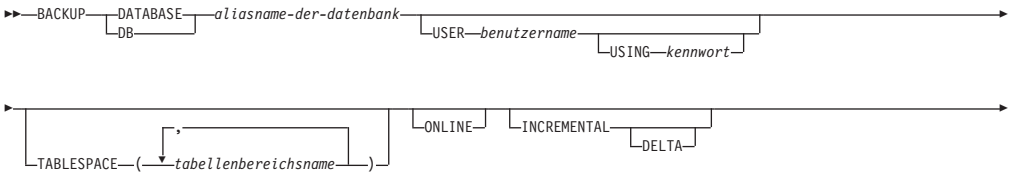
Eine der folgenden Berechtigungen:
- *sysadm*
- *sysctrl*
- *sysmaint*

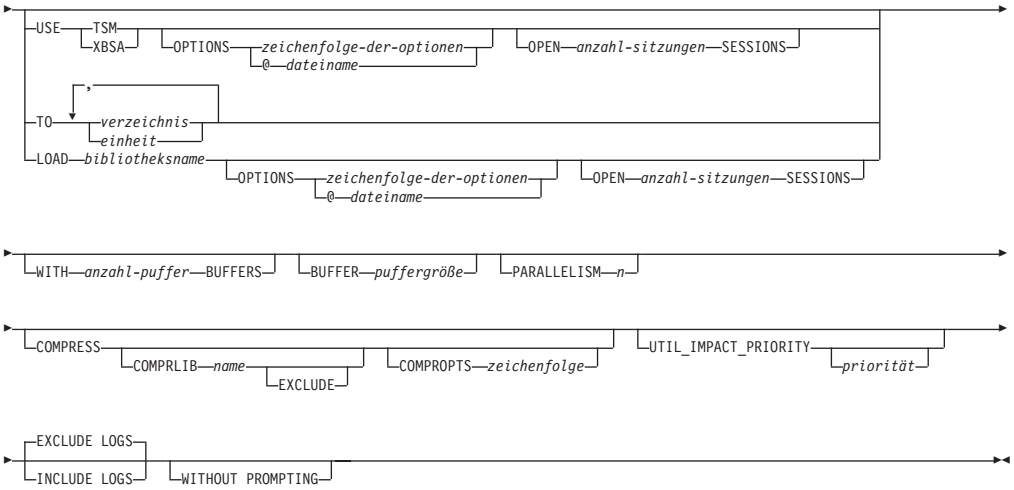
Erforderliche Verbindung:

Datenbank. Mit diesem Befehl stellen Sie automatisch eine Verbindung zur angegebenen Datenbank her.

Anmerkung: Wenn bereits eine Verbindung zu der angegebenen Datenbank existiert, wird diese Verbindung beendet und speziell für die Sicherungsoperation eine neue Verbindung hergestellt. Die Verbindung wird nach Abschluss der Sicherungsoperation beendet.

Befehlssyntax:





Befehlsparameter:

DATABASE aliasname-der-datenbank
Gibt den Aliasnamen der zu sichernden Datenbank an.

USER benutzername
Gibt den Benutzernamen an, unter dem die Datenbank gesichert werden soll.

USING kennwort
Das Kennwort, das verwendet wird, um den Benutzernamen zu authentifizieren. Wenn kein Kennwort angegeben wird, wird der Benutzer zur Eingabe des Kennworts aufgefordert.

TABLESPACE tabellenbereichsname
Eine Namensliste, mit der die zu sichernden Tabellenbereiche angegeben werden.

ONLINE
Gibt eine Onlinesicherung an. Der Standardwert ist eine Offlinesicherung. Eine Onlinesicherung ist nur bei Datenbanken möglich, die mit den aktivierten Optionen *logretain* oder *userexit* konfiguriert wurden. Während einer Onlinesicherung aktiviert DB2 für alle in SMS-Tabellenbereichen vorhandenen Tabellen bei der Verarbeitung der Tabellen IN-Sperren (IN - Intent None) bzw. S-Sperren (S - Share) für die in SMS-Tabellenbereichen vorhandenen LOB-Daten.

INCREMENTAL
Gibt ein kumulatives (inkrementelles) Sicherungsimage an. Ein inkrementelles Sicherungsimage ist eine Kopie aller Datenbankdaten, die seit der letzten erfolgreichen vollständigen Sicherungsoperation geändert wurden.

DELTA
Gibt ein nicht kumulatives (Delta-)Sicherungsimage an. Ein Deltasicherungsimage ist eine Kopie aller Datenbankdaten, die seit der letzten erfolgreichen Sicherungsoperation eines beliebigen Typs geändert wurden.

USE TSM
Gibt an, dass die Sicherung die TSM-Ausgabe (TSM - Tivoli Storage Manager) verwenden soll.

USE XBSA
Gibt an, dass die XBSA-Schnittstelle verwendet werden soll. XBSA (Backup Services APIs) ist eine offene Anwendungsprogrammierschnittstelle für

Anwendungen oder Tools, die eine Datenspeicherverwaltung für Sicherungs- oder Archivierungszwecke benötigen.

OPTIONS

"zeichenfolge-der-optionen"
Gibt die Optionen an, die für die Sicherungsoperation verwendet werden sollen. Die Zeichenfolge wird an die Unterstützungsbibliothek des Herstellers (z. B. TSM) genau so, wie sie eingegeben wurde und ohne Anführungszeichen übermittelt.

Anmerkung: Wird diese Option angegeben, wird der mit dem Datenbankkonfigurationsparameter VENDOROPT angegebene Wert außer Kraft gesetzt.

@dateiname
Gibt an, dass die Optionen, die für die Sicherungsoperation verwendet werden, in einer Datei enthalten sind, die sich auf dem DB2-Server befindet. Die Zeichenfolge wird an die Unterstützungsbibliothek des Herstellers (z. B. TSM) übermittelt. Die Datei muss ein vollständig qualifizierter Dateiname sein.

OPEN anzahl-sitzungen SESSIONS
Die Anzahl E/A-Sitzungen, die zwischen DB2 und TSM oder einem Sicherungsprogramm eines anderen Lieferanten erstellt werden sollen.

Anmerkung: Dieser Parameter hat bei einer Sicherung auf Band, Platte oder eine andere lokale Einheit keine Auswirkung.

TO verzeichnis/einheit
Eine Liste mit Namen von Verzeichnissen oder Bandeinheiten. Der vollständige Pfad, in dem sich das Verzeichnis befindet, muss angegeben werden. Wenn die Parameter USE TSM, TO und LOAD nicht angegeben werden, wird als aktuelles Arbeitsverzeichnis für den Clientcomputer das Standardzielverzeichnis für das Sicherungsimage verwendet. Dieses Zielverzeichnis oder diese Einheit muss auf dem Datenbankserver vorhanden sein. Dieser Parameter kann wiederholt werden, um die Zielverzeichnisse und Einheiten anzugeben, die das Sicherungsimage umfassen wird. Wenn mehr als ein Ziel angegeben wird (z. B. ziel1, ziel2 und ziel3), wird ziel1 zuerst geöffnet. Der Datenträgerheader und die Gerätedateien (einschließlich der Konfigurationsdatei, der Tabellenbereichstabelle und der Protokolldatei) werden in ziel1 platziert. Alle übrigen Ziele werden geöffnet und während der Sicherungsoperation parallel verwendet. Weil es unter Windows-Betriebssystemen keine allgemeine Bandunterstützung gibt, erfordert jeder Bandeinheitentyp einen eindeutigen Einheitentreiber. Benutzer müssen sich an die Benennungseinschränkung (8+3) halten, um eine Sicherung auf ein FAT-Dateisystem unter Windows-Betriebssystemen ausführen zu können.

Die Verwendung von Bandeinheiten und Disketten könnte Nachrichten generieren und zur Eingabe von Benutzereingaben auffordern. Gültige Antwortoptionen sind:

c Fortsetzen. Verwendung der Einheit fortsetzen, die die Warnung generiert hat (zum Beispiel, wenn ein neues Band eingelegt wurde).

d Einheit beenden. Nur die Verwendung der Einheit beenden, die die Warnung generiert hat (zum Beispiel, wenn keine Bänder mehr vorhanden sind).

t Beenden. Die Sicherungsoperation abbrechen.

Wenn das Bandsystem die Möglichkeit zum eindeutigen Verweis auf ein Sicherungsimage nicht unterstützt, wird empfohlen, dass nicht mehrere Sicherungskopien derselben Datenbank auf einem Band gespeichert werden.

LOAD bibliotheksname
Der Name der gemeinsam benutzten Bibliothek (DLL unter Windows-Betriebssystemen), die die Sicherungs- und Wiederherstellungs-E/A-Funktionen des Lieferanten enthält, die verwendet werden sollen. Der Name kann einen vollständigen Pfad enthalten. Wenn kein vollständiger Pfad angegeben wird, wird standardmäßig der Pfad verwendet, auf dem sich das Benutzerexitprogramm befindet.

WITH anzahl-puffer BUFFERS
Die Anzahl zu verwendender Puffer. DB2 wählt automatisch einen optimalen Wert für diesen Parameter aus, wenn Sie nicht explizit einen Wert angeben. Wenn Sie eine Sicherung an mehreren Speicherpositionen erstellen, könnten Sie beispielsweise eine größere Anzahl Puffer angeben, um die Leistung zu verbessern.

BUFFER puffergröße
Die Größe des bei der Erstellung des Sicherungsimages verwendeten Puffers in 4-KB-Seiten. DB2 wählt automatisch einen optimalen Wert für diesen Parameter aus, wenn Sie nicht explizit einen Wert angeben. Der Mindestwert für diesen Parameter ist 8 Seiten.

Wenn Sie ein Band mit variabler Blockgröße verwenden, reduzieren Sie die Puffergröße auf den Bereich, den die Bandeinheit unterstützt. Ansonsten könnte die Sicherungsoperation erfolgreich beendet werden, das Ergebnisimage aber nicht wiederherstellbar sein.

Wenn Sie Bandeinheiten unter SCO UnixWare 7 verwenden, geben Sie die Puffergröße 16 an.

Bei den meisten Versionen von Linux erhalten Sie die Fehlernachricht SQL2025N, Ursachencode 75, wenn Sie für Sicherungsoperationen eine SCSI-Bandeinheit und die DB2-Standardpuffergröße verwenden. Um einen Überlauf der internen SCSI-Puffer unter Linux zu verhindern, sollten Sie die folgende Formel verwenden:

```
pufferseiten <= ST_MAX_BUFFERS * ST_BUFFER_BLOCKS / 4
```

Dabei ist *pufferseiten* der Wert, den Sie mit dem Parameter BUFFER verwenden wollen, und `ST_MAX_BUFFERS` und `ST_BUFFER_BLOCKS` werden im Linux-Kernel im Verzeichnis `drivers/scsi` definiert.

PARALLELISM n
Ermittelt die Anzahl Tabellenbereiche, die vom Sicherungsdienstprogramm parallel gelesen werden können. DB2 wählt automatisch einen optimalen Wert für diesen Parameter aus, wenn Sie nicht explizit einen Wert angeben.

UTIL_IMPACT_PRIORITY *priorität*
Gibt an, dass die Sicherung im gedrosselten Modus mit der angegeben Priorität ausgeführt wird. Das Drosseln ermöglicht es Ihnen, die Leistungsauswirkungen der Sicherungsoperation zu steuern. Die Priorität kann eine beliebige Zahl zwischen 1 und 100 sein. Dabei ist 1 die niedrigste Priorität, 100 die höchste Priorität. Wenn das Schlüsselwort UTIL_IMPACT_PRIORITY ohne Priorität angegeben wird, wird die Sicherung mit einer standardmäßigen Priorität von 50 ausgeführt. Wird UTIL_IMPACT_PRIO-

RITY nicht angegeben, wird die Sicherung im ungedrosselten Modus ausgeführt. Es muss eine Richtlinie für die Auswirkungen definiert werden, indem der Konfigurationsparameter *util_impact_lim* für eine Sicherung angegeben wird, die im gedrosselten Modus ausgeführt werden soll.

COMPRESS
Gibt an, dass die Sicherung komprimiert werden soll.

COMPRLIB *name*
Gibt den Namen der Bibliothek an, die zum Ausführen der Komprimierung verwendet werden soll. Der Name muss ein vollständig qualifizierter Pfad für eine Datei auf dem Server sein. Wird dieser Parameter nicht angegeben, wird die DB2-Standardkomprimierungsbibliothek verwendet. Wenn die angegebene Bibliothek nicht geladen werden kann, schlägt die Sicherung fehl.

EXCLUDE
Gibt an, dass die Komprimierungsbibliothek nicht im Sicherungsimage gespeichert werden wird.

COMPROPTS *zeichenfolge*
Beschreibt einen Block binärer Daten, der an die Initialisierungsroutine in der Komprimierungsbibliothek übergeben wird. DB2 übergibt diese Zeichenfolge direkt vom Client an den Server, daher muss die Handhabung der Bytefolgeumkehrung oder Codepagekonvertierung von der Komprimierungsbibliothek übernommen werden. Ist '@' das erste Zeichen des Datenblocks, werden die restlichen Daten von DB2 als Name einer Datei interpretiert, die sich auf dem Server befindet. DB2 ersetzt in diesem Fall den Inhalt der Zeichenfolge durch den Inhalt dieser Datei und übergibt diesen neuen Wert stattdessen an die Initialisierungsroutine. Die maximal zulässige Länge für die Zeichenfolge *zeichenfolge* ist 1024 Byte.

EXCLUDE LOGS
Gibt an, dass das Sicherungsimage keine Protokolldateien enthalten soll.

Anmerkung: Wird eine Offlinesicherungsoperation ausgeführt, werden Protokolle unabhängig von dieser Option grundsätzlich ausgeschlossen.

INCLUDE LOGS
Gibt an, dass das Sicherungsimage den Bereich an Protokolldateien einschließen soll, der zum Wiederherstellen und aktualisierenden Wiederherstellen dieses Images zu einem konsistenten Zeitpunkt erforderlich ist. Diese Option gilt nicht für Offlinesicherungen.

WITHOUT PROMPTING
Gibt an, dass die Sicherung nicht überwacht ausgeführt wird und dass alle Aktionen, die normalerweise einen Benutzereingriff erfordern, eine Fehlernachricht zurückgeben.

Beispiele:

1. Im folgenden Beispiel ist die Datenbank WSDB auf allen 4 Partitionen definiert und von 0 bis 3 nummeriert. Auf den Pfad /dev3/backup kann von allen Partitionen aus zugegriffen werden. Partition 0 ist die Katalogtabellenpartition und muss separat gesichert werden, da es sich dabei um eine Offlinesicherung handelt. Wenn Sie eine Offlinesicherung für alle Partitionen der Datenbank WSDB in das Verzeichnis /dev3/backup ausführen wollen, setzen Sie auf einer der Datenbankpartitionen die folgenden Befehle ab:

```
db2_all '<<+0< db2 BACKUP DATABASE wsdb TO /dev3/backup'
db2_all '|<<-0< db2 BACKUP DATABASE wsdb TO /dev3/backup'
```

Im zweiten Befehl setzt das Dienstprogramm db2_all nacheinander für jede Datenbankpartition (außer Partition 0) denselben Sicherungsbefehl ab. Alle vier Sicherungsimages der Datenbankpartitionen werden im Verzeichnis /dev3/backup gespeichert.

2. Im folgenden Beispiel wird die Datenbank SAMPLE in zwei gleichzeitig ablaufenden TSM-Clientsitzungen auf einem TSM-Server gesichert. DB2 berechnet die optimale Puffergröße für diese Umgebung.

   ```
   db2 backup database sample use tsm open 2 sessions with 4 buffers
   ```

3. Im folgenden Beispiel wird eine Sicherung auf Tabellenbereichsebene von Tabellenbereichen (syscatspace, userspace1) der Datenbank payroll auf Band ausgeführt.

   ```
   db2 backup database payroll tablespace (syscatspace, userspace1) to
      /dev/rmt0, /dev/rmt1 with 8 buffers without prompting
   ```

4. Mit den Schlüsselwörtern USE TSM OPTIONS können die TSM-Informationen angegeben werden, die für die Sicherungsoperation verwendet werden sollen. Das folgende Beispiel zeigt, wie mit den Schlüsselwörtern USE TSM OPTIONS ein vollständig qualifizierter Dateiname angegeben wird:

   ```
   db2 backup db sample use TSM options @/u/dmcinnis/myoptions.txt
   ```

   Die Datei myoptions.txt enthält die folgenden Informationen: -fromnode=bar -fromowner=dmcinnis

5. Es folgt ein Beispiel für eine Strategie zur wöchentlichen Teilsicherung für eine wiederherstellbare Datenbank. Diese Strategie umfasst eine wöchentliche vollständige Sicherung der Datenbank, eine tägliche nicht kumulative Sicherung (Deltasicherung) sowie eine kumulative Sicherung (inkrementell) in der Wochenmitte:

   ```
   (So) db2 backup db sample use tsm
   (Mo) db2 backup db sample online incremental delta use tsm
   (Di) db2 backup db sample online incremental delta use tsm
   (Mi) db2 backup db sample online incremental use tsm
   (Do) db2 backup db sample online incremental delta use tsm
   (Fr) db2 backup db sample online incremental delta use tsm
   (Sa) db2 backup db sample online incremental use tsm
   ```

6. Im folgenden Beispiel werden drei identische Zielverzeichnisse für eine Sicherungsoperation für die Datenbank SAMPLE angegeben. Dies ist beispielsweise erforderlich, wenn das Zieldateisystem aus mehreren physischen Platten besteht.

   ```
   db2 backup database sample to /dev3/backup, /dev3/backup, /dev3/backup
   ```

   Die Daten werden gleichzeitig in den drei Zielverzeichnissen gesichert, und es werden drei Sicherungsimages mit den Erweiterungen .001, .002 und .003 erstellt.

Zugehörige Referenzen:

- „RESTORE DATABASE" auf Seite 102
- „ROLLFORWARD DATABASE" auf Seite 144

db2Backup - Datenbank sichern

Erstellt eine Sicherungskopie einer Datenbank bzw. eines Tabellenbereichs.

Geltungsbereich:

Diese API wirkt sich nur auf die Datenbankpartition aus, auf der sie ausgeführt wird.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Erforderliche Verbindung:

Datenbank. Mit dieser API stellen Sie automatisch eine Verbindung zur angegebenen Datenbank her.

Die Verbindung wird nach Abschluss der Sicherung beendet.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2Backup */
/* ... */
SQL_API_RC SQL_API_FN
db2Backup (
  db2Uint32 versionNumber,
  void         *pDB2BackupStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2BackupStruct
{
  char                        *piDBAlias;
  char                        oApplicationId[SQLU_APPLID_LEN+1];
  char                        oTimestamp[SQLU_TIME_STAMP_LEN+1];
  struct db2TablespaceStruct  *piTablespaceList;
  struct db2MediaListStruct   *piMediaList;
  char                        *piUsername;
  char                        *piPassword;
  void                        *piVendorOptions;
  db2Uint32                   iVendorOptionsSize;
  db2Uint32                   oBackupSize;
  db2Uint32                   iCallerAction;
  db2Uint32                   iBufferSize;
  db2Uint32                   iNumBuffers;
  db2Uint32                   iParallelism;
   db2Uint32 iOptions;
  db2Uint32                   iUtilImpactPriority;
  char                        *piComprLibrary;
  void                        *piComprOptions;
  db2Uint32                   iComprOptionsSize;
} db2BackupStruct;
```

```
typedef SQL_STRUCTURE db2TablespaceStruct
{
  char                        **tablespaces;
  db2Uint32                   numTablespaces;
} db2TablespaceStruct;

typedef SQL_STRUCTURE db2MediaListStruct
{
  char                        **locations;
  db2Uint32                   numLocations;
  char                        locationType;
} db2MediaListStruct;
/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2Backup */
/* ... */
SQL_API_RC SQL_API_FN
db2gBackup (
  db2Uint32 versionNumber,
  void         *pDB2gBackupStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2gBackupStruct
{
  char                        *piDBAlias;
  db2Uint32                   iDBAliasLen;
  char                        *poApplicationId;
  db2Uint32                   iApplicationIdLen;
  char                        *poTimestamp;
  db2Uint32                   iTimestampLen;
  struct db2gTablespaceStruct *piTablespaceList;
  struct db2gMediaListStruct  *piMediaList;
  char                        *piUsername;
  db2Uint32                   iUsernameLen;
  char                        *piPassword;
   db2Uint32 iPasswordLen;
  void                        *piVendorOptions;
  db2Uint32                   iVendorOptionsSize;
  db2Uint32                   oBackupSize;
  db2Uint32                   iCallerAction;
  db2Uint32                   iBufferSize;
  db2Uint32                   iNumBuffers;
  db2Uint32                   iParallelism;
   db2Uint32 iOptions;
  db2Uint32                   iUtilImpactPriority;
  char                        *piComprLibrary;
  db2Uint32                   iComprLibraryLen;
  void                        *piComprOptions;
  db2Uint32                   iComprOptionsSize;
} db2gBackupStruct;

typedef SQL_STRUCTURE db2gTablespaceStruct
{
  struct db2Char              *tablespaces;
  db2Uint32                   numTablespaces;
} db2gTablespaceStruct;

typedef SQL_STRUCTURE db2gMediaListStruct
{
  struct db2Char              *locations;
  db2Uint32                   numLocations;
  char                        locationType;
} db2gMediaListStruct;
```

```
typedef SQL_STRUCTURE db2Char
{
   char            *pioData;
   db2Uint32       iLength;
   db2Uint32       oLength;
} db2Char;
/* ... */
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2BackupStruct*, übergeben wird.

pDB2BackupStruct
Eingabe. Ein Zeiger auf die Struktur *db2BackupStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

piDBAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der zu sichernden Datenbank (wie im Systemdatenbankverzeichnis katalogisiert) enthält.

iDBAliasLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Datenbankaliasnamens in Byte angibt.

oApplicationId
Ausgabe. Die API gibt eine Zeichenfolge zurück, die den Agenten angibt, der die Anwendung bedient. Dieser Parameter kann verwendet werden, um mit dem Datenbankmonitor Informationen zum Fortschritt der Sicherungsoperation zu erhalten.

poApplicationId
Ausgabe. Geben Sie einen Puffer der Länge `SQLU_APPLID_LEN+1` (definiert in `sqlutil.h`) an. Die API gibt eine Zeichenfolge zurück, die den Agenten angibt, der die Anwendung bedient. Dieser Parameter kann verwendet werden, um mit dem Datenbankmonitor Informationen zum Fortschritt der Sicherungsoperation zu erhalten.

iApplicationIdLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Puffers *poApplicationId* in Byte angibt. Diese Zahl sollte gleich `SQLU_APPLID_LEN+1` (definiert in `sqlutil.h`) sein.

oTimestamp
Ausgabe. Die API gibt die Zeitmarke des Sicherungsimages zurück.

poTimestamp
Ausgabe. Geben Sie einen Puffer der Länge `SQLU_TIME_STAMP_LEN+1` (definiert in `sqlutil.h`) an. Die API gibt die Zeitmarke des Sicherungsimages zurück.

iTimestampLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Puffers *poTimestamp* in Byte angibt. Diese Zahl sollte gleich `SQLU_TIME_STAMP_LEN+1` (definiert in `sqlutil.h`) sein.

piTablespaceList
Eingabe. Eine Liste der zu sichernden Tabellenbereiche. Nur für Sicherungen auf Tabellenbereichsebene erforderlich. Muss für eine Sicherung auf Datenbankebene NULL sein. Siehe Struktur db2TablespaceStruct.

piMediaList
Eingabe. Diese Struktur ermöglicht dem aufrufenden Programm die Angabe der Zieladresse für die Sicherungsoperation. Die bereitgestellten Daten hängen vom Wert des Parameters *locationType* ab. Folgende Werte sind für *locationType* (definiert in `sqlutil.h`) gültig:

SQLU_LOCAL_MEDIA
Lokale Einheiten (eine Kombination von Bändern, Platten oder Disketten).

SQLU_TSM_MEDIA
TSM. Wenn der Positionszeiger auf NULL gesetzt ist, wird die mit DB2 gelieferte gemeinsam benutzte Bibliothek von TSM verwendet. Wenn Sie eine andere Version der gemeinsam benutzten Bibliothek von TSM wünschen, verwenden Sie `SQLU_OTHER_MEDIA`, und geben Sie den Namen der gemeinsam benutzten Bibliothek an.

SQLU_OTHER_MEDIA
Produkt eines anderen Lieferanten. Geben Sie den Namen der gemeinsam benutzten Bibliothek im Feld *locations* an.

SQLU_USER_EXIT
Benutzerexit. Es ist keine zusätzliche Eingabe erforderlich (nur verfügbar, wenn sich der Server unter OS/2 befindet).

Weitere Informationen finden Sie in der Struktur *db2MediaListStruct*.

piUsername
Eingabe. Eine Zeichenfolge, die den Benutzernamen enthält, der beim Versuch einer Verbindung verwendet wird. Kann NULL sein.

iUsernameLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Benutzernamens in Byte angibt. Wird auf null gesetzt, wenn kein Benutzername angegeben wird.

piPassword
Eingabe. Eine Zeichenfolge, die das Kennwort enthält, das mit dem Benutzernamen verwendet werden soll. Kann NULL sein.

iPasswordLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Kennworts in Byte angibt. Wird auf null gesetzt, wenn kein Kennwort angegeben wird.

piVendorOptions
Eingabe. Wird verwendet, um Informationen von der Anwendung an die Funktionen anderer Lieferanten zu übergeben. Diese Datenstruktur muss unstrukturiert sein, d. h., es Zwischenstufen werden nicht unterstützt. Beachten Sie, dass für diese Daten keine Bytefolgeumkehrung durchgeführt und die Codepage nicht überprüft wird.

iVendorOptionsSize
Eingabe. Die Länge des Felds *piVendorOptions*, das nicht länger als 65535 Byte sein darf.

oBackupSize
Ausgabe. Größe des Sicherungsimages (in MB).

iCallerAction
Eingabe. Gibt die auszuführende Aktion an. Gültige Werte (definiert in `db2ApiDf.h`) sind:

DB2BACKUP_BACKUP
Die Sicherung starten.

DB2BACKUP_NOINTERRUPT
Die Sicherung starten. Gibt an, dass die Sicherung nicht überwacht ausgeführt wird und dass Szenarios, die normalerweise Benutzereingriff erfordern, entweder versucht werden, ohne vorher an das aufrufende Programm zurückgegeben zu werden, oder einen Fehler generieren. Verwenden Sie diese Aktion z. B. dann, wenn bekannt ist, dass alle für die Sicherungsoperation erforderlichen Datenträger angehängt wurden, und Dienstprogrammeingabeaufforderungen nicht erwünscht sind.

DB2BACKUP_CONTINUE
Die Sicherung fortsetzen, nachdem der Benutzer die vom Dienstprogramm angeforderten Aktionen ausgeführt hat (z. B. das Anhängen eines neuen Bandes).

DB2BACKUP_TERMINATE
Die Sicherung beenden, nachdem der Benutzer eine vom Dienstprogramm angeforderte Aktion nicht ausführen konnte.

DB2BACKUP_DEVICE_TERMINATE
Eine bestimmte Einheit aus der Liste der vom Sicherungsdienstprogramm verwendeten Einheiten entfernen. Wenn ein bestimmter Datenträger voll ist, gibt das Sicherungsdienstprogramm eine Warnung an das aufrufende Programm zurück (während der Prozess unter Verwendung der übrigen Einheiten fortgesetzt wird). Rufen Sie das Sicherungsdienstprogramm erneut mit dieser Aktion auf, um die Einheit, die die Warnung generiert hat, aus der Liste der verwendeten Einheiten zu entfernen.

DB2BACKUP_PARM_CHK
Parameter auswerten, ohne eine Sicherung auszuführen. Diese Option beendet die Datenbankverbindung nicht, nachdem der Aufruf zurückgegeben wurde. Nach einer erfolgreichen Rückgabe dieses Aufrufs wird erwartet, dass der Benutzer einen Aufruf mit `SQLUB_CONTINUE` absetzt, um mit der Aktion fortzufahren.

DB2BACKUP_PARM_CHK_ONLY
Parameter auswerten, ohne eine Sicherung auszuführen. Bevor dieser Aufruf zurückgegeben wird, wird die durch diesen Aufruf hergestellte Datenbankverbindung beendet, und es ist kein darauf folgender Aufruf erforderlich.

iBufferSize
Eingabe. Sicherungspuffergröße in 4 KB großen Zuordnungseinheiten (Seiten). Die Mindestanzahl ist 8 Einheiten.

iNumBuffers
Eingabe. Gibt die Anzahl der zu verwendenden Sicherungspuffer an. Der Mindestwert ist 2, der Maximalwert wird vom Speicher begrenzt.

iParallelism
Eingabe. Grad der Parallelität (Anzahl Puffermanipulatoren). Der Mindestwert ist 1, der Maximalwert ist 1024.

iOptions
Eingabe. Ein Bitmap der Sicherungsmerkmale. Die Optionen müssen mit dem bitweisen OR-Operator kombiniert werden, um einen Wert für *iOptions* zu erzeugen. Gültige Werte (definiert in `db2ApiDf.h`) sind:

DB2BACKUP_OFFLINE
Eine exklusive Verbindung zur Datenbank herstellen.

DB2BACKUP_ONLINE
Den Datenbankzugriff durch andere Anwendungen ermöglichen, während die Sicherungsoperation ausgeführt wird.

Anmerkung: Eine Onlinesicherungsoperation kann blockiert erscheinen, wenn Benutzer Sperren auf SMS LOB-Daten aufrecht erhalten.

DB2BACKUP_DB
Eine Gesamtsicherung der Datenbank.

DB2BACKUP_TABLESPACE
Eine Sicherung auf Tabellenbereichsebene. Geben Sie für eine Sicherung auf Tabellenbereichsebene im Parameter *piTablespaceList* eine Liste mit Tabellenbereichen an.

DB2BACKUP_INCREMENTAL
Ein kumulatives (inkrementelles) Sicherungsimage. Ein inkrementelles Sicherungsimage ist eine Kopie aller Datenbankdaten, die seit der letzten erfolgreichen vollständigen Sicherungsoperation geändert wurden.

DB2BACKUP_DELTA
Ein nicht kumulatives Sicherungsimage (Deltasicherungsimage). Ein Deltasicherungsimage ist eine Kopie aller Datenbankdaten, die seit der letzten erfolgreichen Sicherungsoperation einer beliebigen Art geändert wurden.

DB2BACKUP_COMPRESS
Gibt an, dass die Sicherung komprimiert werden soll.

DB2BACKUP_INCLUDE_COMPR_LIB
Gibt an, dass die zur Komprimierung des Sicherungsimages verwendete Bibliothek in das Sicherungsimage aufgenommen werden soll.

DB2BACKUP_EXCLUDE_COMPR_LIB
Gibt an, dass die zur Komprimierung des Sicherungsimages verwendete Bibliothek nicht in das Sicherungsimage aufgenommen werden soll.

DB2BACKUP_INCLUDE_LOGS
Gibt an, dass das Sicherungsimage auch den Bereich an Protokolldateien einschließen soll, der zum Wiederherstellen und aktualisierenden Wiederherstellen dieses Images bis zu einem konsistenten Zeitpunkt erforderlich ist. Diese Option gilt nicht für Offlinesicherungen oder Sicherungen, die mehrere Partitionen umfassen.

DB2BACKUP_EXCLUDE_LOGS
Gibt an, dass das Sicherungsimage keine Protokolldateien enthalten soll.

Anmerkung: Wird eine Offlinesicherungsoperation ausgeführt, werden Protokolle unabhängig von dieser Option grundsätzlich ausgeschlossen.

iUtilImpactPriority
Ein Prioritätswert, der während der Sicherung verwendet wird. Der

Prioritätswert kann jede Zahl zwischen 0 und 100 sein, wobei 0 nicht gedrosselt und 100 die höchste Priorität darstellt.

piComprLibrary
Eingabe. Gibt den Namen der externen Bibliothek an, die zur Komprimierung des Sicherungsimages verwendet werden soll. Der Name muss ein vollständig qualifizierter Pfad für eine Datei auf dem Server sein. Wenn der Wert ein Nullzeiger oder ein Zeiger auf eine leere Zeichenfolge ist, verwendet DB2 die Standardkomprimierungsbibliothek. Wenn die angegebene Bibliothek nicht gefunden werden kann, schlägt die Sicherung fehl.

piComprLibraryLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die in Byte die Länge des Namens der in `piComprLibrary` angegebenen Bibliothek angibt. Wenn kein Bibliotheksname angegeben wurde, ist dieser Parameter auf null gesetzt.

piComprOptions
Eingabe. Beschreibt einen Block binärer Daten, der an die Initialisierungsroutine in der Komprimierungsbibliothek übergeben wird. DB2 übergibt diese Zeichenfolge direkt vom Client an den Server, daher muss die Handhabung der Bytefolgeumkehrung oder Codepagekonvertierung von der Komprimierungsbibliothek übernommen werden. Ist '@' das erste Zeichen des Datenblocks, werden die restlichen Daten von DB2 als Name einer Datei interpretiert, die sich auf dem Server befindet. DB2 ersetzt in diesem Fall den Inhalt von `piComprOptions` und `iComprOptionsSize` durch den Inhalt bzw. die Größe dieser Datei und übergibt diesen neuen Wert stattdessen an die Initialisierungsroutine.

iComprOptionsSize
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Größe des als `piComprOptions` übergebenen Datenblocks angibt. `iComprOptionsSize` wird nur dann auf null gesetzt, wenn `piComprOptions` ein Nullzeiger ist.

tablespaces
Ein Zeiger auf die Liste der zu sichernden Tabellenbereiche. Für C besteht diese Liste aus auf Null endenden Zeichenfolgen. Die generische Version ist eine Liste mit *db2Char*-Strukturen.

numTablespaces
Anzahl der Einträge im Parameter *tablespaces*.

locations
Ein Zeiger auf die Liste der Datenträgerpositionen. Für C besteht diese Liste aus auf Null endenden Zeichenfolgen. Die generische Version ist eine Liste mit *db2Char*-Strukturen.

numLocations
Die Anzahl der Einträge im Parameter *locations*.

locationType
Ein Zeichen, das den Datenträgertyp angibt. Gültige Werte (definiert in `sqlutil.h`) sind:

SQLU_LOCAL_MEDIA
Lokale Einheiten (Bänder, Platten, Disketten oder benannte Pipes).

SQLU_TSM_MEDIA
Tivoli Storage Manager.

SQLU_OTHER_MEDIA
Bibliothek eines anderen Herstellers.

SQLU_USER_EXIT
Benutzerexit (nur verfügbar, wenn sich der Server unter OS/2 befindet).

pioData
Ein Zeiger auf den Puffer für Zeichendaten.

iLength
Eingabe. Die Größe des Puffers *pioData*.

oLength
Ausgabe. Zur zukünftigen Verwendung reserviert.

Zugehörige Referenzen:

- „sqlemgdb - Migrate Database" im Handbuch *Administrative API Reference*
- „db2Rollforward - Datenbank aktualisierend wiederherstellen" auf Seite 154
- „SQLCA" im Handbuch *Administrative API Reference*
- „db2Restore - Datenbank wiederherstellen" auf Seite 113

Zugehörige Beispiele:

- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.sqC -- How to recover a database (C++)"

Sicherungssitzungen - Beispiele für CLP

Beispiel 1

Im folgenden Beispiel wird die Datenbank SAMPLE in 2 gleichzeitig ablaufenden TSM-Clientsitzungen auf einem TSM-Server gesichert. Das Sicherungsdienstprogramm wird 4 Puffer verwenden. 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 verfügbarer Prozessoren und der Anzahl zu sichernder 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
```

Beispiel 2

Es folgt ein Beispiel für eine Strategie zur wöchentlichen Teilsicherung für eine wiederherstellbare Datenbank. Diese Strategie umfasst eine wöchentliche vollständige Sicherung der Datenbank, eine tägliche nicht kumulative Sicherung (Deltasicherung) sowie eine kumulative Sicherung (Teilsicherung) 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
```

Beispiel 3

Setzen Sie den folgenden Befehl ab, um in einer Windows-Umgebung eine Sicherungsoperation auf eine Bandeinheit einzuleiten:

```
db2 backup database sample to \\.\tape0
```

Zugehörige Tasks:

- „Verwenden des Sicherungsdienstprogramms" auf Seite 74

Optimieren der Leistung des Sicherungsdienstprogramms

Wenn Sie eine Sicherungsoperation ausführen, wählt DB2® 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. Die Zielsetzung dabei ist, die zum Ausführen einer Sicherungsoperation erforderliche Zeit zu minimieren. Wenn Sie für die folgenden Parameter des Befehls BACKUP DATABASE nicht explizit Werte angeben, wählt DB2 einen Wert für die Parameter aus:

- WITH anzahl-puffer BUFFERS
- PARALLELISM n
- BUFFER puffergröße

Die von den Konfigurationsparametern BACKBUFSZ und RESTBUFSZ des Datenbankmanagers angegebenen Werte werden ignoriert. Wenn Sie diese Werte verwenden wollen, müssen Sie beim Absetzen des Befehls BACKUP DATABASE explizit eine Puffergröße angeben.

Sie können auch eine der folgenden Aktionen ausführen, um die für eine Sicherungsoperation erforderliche Zeit zu reduzieren:

- Geben Sie eine Tabellenbereichssicherung 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 komprimierte Sicherung gestartet werden. Jeder Prozess oder Thread ist einem bestimmten Tabellenbereich zugeordnet, daher hat es keinen Nutzen, einen Wert für den Parameter PARALLELISM anzugeben, der größer als die Anzahl zu sichernder Tabellenbereiche ist. Nach der Sicherung 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 Sicherungspuffergröße.

  Die ideale Puffergröße für eine Sicherung ist ein Vielfaches der Speicherbereichsgröße 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 Sicherungsziele (oder Sitzungen), um sicherzustellen, dass die Sicherungszieleinheiten nicht auf Daten warten müssen.
- Verwenden Sie mehrere Zieleinheiten.

Zugehörige Konzepte:

- „Sicherung - Übersicht" auf Seite 71

Zugehörige Tasks:

- „Verwenden des Sicherungsdienstprogramms" auf Seite 74

Kapitel 3. Datenbankwiederherstellung

In diesem Abschnitt wird das DB2 UDB-Wiederherstellungsdienstprogramm beschrieben, mit dem zuvor gesicherte, beschädigte Datenbanken und Tabellenbereiche wiederhergestellt werden können.

Die folgenden Themen werden behandelt:

Wiederherstellung - Übersicht

In der einfachsten Form des DB2®-Befehls RESTORE DATABASE müssen Sie lediglich den Aliasnamen der Datenbank angeben, die Sie wiederherstellen wollen. Beispiel:

```
db2 restore db sample
```

Da die Datenbank SAMPLE vorhanden ist und beim Absetzen des Befehls RESTORE DATABASE ersetzt werden wird, wird in diesem Beispiel die folgende Nachricht zurückgegeben:

```
SQL2539W  Achtung!  Wiederherstellung in eine bestehende Datenbank, die mit der
Datenbank des Sicherungsimages identisch ist. Die Datenbankdateien werden gelöscht.
Fortfahren ? (j/n)
```

Wenn Sie `j` angeben, sollte die Wiederherstellungsoperation erfolgreich beendet werden können.

Für die Wiederherstellung einer Datenbank ist eine exklusive Verbindung erforderlich, d. h., es können keine Anwendungen für die Datenbank ausgeführt werden, sobald die Operation gestartet ist. Das Wiederherstellungsdienstprogramm verhindert, dass andere Anwendungen vor der erfolgreichen Beendigung der Operation auf die Datenbank zugreifen. Die Wiederherstellung eines Tabellenbereichs kann hingegen online erfolgen.

Ein Tabellenbereich kann erst nach der erfolgreichen Beendigung der Wiederherstellungsoperation (und anschließender aktualisierender Wiederherstellung) 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 Wiederherstellungsoperation können Sie ein Sicherungsimage auf Tabellenbereichsebene oder ein Image einer vollständigen Datenbanksicherung verwenden, aus dem dem Sie mindestens einen Tabellenbereich auswählen. Alle Protokolldateien, die diesen Tabellenbereichen ab dem Zeitpunkt der Erstellung des Sicherungsimages zugeordnet wurden, müssen vorhanden sein.

Zugehörige Konzepte:

- „Einschließen von Protokolldateien in ein Sicherungsimage" auf Seite 60

Erforderliche Zugriffsrechte und Berechtigungen zur Verwendung des Wiederherstellungsdienstprogramms

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf sie zuzugreifen. Berechtigungsstufen stellen eine Methode dar, Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zu gruppieren. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte. Benutzer können nur auf solche Objekte zugreifen, für die sie die entsprechende Berechtigung besitzen, d. h., für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Sie benötigen die Berechtigung SYSADM, SYSCTRL oder SYSMAINT, um eine *vorhandene* Datenbank mit Hilfe einer Sicherungskopie der gesamten Datenbank wiederherstellen zu können. Für das Wiederherstellen in eine *neue* Datenbank ist die Berechtigung SYSADM oder SYSCTRL erforderlich.

Verwenden des Wiederherstellungsprogramms

Vorbedingungen:

Bei der Wiederherstellung in eine *vorhandene* Datenbank sollte noch keine Verbindung zu der wiederherzustellenden Datenbank bestehen. Das Wiederherstellungsdienstprogramm stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der Wiederherstellungsoperation beendet wird. Bei der Wiederherstellung in eine *neue* Datenbank ist zum Erstellen der Datenbank die Herstellung einer Verbindung zu einem Exemplar erforderlich. Bei der Wiederherstellung in eine *neue ferne* Datenbank müssen Sie zuerst eine Verbindung zu dem Exemplar herstellen, auf dem sich die neue Datenbank befinden soll. Erstellen Sie anschließend die neue Datenbank, und geben Sie dabei die Codepage und das Gebiet des Servers an.

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln.

Einschränkungen:

Für das Wiederherstellungsdienstprogramm gelten die folgenden Einschränkungen:

- Das Wiederherstellungsdienstprogramm kann nur verwendet werden, wenn die betreffende Datenbank zuvor mit dem DB2-Sicherungsdienstprogramm gesichert wurde.
- Es kann keine Wiederherstellungsoperation gestartet werden, wenn ein Prozess zur aktualisierenden Wiederherstellung aktiv ist.
- Ein Tabellenbereich kann nur dann wiederhergestellt werden, wenn der betreffende Tabellenbereich existiert und mit dem wiederherzustellenden identisch ist; „identisch" bedeutet, dass der Tabellenbereich nicht zwischen der Sicherungs- und der Wiederherstellungsoperation gelöscht und erneut erstellt wurde.
- Es ist nicht möglich, die Sicherungskopie eines Tabellenbereichs in eine neue Datenbank wiederherzustellen.
- Für die Systemkatalogtabellen ist eine Onlinewiederherstellung der Tabellenbereiche nicht möglich.

Prozedur:

Das Wiederherstellungsdienstprogramm kann über den Befehlszeilenprozessor (CLP), das Notizbuch bzw. den Assistenten **Datenbank wiederherstellen** in der Steuerzentrale oder die Anwendungsprogrammierschnittstelle **db2Restore** aufgerufen werden.

Es folgt ein Beispiel für den Befehl RESTORE DATABASE, der über den CLP abgesetzt wird:

```
db2 restore db sample from D:\DB2Backups taken at 20010320122644
```

Gehen Sie wie folgt vor, um das Notizbuch bzw. den Assistenten **Datenbank wiederherstellen** zu öffnen:

1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis Sie den Ordner **Datenbanken** finden.
2. Klicken Sie den Ordner **Datenbanken** an. Alle vorhandenen Datenbanken werden auf der rechten Seite des Fensters, im Inhaltsteilfenster, angezeigt.
3. Klicken Sie im Inhaltsteilfenster die gewünschte Datenbank mit Maustaste 2 an, und wählen Sie **Datenbank wiederherstellen** oder **Wiederherstellen —> Datenbank mit Assistent** im Kontextmenü aus. Das Notizbuch bzw. der Assistent **Datenbank wiederherstellen** wird geöffnet.

Zusatzinformationen bietet die Onlinehilfefunktion der Steuerzentrale.

Zugehörige Konzepte:

- „Administrative APIs in Embedded SQL or DB2 CLI Programs" im Handbuch *Application Development Guide: Programming Client Applications*
- „Einführung der Plug-in-Architektur für die Steuerzentrale" im Handbuch *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „db2Restore - Datenbank wiederherstellen" auf Seite 113

Verwenden der Teilwiederherstellung in einer Test- und Produktionsumgebung

Wenn eine Produktionsdatenbank für Teilsicherung und Wiederherstellung aktiviert wurde, können Sie ein Teilsicherungs- oder Deltasicherungsimage verwenden, um eine Testdatenbank zu erstellen oder zu aktualisieren. Sie können hierzu entweder manuelle oder automatische Teilwiederherstellung verwenden. Zum Wiederherstellen des Sicherungsimages 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 Sicherungsimages:

```
backup db prod
Sicherung erfolgreich. Die Zeitmarke für diese Sicherungsimagedatei ist: <zm1>

backup db prod incremental
Sicherung erfolgreich. Die Zeitmarke für diese Sicherungsimagedatei ist: <zm2>
```

wäre dies ein Beispiel für eine manuelle Teilwiederherstellung:

```
restore db prod incremental taken at <ts2> into test without
prompting
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.

restore db prod incremental taken at <ts1> into test without
prompting
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.

restore db prod incremental taken at <ts2> into test without
prompting
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Falls die Datenbank TEST bereits vorhanden ist, wird die Wiederherstellungsoperation alle bereits vorhandenen Daten überschreiben. Wenn die Datenbank TEST noch nicht vorhanden ist, wird das Wiederherstellungsdienstprogramm die Datenbank erstellen und anschließend mit den Daten aus den Sicherungsimages füllen.

Da automatische Teilwiederherstellungsoperationen vom Datenbankprotokoll abhängen, werden die Wiederherstellungsschritte leicht abweichen, je nach dem, ob die Testdatenbank vorhanden ist oder nicht. Wenn eine automatische Teilwiederherstellung in die Datenbank TEST ausgeführt werden soll, muss deren Protokoll das Sicherungsimageprotokoll für die Datenbank PROD enthalten. Das Datenbankprotokoll für das Sicherungsimage ersetzt alle Datenbankprotokolle, die bereits für die Datenbank TEST vorhanden sind, wenn:

- die Datenbank TEST beim Absetzen des Befehls RESTORE DATABASE nicht vorhanden ist oder
- die Datenbank TEST beim Absetzen des Befehls RESTORE DATABASE bereits vorhanden ist und die Datenbank TEST keine Datensätze enthält.

Das folgende Beispiel zeigt eine automatische Teilwiederherstellung 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 Wiederherstellungsdienstprogramm wird die Datenbank TEST erstellen und mit Daten füllen.

Wenn die Datenbank TEST vorhanden ist und das Datenbankprotokoll Einträge enthält, müssen Sie die Datenbank vor der automatischen Teilwiederherstellung 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" bei der Wiederherstellung der Datenbank im Modus 'No Interrupt'
festgestellt.
```

In diesem Fall hat der Befehl RESTORE DATABASE dieselbe Auswirkung, als 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 die automatische Teilwiederherstellung ausgeführt wird:

```
restore db prod incremental automatic taken at <zm2>  into test without
prompting
SQL2540W  RESTORE wurde erfolgreich ausgeführt, es wurde jedoch eine Warnung
"2539" bei der Wiederherstellung der Datenbank im Modus 'No Interrupt'
festgestellt.
```

Sie können weiter Teil- oder Deltasicherungen der Testdatenbank erstellen, ohne zuvor eine Gesamtsicherung der Datenbank vorzunehmen. Wenn es später jedoch erforderlich wird, eines der Teil- oder Deltasicherungsimages wiederzuherstellen, müssen Sie eine manuelle Teilwiederherstellung durchführen. Dies liegt daran, dass Operationen zur automatischen Teilwiederherstellung erfordern, dass alle während einer automatischen Teilwiederherstellung wiederhergestellten Sicherungsimages aus demselben Datenbankaliasnamen erstellt sind.

Wenn Sie eine Datenbankgesamtsicherung der Testdatenbank erstellen, nachdem Sie die Wiederherstellungsoperation unter Verwendung des Produktionssicherungsimages beendet haben, können Sie Teil- oder Deltasicherungen erstellen und diese entweder manuell oder im automatischen Modus wiederherstellen.

Zugehörige Konzepte:

• „Teilsicherung und Teilwiederherstellung” auf Seite 30

Zugehörige Referenzen:

- „BACKUP DATABASE" auf Seite 79
- „RESTORE DATABASE" auf Seite 102
- „LIST HISTORY" auf Seite 294

Umdefinieren von Tabellenbereichsbehältern während einer Wiederherstellungsoperation (umgeleitete Wiederherstellung)

Während der Sicherung einer Datenbank werden alle Tabellenbereichsbehälter protokolliert, die von den hierbei gesicherten Tabellenbereichen verwendet werden. Während einer Wiederherstellungsoperation wird überprüft, ob alle bei der Sicherung aufgelisteten Behälter weiterhin existieren und zugriffsbereit sind. Wenn aufgrund eines Datenträgerfehlers (oder aus einem anderen Grund) mindestens ein Behälter nicht zugriffsbereit ist, schlägt die Wiederherstellung fehl. In diesem Fall ist für eine erfolgreiche Wiederherstellungsoperation eine Umleitung in andere Behälter erforderlich. DB2® bietet Unterstützung für das Hinzufügen, Ändern oder Entfernen von Tabellenbereichsbehältern.

Sie können Tabellenbereichsbehälter erneut definieren, indem Sie den Befehl RESTORE DATABASE aufrufen und den Parameter REDIRECT angeben, oder indem Sie die Seite **Behälter** des Notizbuchs **Datenbank wiederherstellen** in der Steuerzentrale verwenden. Der Prozess des Aufrufens einer umgeleiteten Wiederherstellung eines Teilsicherungsimages ist dem Prozess für ein Nicht-Teilsicherungsimage ähnlich: Rufen Sie den Befehl RESTORE DATABASE mit dem Parameter REDIRECT auf, und geben Sie das Sicherungsimage an, aus dem die Teilwiederherstellung der Datenbank erfolgen soll.

Während einer umgeleiteten Wiederherstellungsoperation werden Verzeichnis- und Dateibehälter automatisch erstellt, sofern sie nicht bereits vorhanden sind. Einheitenbehälter werden nicht automatisch vom Datenbankmanager erstellt.

Die Umleitung von Behältern bietet eine größere Flexibilität bei der Verwaltung von Tabellenbereichsbehältern. So wird beispielsweise das Hinzufügen von Behältern zu SMS-Tabellenbereichen zwar nicht unterstützt, Sie könnten jedoch zu diesem Zweck bei einer umgeleiteten Wiederherstellung einen zusätzlichen Behälter angeben.

Das folgende Beispiel zeigt, wie für die Datenbank SAMPLE eine umgeleitete Wiederherstellung ausgeführt wird:

```
db2 restore db sample redirect without prompting
SQL1277N Beim Wiederherstellen wurde festgestellt, dass auf einen oder mehrere
Behälter nicht zugegriffen werden kann, oder dass dieser/diese den Status
'Speicher muss definiert sein' hat/haben.
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.
```

Zugehörige Referenzen:

- „RESTORE DATABASE" auf Seite 102
- „Wiederherstellungssitzungen - Beispiele für CLP" auf Seite 125

Zugehörige Beispiele:

- „dbrecov.out -- HOW TO RECOVER A DATABASE (C)"
- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.out -- HOW TO RECOVER A DATABASE (C++)"
- „dbrecov.sqC -- How to recover a database (C++)"

Wiederherstellen in eine vorhandene Datenbank

Sie können das Sicherungsimage einer vollständigen Datenbank in eine bereits vorhandene Datenbank wiederherstellen. Das Sicherungsimage kann sich von der vorhandenen Datenbank durch den Aliasnamen, den Datenbanknamen oder die Datenbanknummer unterscheiden.

Eine Datenbank*nummer* ist die eindeutige Kennung einer Datenbank, die sich während der Bestehens der Datenbank nicht ändert. Diese Nummer wird beim Erstellen der Datenbank vom Datenbankmanager zugeordnet. DB2® verwendet immer die Datenbanknummer aus dem Sicherungsimage.

Beim Wiederherstellen in eine vorhandene Datenbank führt das Wiederherstellungsdienstprogramm Folgendes aus:

- Löschen der Tabellen-, Index- und Langfelddaten der vorhandenen Datenbank und Ersetzen dieser Informationen durch die Daten des Sicherungsimages.
- Ersetzen der Tabelleneinträge der wiederherzustellenden Tabellenbereiche.
- Beibehalten der Datei des Wiederherstellungsprotokolls, sofern sie nicht beschädigt oder leer ist. Wenn die Datei des Wiederherstellungsprotokolls beschädigt ist oder keine Einträge enthält, kopiert der Datenbankmanager die Datei aus dem Sicherungsimage.
- Beibehalten des Authentifizierungstyps für die vorhandene Datenbank.
- Beibehalten der Datenbankverzeichnisse der vorhandenen Datenbank. Die Verzeichnisse definieren die Speicherposition und die Art der Katalogisierung der Datenbank.
- Vergleichen der Datenbanknummern. Bei unterschiedlichen Datenbanknummern:
  - Löschen der Protokolle, die der vorhandenen Datenbank zugeordnet sind.
  - Kopieren der Datenbankkonfigurationsdatei aus dem Sicherungsimage.
  - Setzen von NEWLOGPATH auf den Wert des Datenbankkonfigurationsparameters *logpath*, falls NEWLOGPATH im Befehl RESTORE DATABASE angegeben wurde.

  Bei identischen Datenbanknummern:
  - Löschen der Protokolle, wenn es sich um ein Image einer nicht wiederherstellbaren Datenbank handelt.
  - Beibehalten der aktuellen Konfigurationsdatei der Datenbank, sofern diese nicht beschädigt ist. In diesem Fall wird diese Datei aus dem Sicherungsimage kopiert.
  - Setzen von NEWLOGPATH auf den Wert des Datenbankkonfigurationsparameters *logpath*, falls NEWLOGPATH im Befehl RESTORE DATABASE angegeben wurde. Andernfalls wird der aktuelle Protokollpfad in die Datenbankkonfigurationsdatei kopiert. 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 in eine neue Datenbank

Sie können eine neue Datenbank erstellen und anschließend das Sicherungsimage einer vollständigen Datenbank in diese neue Datenbank wiederherstellen. Wenn Sie keine neue Datenbank erstellen, erstellt das Wiederherstellungsdienstprogramm eine neue Datenbank.

Beim Wiederherstellen in eine neue Datenbank führt das Wiederherstellungsdienstprogramm 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 Wiederherstellungsdienstprogramm eine Datenbank mit einem Aliasnamen, der im Parameter für den Aliasnamen der Quellendatenbank angegeben wurde.)
- Wiederherstellen der Konfigurationsdatei der Datenbank aus dem Sicherungsimage.
- 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 Sicherungsimage.
- Wiederherstellen des Inhalts aus den Datenbankverzeichnissen im Sicherungsimage.
- Wiederherstellen der Datei des Wiederherstellungsprotokolls für die Datenbank.

RESTORE DATABASE

Stellt eine beschädigte Datenbank wiederher, die mit dem Sicherungsdienstprogramm gesichert wurde. Die wiederhergestellte Datenbank befindet sich in demselben Status, in dem sie sich bei der Erstellung der Sicherungskopie befand. Dieses Dienstprogramm ermöglicht neben der Wiederherstellung in eine neue Datenbank auch das Überschreiben einer Datenbank mit einem anderen Image.

Sie können Datenbanken, die mit DB2 Version 8 auf 32-Bit-Windows-Plattformen erstellt wurden, in DB2 Version 8 auf 64-Bit-Windows-Plattformen wiederherstellen und umgekehrt. Sie können Datenbanken, die mit DB2 Version 8 auf 32-Bit-Linux-Plattformen (Intel) erstellt wurden, in DB2 Version 8 auf 64-Bit-Linux-Plattformen (Intel) wiederherstellen und umgekehrt. Sie können Datenbanken, die mit DB2 Version 8 unter AIX, HP-UX oder in der Solaris-Betriebsumgebung (32 Bit oder 64 Bit) erstellt wurden, in DB2 Version 8 unter AIX, HP-UX oder in der Solaris-Betriebsumgebung (32 Bit oder 64 Bit) wiederherstellen.

Mit dem Wiederherstellungsdienstprogramm können auch Sicherungsimages wiederhergestellt werden, die mit einer früheren Version von DB2 (maximal zwei Versionen früher) erstellt wurden, sofern die Wortgröße (32 Bit oder 64 Bit) identisch ist. Plattformübergreifende Wiederherstellungsoperationen aus Sicherungsimages, die mit einer früheren DB2-Version erstellt wurden, werden nicht unterstützt. Falls eine Migration erforderlich ist, wird sie nach Beendigung der Wiederherstellungsoperation automatisch aufgerufen.

Falls die Datenbank zum Zeitpunkt der Sicherungsoperation für die aktualisierende Wiederherstellung konfiguriert war, kann die Datenbank in den Status ver-

setzt werden, in dem sie sich vor dem Auftritt der Beschädigung befand. Rufen Sie dazu nach erfolgreicher Ausführung einer Wiederherstellungsoperation das Dienstprogramm zur aktualisierenden Wiederherstellung auf.

Mit diesem Dienstprogramm sind auch Wiederherstellungen von Sicherungen auf Tabellenbereichsebene möglich.

Teil- und Deltaimages können jeweils nicht auf anderen Betriebssystemen oder mit anderer Maschinenwortgröße (32 Bit oder 64 Bit) wiederhergestellt werden.

Im Anschluss an eine erfolgreiche Wiederherstellung aus einer Umgebung in eine andere Umgebung muss zuerst eine nicht inkrementelle Sicherung erstellt werden, bevor Teil- oder Deltasicherungen erstellt werden dürfen. (Diese Einschränkung gilt nicht bei Wiederherstellungen in dieselbe Umgebung.)

Auch bei erfolgreicher Wiederherstellung aus einer Umgebung in eine andere Umgebung sind einige Punkte zu beachten: Pakete müssen vor ihrer Verwendung erneut gebunden werden (mit dem Befehl BIND, dem Befehl REBIND oder dem Dienstprogramm `db2rbind`), SQL-Prozeduren müssen gelöscht und erneut erstellt werden, und alle externen Bibliotheken müssen auf der neuen Plattform erneut erstellt werden. (Diese Aktionen müssen bei Wiederherstellungen in dieselbe Umgebung nicht ausgeführt werden.)

Geltungsbereich:

Dieser Befehl wirkt sich nur auf den Knoten aus, auf dem er ausgeführt wird.

Berechtigung:

Zur Wiederherstellung in eine vorhandene Datenbank eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Zur Wiederherstellung in eine neue Datenbank eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*

Erforderliche Verbindung:

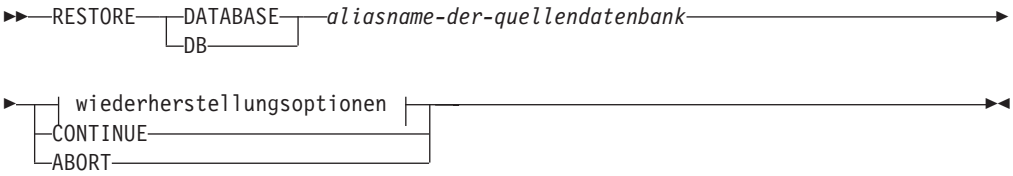
Die erforderliche Verbindung wird je nach dem Typ der Wiederherstellung, die Sie ausführen wollen, variieren:

- Zur Wiederherstellung in eine vorhandene Datenbank ist eine Datenbankverbindung erforderlich. Mit diesem Befehl stellen Sie automatisch eine exklusive Verbindung zur angegebenen Datenbank her.
- Zur Wiederherstellung in eine neue Datenbank ist eine Exemplar- und eine Datenbankverbindung erforderlich. Die Verbindung zu einem Exemplar ist erforderlich, um die Datenbank zu erstellen.

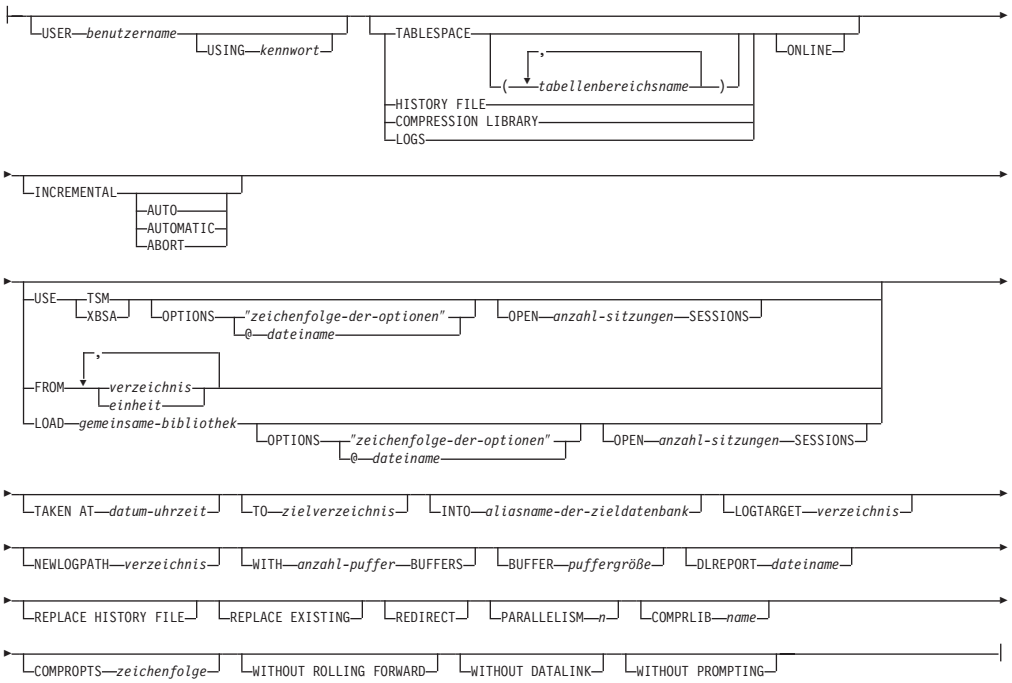
  Wenn Sie eine Wiederherstellung in eine neue Datenbank auf einem Exemplar ausführen, das nicht mit dem aktuellen Exemplar identisch ist, müssen Sie zuerst eine Verbindung zu dem Exemplar herstellen, auf dem sich die neue Datenbank befinden wird. Bei dem neuen Exemplar kann es sich um ein lokales

oder ein fernes Exemplar handeln. Das aktuelle Exemplar wird vom Wert der Umgebungsvariable DB2INSTANCE definiert.

Befehlssyntax:



Wiederherstellungsoptionen:



Befehlsparameter:

DATABASE *aliasname-der-quellendatenbank*
Aliasname der Quellendatenbank, aus der die Sicherung erstellt wurde.

CONTINUE
Gibt an, dass die Behälter umdefiniert wurden und dass der letzte Schritt in einer umgeleiteten Wiederherstellungsoperation ausgeführt werden sollte.

ABORT
Dieser Parameter hat folgende Funktion:

- Er stoppt eine umgeleitete Wiederherstellungsoperation. Dies ist nützlich, wenn ein Fehler aufgetreten ist, der die Wiederholung mindestens eines Schrittes erfordert. Nachdem RESTORE DATABASE mit der Option ABORT abgesetzt wurde, muss jeder Schritt einer umgeleiteten Wiederherstellungsoperation wiederholt werden, einschließlich RESTORE DATABASE mit der Option REDIRECT.
- Er beendet eine Teilwiederherstellungsoperation vor der Fertigstellung.

USER *benutzername*
Gibt den Benutzernamen an, unter dem die Datenbank wiederhergestellt werden soll.

USING *kennwort*
Das Kennwort, das verwendet wird, um den Benutzernamen zu authentifizieren. Wenn kein Kennwort angegeben wird, wird der Benutzer zur Eingabe des Kennworts aufgefordert.

TABLESPACE *tabellenbereichsname*
Eine Namensliste, mit der die wiederherzustellenden Tabellenbereiche angegeben werden.

ONLINE
Dieses Schlüsselwort ist nur gültig, wenn Sie eine Wiederherstellungsoperation auf Tabellenbereichsebene ausführen. Es wird angegeben, damit ein Sicherungsimage online wiederhergestellt werden kann. Das heißt, dass andere Agenten eine Verbindung zur Datenbank herstellen können, während das Sicherungsimage wiederhergestellt wird, und dass die Daten in anderen Tabellenbereichen verfügbar sind, während die angegebenen Tabellenbereiche wiederhergestellt werden.

HISTORY FILE
Dieses Schlüsselwort wird angegeben, damit nur die Protokolldatei aus dem Sicherungsimage wiederhergestellt wird.

COMPRESSION LIBRARY
Dieses Schlüsselwort wird angegeben, damit nur die Komprimierungsbibliothek aus dem Sicherungsimage wiederhergestellt wird. Wenn das Objekt im Sicherungsimage vorhanden ist, wird es im Datenbankverzeichnis wiederhergestellt. Wenn das Objekt im Sicherungsimage nicht vorhanden ist, schlägt die Wiederherstellungsoperation fehl.

LOGS Dieses Schlüsselwort wird angegeben, damit nur die im Sicherungsimage vorhandene Gruppe von Protokolldateien wiederhergestellt wird. Wenn das Sicherungsimage keine Protokolldateien enthält, schlägt die Wiederherstellungsoperation fehl. Wird diese Option angegeben, muss auch die Option LOGTARGET angegeben werden.

INCREMENTAL
Ohne zusätzliche Parameter gibt INCREMENTAL eine manuelle, kumulative Wiederherstellungsoperation an. Bei der manuellen Wiederherstellung muss der Benutzer für alle beteiligten Images jeden Wiederherstellungsbefehl manuell absetzen. Gehen Sie dazu in folgender Reihenfolge vor: letzes, erstes, zweites, drittes usw. bis einschließlich des letzten Images.

INCREMENTAL AUTOMATIC/AUTO
Gibt eine automatische, kumulative Wiederherstellungsoperation an.

INCREMENTAL ABORT
Gibt den Abbruch einer laufenden manuellen, kumulativen Wiederherstellungsoperation an.

USE TSM
Gibt an, dass die Datenbank aus einer von TSM verwalteten Ausgabe wiederhergestellt werden soll.

OPTIONS

"zeichenfolge-der-optionen"
Gibt die Optionen an, die für die Wiederherstellungsoperation verwendet werden sollen. Die Zeichenfolge wird an die Unter-

stützungsbibliothek des Herstellers (z. B. TSM) genau so, wie sie eingegeben wurde und ohne Anführungszeichen übermittelt.

Anmerkung: Wird diese Option angegeben, wird der mit dem Datenbankkonfigurationsparameter VENDOROPT angegebene Wert außer Kraft gesetzt.

@dateiname
Gibt an, dass die Optionen, die für die Wiederherstellungsoperation verwendet werden, in einer Datei enthalten sind, die sich auf dem DB2-Server befindet. Die Zeichenfolge wird an die Unterstützungsbibliothek des Herstellers (z. B. TSM) übermittelt. Die Datei muss ein vollständig qualifizierter Dateiname sein.

OPEN *anzahl-sitzungen* **SESSIONS**
Gibt die Anzahl E/A-Sitzungen an, die mit TSM oder einem Programm eines anderen Lieferanten verwendet werden sollen.

USE XBSA
Gibt an, dass die XBSA-Schnittstelle verwendet werden soll. XBSA (Backup Services APIs) ist eine offene Anwendungsprogrammierschnittstelle für Anwendungen oder Tools, die eine Datenspeicherverwaltung für Sicherungs- oder Archivierungszwecke benötigen.

FROM *verzeichnis/einheit*
Der vollständig qualifizierte Pfad des Verzeichnisses oder der Einheit mit dem Sicherungsimage. Wenn USE TSM, FROM und LOAD nicht angegeben werden, ist der Standardwert das aktuelle Arbeitsverzeichnis der Clientmaschine. Dieses Zielverzeichnis oder diese Einheit muss auf dem Datenbankserver vorhanden sein.

Unter Windows-Betriebssystemen darf das angegebene Verzeichnis kein Verzeichnis sein, das von DB2 generiert worden ist. Wenn z. B. die folgenden Befehle abgesetzt wurden:

```
db2 backup database sample to c:\backup
db2 restore database sample from c:\backup
```

Wenn Sie diese Befehle verwenden, generiert DB2 im Verzeichnis `c:\backup` Unterverzeichnisse, um zu ermöglichen, dass mehrere Sicherungen in das angegebene Verzeichnis der höchsten Ebene gestellt werden können. Die von DB2 generierten Unterverzeichnisse sollten ignoriert werden. Verwenden Sie den Parameter TAKEN AT, um genau anzugeben, welches Sicherungsimage wiederhergestellt werden soll. Im selben Pfad könnten mehrere Sicherungsimages gespeichert sein.

Wenn mehrere Elemente angegeben werden und das letzte Element eine Bandeinheit ist, wird der Benutzer nach einem weiteren Band gefragt. Gültige Antwortoptionen sind:

c Fortsetzen. Verwendung der Einheit fortsetzen, die die Warnung generiert hat (zum Beispiel, wenn ein neues Band eingelegt wurde).

d Einheit beenden. Nur die Verwendung der Einheit beenden, die die Warnung generiert hat (zum Beispiel Beenden, wenn keine Bänder mehr vorhanden sind).

t Beenden. Die Wiederherstellungsoperation abbrechen, nachdem der Benutzer eine vom Dienstprogramm angeforderte Aktion nicht ausführen konnte.

LOAD *gemeinsame-bibliothek*
Der Name der gemeinsam benutzten Bibliothek (DLL unter Windows-Betriebssystemen), die die Sicherungs- und Wiederherstellungs-E/A-Funktionen des Lieferanten enthält, die verwendet werden sollen. Der Name kann einen vollständigen Pfad enthalten. Wenn kein vollständiger Pfad angegeben wird, wird standardmäßig der Pfad verwendet, in dem sich das Benutzerexitprogramm befindet.

TAKEN AT *datum-uhrzeit*
Die Zeitmarke des Datenbanksicherungsimages. Die Zeitmarke wird nach erfolgreicher Fertigstellung der Sicherungsoperation angezeigt und ist ein Teil des Pfadnamens des Sicherungsimages. Sie wird in der Form *jjjjmmtthhmmss* angegeben. Sie können auch einen Teil einer Zeitmarke angeben. Wenn z. B. zwei unterschiedliche Sicherungsimages mit den Zeitmarken 20021001010101 und 20021002010101 vorhanden sind, wird nach Eingabe von `20021002` das Image mit der Zeitmarke `20021002010101` verwendet. Wenn für diesen Parameter kein Wert angegeben wird, darf sich nur ein einziges Sicherungsimage auf dem Quellendatenträger befinden.

TO *zielverzeichnis*
Das Zieldatenbankverzeichnis. Dieser Parameter wird ignoriert, wenn das Dienstprogramm in eine vorhandene Datenbank wiederherstellt. Sie müssen ein lokales Laufwerk und Verzeichnis angeben.

Anmerkung: Geben Sie bei Verwendung dieses Parameters unter Windows-Betriebssystemen nur den Laufwerkbuchstaben an. Wenn Sie einen Pfad eingeben, wird ein Fehler zurückgegeben.

INTO *aliasname-der-zieldatenbank*
Der Aliasname der Zieldatenbank. Wenn die Zieldatenbank nicht vorhanden ist, wird sie erstellt.

Wenn Sie eine Datenbanksicherung in eine vorhandene Datenbank wiederherstellen, übernimmt die wiederhergestellte Datenbank den Aliasnamen und den Datenbankname der vorhandenen Datenbank. Bei einer Wiederherstellung in eine nicht vorhandene Datenbank wird die neue Datenbank mit dem von Ihnen angegebenen Aliasnamen und Datenbanknamen erstellt. Dieser neue Datenbankname muss für das System, auf dem Sie die Datenbank wiederherstellen, eindeutig sein.

LOGTARGET *verzeichnis*
Der absolute Pfadname eines vorhandenen Verzeichnisses auf dem Datenbankserver, das als Zielverzeichnis zum Extrahieren von Protokolldateien aus einem Sicherungsimage verwendet werden soll. Wenn diese Option angegeben wird, werden die im Sicherungsimage vorhandenen Protokolldateien in das Zielverzeichnis extrahiert. Wird diese Option nicht angegeben, werden die in einem Sicherungsimage vorhandenen Protokolldateien nicht extrahiert. Wenn Sie nur die Protokolldateien aus einem Sicherungsimage extrahieren wollen, verwenden Sie die Option LOGS.

NEWLOGPATH *verzeichnis*
Der absolute Pfadname eines Verzeichnisses, das nach der Wiederherstellungsoperation für aktive Protokolldateien verwendet wird. Dieser Parameter hat dieselbe Funktion wie der Datenbankkonfigurationsparameter *newlogpath*, mit der Ausnahme, dass seine Auswirkung auf die Wiederherstellungsoperation begrenzt ist, in der er angegeben wird. Der Parameter kann verwendet werden, wenn der Protokollpfad im Sicherungsimage nach der Wiederherstellungsoperation nicht zur Verwen-

dung geeignet ist. Dies ist z. B. der Fall, wenn der Pfad nicht mehr gültig ist oder von einer anderen Datenbank verwendet wird.

WITH *anzahl-puffer* **BUFFERS**
Die Anzahl zu verwendender Puffer. DB2 wählt automatisch einen optimalen Wert für diesen Parameter aus, wenn Sie nicht explizit einen Wert angeben. Sie können eine größere Anzahl Puffer verwenden, um die Leistung zu verbessern, wenn von mehreren Quellen gelesen wird oder wenn der Wert von PARALLELISM erhöht wurde.

BUFFER *puffergröße*
Die Größe in Seiten des für die Wiederherstellungsoperation verwendeten Puffers. DB2 wählt automatisch einen optimalen Wert für diesen Parameter aus, wenn Sie nicht explizit einen Wert angeben. Der Mindestwert für diesen Parameter ist 8 Seiten.

Die Größe der Wiederherstellungspuffer muss eine positive ganze Zahl und ein Vielfaches der Sicherungspuffergröße sein, die bei der Sicherungsoperation angegeben wurde. Wenn eine ungültige Puffergröße angegeben wird, werden Puffer der kleinsten zulässigen Größe zugeordnet.

DLREPORT *dateiname*
Wenn der Dateiname angegeben wird, muss er als absoluter Pfad angegeben werden. Gibt die Dateien aus, deren Verbindung während einer Wiederherstellungsoperation als Ergebnis einer schnellen Abstimmung aufgehoben wird. Diese Option sollte nur verwendet werden, wenn die wiederherzustellende Tabelle einen Spaltentyp DATALINK und verbundene Dateien hat.

REPLACE HISTORY FILE
Gibt an, dass die Wiederherstellungsoperation die Protokolldatei auf der Platte durch die Protokolldatei aus dem Sicherungsimage ersetzen soll.

REPLACE EXISTING
Wenn bereits eine Datenbank mit demselben Aliasnamen wie dem Aliasnamen der Zieldatenbank vorhanden ist, gibt dieser Parameter an, dass das Wiederherstellungsdienstprogramm die vorhandene Datenbank durch die wiederhergestellte Datenbank ersetzen soll. Dies ist bei Prozeduren nützlich, die das Wiederherstellungsdienstprogramm aufrufen, weil der Befehlszeilenprozessor den Benutzer nicht auffordert, das Löschen einer vorhandenen Datenbank zu bestätigen. Wenn der Parameter WITHOUT PROMPTING angegeben wird, ist es nicht erforderlich, dass Sie den Parameter REPLACE EXISTING angeben. In diesem Fall schlägt die Operation allerdings fehl, wenn Ereignisse auftreten, die normalerweise einen Benutzereingriff erfordern.

REDIRECT
Gibt eine umgeleitete Wiederherstellungsoperation an. Diesem Befehl sollte mindestens ein Befehl SET TABLESPACE CONTAINERS und darauf ein Befehl RESTORE DATABASE mit der Option CONTINUE folgen, um eine umgeleitete Wiederherstellungsoperation zu beenden.

Anmerkung: Alle Befehle mit einer einzelnen umgeleiteten Wiederherstellungsoperation müssen über dasselbe Fenster oder dieselbe CLP-Sitzung aufgerufen werden.

WITHOUT ROLLING FORWARD
Gibt an, dass die Datenbank nicht in den Status *Aktualisierende Wiederherstellung anstehend* versetzt werden soll, nachdem sie erfolgreich wiederhergestellt worden ist.

Wenn sich die Datenbank nach einer erfolgreichen Wiederherstellungsoperation im Status *Aktualisierende Wiederherstellung anstehend* befindet, muss der Befehl ROLLFORWARD aufgerufen werden, bevor die Datenbank wieder verwendet werden kann.

Wird diese Option beim Wiederherstellen auf der Grundlage eines Onlinesicherungsimages (Onlinesicherungsabbildes) angegeben, wird Fehler SQL2537N zurückgegeben.

WITHOUT DATALINK
Gibt an, dass Tabellen mit DATALINK-Spalten in den Status DRP (Data-Link Reconcile Pending) versetzt werden sollen und dass keine Abstimmung der verbundenen Dateien ausgeführt werden soll.

PARALLELISM *n*
Gibt die Anzahl Puffermanipulatoren an, die während der Wiederherstellungsoperation erzeugt werden sollen. DB2 wählt automatisch einen optimalen Wert für diesen Parameter aus, wenn Sie nicht explizit einen Wert angeben.

COMPRLIB *name*
Gibt den Namen der Bibliothek an, die zum Ausführen der Dekomprimierung verwendet werden soll. Der Name muss ein vollständig qualifizierter Pfad für eine Datei auf dem Server sein. Wird dieser Parameter nicht angegeben, versucht DB2, die im Image gespeicherte Bibliothek zu verwenden. Wurde die Sicherung nicht komprimiert, wird der Wert dieses Parameters ignoriert. Wenn die angegebene Bibliothek nicht geladen werden kann, schlägt die Wiederherstellung fehl.

COMPROPTS *zeichenfolge*
Beschreibt einen Block binärer Daten, der an die Initialisierungsroutine in der Dekomprimierungsbibliothek übergeben wird. DB2 übergibt diese Zeichenfolge direkt vom Client an den Server, daher muss die Handhabung der Bytefolgeumkehrung oder Codepagekonvertierung von der Dekomprimierungsbibliothek übernommen werden. Ist '@' das erste Zeichen des Datenblocks, werden die restlichen Daten von DB2 als Name einer Datei interpretiert, die sich auf dem Server befindet. DB2 ersetzt in diesem Fall den Inhalt der *zeichenfolge* durch den Inhalt dieser Datei und übergibt diesen neuen Wert stattdessen an die Initialisierungsroutine. Die maximal zulässige Länge für die Zeichenfolge ist 1024 Byte.

WITHOUT PROMPTING
Gibt an, dass die Wiederherstellungsoperation nicht überwacht ausgeführt werden soll. Aktionen, die normalerweise einen Benutzereingriff erfordern, geben eine Fehlernachricht zurück. Wenn Sie eine austauschbare Datenträgereinheit wie z. B. ein Band oder eine Diskette verwenden, wird der Benutzer am Ende der Einheit benachrichtigt, auch wenn Sie diese Option angegeben haben.

Beispiele:

1. Im folgenden Beispiel ist die Datenbank WSDB auf allen 4 Partitionen definiert und von 0 bis 3 nummeriert. Auf den Pfad /dev3/backup kann von allen Partitionen aus zugegriffen werden. Die folgenden Offlinesicherungsimages stehen im Verzeichnis /dev3/backup zur Verfügung:

```
wsdb.0.db2inst1.NODE0000.CATN0000.20020331234149.001
wsdb.0.db2inst1.NODE0001.CATN0000.20020331234427.001
wsdb.0.db2inst1.NODE0002.CATN0000.20020331234828.001
wsdb.0.db2inst1.NODE0003.CATN0000.20020331235235.001
```

Wenn Sie als Erstes die Katalogtabellenpartition wiederherstellen wollen und anschließend alle übrigen Datenbankpartitionen der Datenbank WSDB, die sich im Verzeichnis /dev3/backup befinden, setzen Sie auf einer der Datenbankpartitionen die folgenden Befehle ab:

```
db2_all '<<+0< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234149
  INTO wsdb REPLACE EXISTING'
db2_all '<<+1< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234427
  INTO wsdb REPLACE EXISTING'
db2_all '<<+2< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234828
  INTO wsdb REPLACE EXISTING'
db2_all '<<+3< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331235235
  INTO wsdb REPLACE EXISTING'
```

Das Dienstprogramm db2_all setzt den Wiederherstellungsbefehl für jede angegebene Datenbankpartition ab.

2. Es folgt ein typisches Beispielszenario für die umgeleitete Wiederherstellung einer Datenbank mit dem Aliasnamen MEINEDB:
   a. Setzen Sie einen Befehl RESTORE DATABASE mit der Option REDIRECT ab.

      ```
      db2 restore db meinedb replace existing redirect
      ```

      Nach der erfolgreichen Ausführung von Schritt 1 und vor der Vollendung von Schritt 3 kann die Wiederherstellungsoperation durch Absetzen des folgenden Befehls abgebrochen werden:

      ```
      db2 restore db meinedb abort
      ```

   b. Setzen Sie für jeden Tabellenbereich, dessen Behälter erneut definiert werden müssen, einen Befehl SET TABLESPACE CONTAINERS ab. Beispiel:

      ```
      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 Behältern der wiederhergestellten Datenbank um die in diesem Schritt angegebenen Behälter handelt.

   c. Nach der erfolgreichen Durchführung der Schritte 1 und 2 setzen Sie den folgenden Befehl ab:

      ```
      db2 restore db meinedb continue
      ```

      Dies ist der letzte Schritt der umgeleiteten Wiederherstellung.

   d. Falls Schritt 3 fehlschlägt oder die Wiederherstellungsoperation abgebrochen wurde, kann die umgeleitete Wiederherstellung erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen.
3. Es folgt ein Beispiel für eine Strategie zur wöchentlichen Teilsicherung für eine wiederherstellbare Datenbank. Diese Strategie umfasst eine wöchentliche vollständige Sicherung der Datenbank, eine tägliche nicht kumulative Sicherung (Deltasicherung) sowie eine kumulative Sicherung (inkrementell) in der Wochenmitte:

   ```
   (So) backup db meinedb use tsm
   (Mo) backup db meinedb online incremental delta use tsm
   (Di) backup db meinedb online incremental delta use tsm
   (Mi) backup db meinedb online incremental use tsm
   ```

```
(Do) backup db meinedb online incremental delta use tsm
(Fr) backup db meinedb online incremental delta use tsm
(Sa) backup db meinedb online incremental use tsm
```

Setzen Sie den folgenden Befehl ab, um die Datenbank automatisch aus den Images wiederherzustellen, die am Freitagmorgen erstellt wurden:

```
restore db meinedb incremental automatic taken at (Fr)
```

Setzen Sie den folgenden Befehl ab, um die Datenbank manuell aus den Images wiederherzustellen, die am Freitagmorgen erstellt wurden:

```
restore db meinedb incremental taken at (Fr)
restore db meinedb incremental taken at (So)
restore db meinedb incremental taken at (Mi)
restore db meinedb incremental taken at (Do)
restore db meinedb incremental taken at (Fr)
```

4. Setzen Sie den folgenden Befehl ab, um ein Sicherungsimage, das Protokolldateien enthält, für den Transport an ein fernes System zu erstellen:

   ```
   backup db sample online to /dev3/backup include logs
   ```

   Setzen Sie den folgenden Befehl ab, um dieses Sicherungsimage wiederherzustellen, einen Pfad für LOGTARGET festzulegen und diesen Pfad bei einer aktualisierenden Wiederherstellung anzugeben:

   ```
   restore db sample from /dev3/backup logtarget /dev3/logs
   rollforward db sample to end of logs and stop overflow log path /dev3/logs
   ```

5. Setzen Sie den folgenden Befehl ab, um nur die Protokolldateien aus einem Sicherungsimage wiederherzustellen, das Protokolldateien enthält:

   ```
   restore db sample logs from /dev3/backup logtarget /dev3/logs
   ```

6. Mit den Schlüsselwörtern USE TSM OPTIONS können die TSM-Informationen angegeben werden, die für die Wiederherstellungsoperation verwendet werden sollen. Lassen Sie auf Windows-Plattformen die Option -fromowner weg.

   - Angabe einer begrenzten Zeichenfolge:

     ```
     db2 restore db sample use TSM options "-fromnode bar -fromowner dmcinnis"
     ```

   - Angabe einer vollständig qualifizierten Datei:

     ```
     db2 restore db sample use TSM options @/u/dmcinnis/myoptions.txt
     ```

     Die Datei `myoptions.txt` enthält die folgenden Informationen: -fromnode=bar -fromowner=dmcinnis

Hinweise:

- Ein RESTORE DATABASE-Befehl im Format `db2 restore db <name>` führt eine Wiederherstellung der gesamten Datenbank mit einem Datenbankimage aus sowie eine Wiederherstellung der Tabellenbereiche, die in einem Tabellenbereichsimage gefunden werden. Alle RESTORE DATABASE-Befehle im Format `db2 restore db <name> tablespace` führen eine Wiederherstellung auf Tabellenbereichsebene der im Image gefundenen Tabellenbereiche aus. Alle RESTORE DATABASE-Befehle, in denen eine Liste mit Tabellenbereichen angegeben wird, führen eine Wiederherstellung aller ausdrücklich aufgeführten Tabellenbereiche aus.
- Im Anschluss an die Wiederherstellung einer Onlinesicherung müssen Sie eine aktualisierende Wiederherstellung ausführen.
- Wenn ein Sicherungsimage komprimiert ist, stellt DB2 dies fest und dekomprimiert die Daten automatisch, bevor die Wiederherstellung gestartet wird. Wird mit der Anwendungsprogrammierschnittstelle `db2Restore` eine Bibliothek angegeben, wird diese zur Dekomprimierung der Daten verwendet. Andernfalls wird

eine im Sicherungsimage gespeicherte Bibliothek verwendet. Ist keine Bibliothek vorhanden, können die Daten nicht dekomprimiert werden, und die Wiederherstellung schlägt fehl.

- Wenn die Komprimierungsbibliothek aus einem Sicherungsimage wiederhergestellt wird (entweder explizit durch Angeben des Wiederherstellungstyps DB2RESTORE_COMPR_LIB oder implizit durch Ausführen einer normalen Wiederherstellung einer komprimierten Sicherung), muss die Wiederherstellungsoperation auf derselben Plattform und unter demselben Betriebssystem wie die Sicherung ausgeführt werden. Wenn die Plattform, auf der die Sicherung erstellt wurde, nicht mit der Plattform identisch ist, auf der die Wiederherstellung ausgeführt wird, schlägt die Wiederherstellungsoperation fehl, auch wenn DB2 normalerweise plattformübergreifende Wiederherstellungen für diese beiden Systeme unterstützt.
- Zum Wiederherstellen von Protokolldateien aus dem entsprechenden Sicherungsimage muss über die Option LOGTARGET ein vollständig qualifizierter und gültiger Pfad angegeben werden, der auf dem DB2-Server vorhanden ist. Sind diese Bedingungen erfüllt, schreibt das Wiederherstellungsdienstprogramm die Protokolldateien aus dem Image in den Zielpfad. Wenn die Option LOGTARGET während der Wiederherstellung eines Sicherungsimages angegeben wird, das keine Protokolldateien enthält, gibt die Wiederherstellung einen Fehler zurück, bevor mit der Wiederherstellung von Tabellenbereichsdaten begonnen wird. Eine Wiederherstellung schlägt ebenfalls mit einer Fehlermeldung fehl, wenn ein ungültiger oder schreibgeschützter Pfad für LOGTARGET angegeben wird.
- Sind beim Absetzen des Befehls RESTORE DATABASE bereits Protokolldateien im Pfad für LOGTARGET vorhanden, wird eine Warnung an den Benutzer zurückgegeben. Diese Warnung wird nicht zurückgegeben, wenn WITHOUT PROMPTING angegeben wird.
- Wenn während einer Wiederherstellungsoperation, bei der die Option LOGTARGET angegeben wurde, eine Protokolldatei aus irgendeinem Grund nicht extrahiert werden kann, schlägt die Wiederherstellung fehl, und es wird ein Fehler zurückgegeben. Ist der Name einer Protokolldatei, die aus dem Sicherungsimage extrahiert wird, mit dem Namen einer bereits im Pfad für LOGTARGET vorhandenen Datei identisch, schlägt die Wiederherstellungsoperation fehl, und es wird ein Fehler zurückgegeben. Das Dienstprogramm für die Wiederherstellung von Datenbanken überschreibt bereits im Pfad für LOGTARGET vorhandene Dateien nicht.
- Sie haben auch die Möglichkeit, aus einem Sicherungsimage nur die darin gespeicherten Protokolldateien wiederherzustellen. Geben Sie dazu die Option LOGS zusätzlich zum Pfad für LOGTARGET an. Wenn die Option LOGS ohne einen Pfad für LOGTARGET angegeben wird, führt dies zu einem Fehler. Treten in diesem Betriebsmodus Fehler beim Wiederherstellen von Protokolldateien auf, wird die Wiederherstellungsoperation sofort beendet, und es wird ein Fehler zurückgegeben.
- Bei einer automatischen Teilwiederherstellungsoperation werden nur die Protokolle aus dem Sicherungsimage abgerufen, die sich im Zielimage der Wiederherstellungsoperation befinden. Protokolle, die sich in temporären Images befinden, auf die während des Teilwiederherstellungsprozesses verwiesen wird, werden nicht aus diesen temporären Sicherungsimages extrahiert. Bei einer manuellen Teilwiederherstellungsoperation darf der Pfad für LOGTARGET nur in dem zuletzt abgesetzten Wiederherstellungsbefehl angegeben werden.

Zugehörige Referenzen:

- „BACKUP DATABASE" auf Seite 79

- „ROLLFORWARD DATABASE" auf Seite 144
- „db2move - Database Movement Tool Command" im Handbuch *Command Reference*

db2Restore - Datenbank wiederherstellen

Stellt eine beschädigte Datenbank wiederher, die mit der API "db2Backup - Datenbank sichern" gesichert wurde. Die wiederhergestellte Datenbank befindet sich in demselben Status, in dem sie sich bei der Erstellung der Sicherungskopie befand. Dieses Dienstprogramm ermöglicht neben der Wiederherstellung in eine neue Datenbank auch die Wiederherstellung in eine Datenbank mit einem anderen Namen als dem im Sicherungsimage verwendeten Datenbanknamen.

Mit diesem Dienstprogramm können auch DB2-Datenbanken wiederhergestellt werden, die mit den letzten beiden Releases erstellt wurden.

Mit diesem Dienstprogramm sind auch Wiederherstellungen aus Sicherungen auf Tabellenbereichsebene möglich sowie Wiederherstellungen von Tabellenbereichen aus Datenbanksicherungsimages.

Geltungsbereich:

Diese API wirkt sich nur auf die Datenbankpartition aus, von der aus sie aufgerufen wird.

Berechtigung:

Zur Wiederherstellung in eine vorhandene Datenbank eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Zur Wiederherstellung in eine neue Datenbank eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*

Erforderliche Verbindung:

Zur Wiederherstellung in eine vorhandene Datenbank ist eine Datenbankverbindung erforderlich. Diese API stellt automatisch eine Verbindung zur angegebenen Datenbank her und beendet diese Verbindung, sobald die Wiederherstellungsoperation abgeschlossen ist.

Zur Wiederherstellung in eine neue Datenbank ist eine Exemplar- und eine Datenbankverbindung erforderlich. Die Verbindung zu einem Exemplar ist erforderlich, um die Datenbank zu erstellen.

Wenn Sie eine Wiederherstellung in eine neue Datenbank auf einem Exemplar ausführen, das nicht mit dem aktuellen Exemplar identisch ist (dieses wird von dem Wert der Umgebungsvariablen DB2INSTANCE definiert), müssen Sie zuerst eine Verbindung zu dem Exemplar herstellen, auf dem sich die neue Datenbank befinden wird.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2Restore */
/* ... */
SQL_API_RC SQL_API_FN
db2Restore (
  db2Uint32 versionNumber,
  void         *pDB2RestoreStruct,
  struct sqlca *pSqlca);
/* ... */

typedef SQL_STRUCTURE db2RestoreStruct
{
  char                         *piSourceDBAlias;
  char                         *piTargetDBAlias;
  char                         oApplicationId[SQLU_APPLID_LEN+1];
  char                         *piTimestamp;
  char                         *piTargetDBPath;
  char                         *piReportFile;
  struct db2TablespaceStruct   *piTablespaceList;
  struct db2MediaListStruct    *piMediaList;
  char                         *piUsername;
  char                         *piPassword;
  char                         *piNewLogPath;
  void                         *piVendorOptions;
  db2Uint32                    iVendorOptionsSize;
  db2Uint32                    iParallelism;
  db2Uint32                    iBufferSize;
  db2Uint32                    iNumBuffers;
  db2Uint32                    iCallerAction;
   db2Uint32 iOptions;
  char                         *piComprLibrary;
  void                         *piComprOptions;
  db2Uint32                    iComprOptionsSize;
  char                         *piLogTarget;
} db2RestoreStruct;

typedef SQL_STRUCTURE db2TablespaceStruct
{
  char                         **tablespaces;
  db2Uint32                    numTablespaces;
} db2TablespaceStruct;

typedef SQL_STRUCTURE db2MediaListStruct
{
  char                         **locations;
  db2Uint32                    numLocations;
  char                         locationType;
} db2MediaListStruct;
/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2gRestore */
/* ... */
SQL_API_RC SQL_API_FN
db2gRestore (
  db2Uint32 versionNumber,
  void         *pDB2gRestoreStruct,
  struct sqlca *pSqlca);
```

```
typedef SQL_STRUCTURE db2gRestoreStruct
{
  char                         *piSourceDBAlias;
  db2Uint32                    iSourceDBAliasLen;
  char                         *piTargetDBAlias;
  db2Uint32                    iTargetDBAliasLen;
  char                         *poApplicationId;
  db2Uint32                    iApplicationIdLen;
  char                         *piTimestamp;
  db2Uint32                    iTimestampLen;
  char                         *piTargetDBPath;
  db2Uint32                    iTargetDBPathLen;
  char                         *piReportFile;
  db2Uint32                    iReportFileLen;
  struct db2gTablespaceStruct *piTablespaceList;
  struct db2gMediaListStruct  *piMediaList;
  char                         *piUsername;
  db2Uint32                    iUsernameLen;
  char                         *piPassword;
   db2Uint32 iPasswordLen;
  char                         *piNewLogPath;
  db2Uint32                    iNewLogPathLen;
  void                         *piVendorOptions;
  db2Uint32                    iVendorOptionsSize;
  db2Uint32                    iParallelism;
  db2Uint32                    iBufferSize;
  db2Uint32                    iNumBuffers;
  db2Uint32                    iCallerAction;
   db2Uint32 iOptions;
  char                         *piComprLibrary;
  db2Uint32                    iComprLibraryLen;
  void                         *piComprOptions;
  db2Uint32                    iComprOptionsSize;
  char                         *piLogTarget;
  db2Uint32                    iLogTargetLen;
} db2gRestoreStruct;

typedef SQL_STRUCTURE db2gTablespaceStruct
{
  struct db2Char              *tablespaces;
  db2Uint32                    numTablespaces;
} db2gTablespaceStruct;

typedef SQL_STRUCTURE db2gMediaListStruct
{
  struct db2Char              *locations;
  db2Uint32                    numLocations;
  char                         locationType;
} db2gMediaListStruct;

typedef SQL_STRUCTURE db2Char
{
   char            *pioData;
   db2Uint32       iLength;
   db2Uint32       oLength;
} db2Char;
/* ... */
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2RestoreStruct*, übergeben wird.

pDB2RestoreStruct
Eingabe. Ein Zeiger auf die Struktur *db2RestoreStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

piSourceDBAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der Quellendatenbank des Sicherungsimages enthält.

iSourceDBAliasLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Aliasnamens der Quellendatenbank in Byte angibt.

piTargetDBAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der Zieldatenbank enthält. Wenn dieser Parameter null ist, wird *piSourceDBAlias* verwendet.

iTargetDBAliasLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Aliasnamens der Zieldatenbank in Byte angibt.

oApplicationId
Ausgabe. Die API gibt eine Zeichenfolge zurück, die den Agenten angibt, der die Anwendung bedient. Dieser Parameter kann verwendet werden, um mit dem Datenbankmonitor Informationen zum Fortschritt der Sicherungsoperation zu erhalten.

poApplicationId
Ausgabe. Geben Sie einen Puffer der Länge `SQLU_APPLID_LEN+1` (definiert in `sqlutil.h`) an. Die API gibt eine Zeichenfolge zurück, die den Agenten angibt, der die Anwendung bedient. Dieser Parameter kann verwendet werden, um mit dem Datenbankmonitor Informationen zum Fortschritt der Sicherungsoperation zu erhalten.

iApplicationIdLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Puffers *poApplicationId* in Byte angibt. Diese Zahl sollte gleich `SQLU_APPLID_LEN+1` (definiert in `sqlutil.h`) sein.

piTimestamp
Eingabe. Eine Zeichenfolge, die die Zeitmarke des Sicherungsimages darstellt. Dieses Feld ist optional, wenn auf der angegebenen Quelle nur ein Sicherungsimage vorhanden ist.

iTimestampLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Puffers *piTimestamp* in Byte angibt.

piTargetDBPath
Eingabe. Eine Zeichenfolge, die den relativen oder vollständig qualifizierten Namen des Zieldatenbankverzeichnisses auf dem Server enthält. Wird verwendet, wenn für die wiederhergestellte Sicherung eine neue Datenbank erstellt werden soll; wird andernfalls nicht verwendet.

piReportFile
Eingabe. Wenn der Dateiname angegeben wird, muss er vollständig qualifiziert werden. Die Datalinks-Dateien, deren Verbindung während einer Wiederherstellungsoperation als Ergebnis einer schnellen Abstimmung aufgehoben wird, werden ausgegeben.

iReportFileLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Puffers *piReportFile* in Byte angibt.

piTablespaceList
Eingabe. Eine Liste der wiederherzustellenden Tabellenbereiche. Wird bei der Wiederherstellung einer Untergruppe von Tabellenbereichen aus einem Datenbank- oder Tabellenbereichssicherungsimage verwendet. Siehe Struktur *db2TablespaceStruct*. Es gelten die folgenden Einschränkungen:

- Die Datenbank muss wiederherstellbar sein, d. h. die Parameter *logretain* oder *userexit* müssen aktiviert sein.
- Die Datenbank, in die wiederhergestellt wird, muss dieselbe Datenbank sein, die zur Erstellung des Sicherungsimages verwendet wurde. Über die Funktion zur Tabellenbereichswiederherstellung können einer Datenbank also keine Tabellenbereiche hinzugefügt werden.
- Das Dienstprogramm zur aktualisierenden Wiederherstellung stellt sicher, dass in einer Umgebung mit partitionierten Datenbanken wiederhergestellte Tabellenbereiche mit allen anderen Datenbankpartitionen synchronisiert werden, die dieselben Tabellenbereiche enthalten. Wenn eine Wiederherstellungsoperation für Tabellenbereiche angefordert wird und *piTablespaceList* NULL ist, versucht das Wiederherstellungsdienstprogramm, alle im Sicherungsimage vorhandenen Tabellenbereiche wiederherzustellen.

Beim Wiederherstellen eines Tabellenbereichs, der seit seiner Sicherung umbenannt wurde, müssen Sie im Wiederherstellungsbefehl den neuen Tabellenbereichsnamen verwenden. Wenn Sie den alten Tabellenbereichsnamen verwenden, wird der Tabellenbereich nicht gefunden.

piMediaList
Eingabe. Quellendatenträger für das Sicherungsimage. Die bereitgestellten Daten hängen vom Wert des Feldes *locationType* ab. Die folgenden Werte sind für *locationType* (definiert in `sqlutil.h`) gültig:

SQLU_LOCAL_MEDIA
Lokale Einheiten (eine Kombination von Bändern, Platten oder Disketten).

SQLU_XBSA_MEDIA
XBSA-Schnittstelle. XBSA (Backup Services APIs) ist eine offene Anwendungsprogrammierschnittstelle für Anwendungen oder Tools, die eine Datenspeicherverwaltung für Sicherungs- oder Archivierungszwecke benötigen.

SQLU_TSM_MEDIA
TSM. Wenn der Positionszeiger auf NULL gesetzt ist, wird die mit DB2 gelieferte gemeinsam benutzte Bibliothek von TSM verwendet. Wenn Sie eine andere Version der gemeinsam benutzten Bibliothek von TSM wünschen, verwenden Sie `SQLU_OTHER_MEDIA`, und geben Sie den Namen der gemeinsam benutzten Bibliothek an.

SQLU_OTHER_MEDIA
Produkt eines anderen Lieferanten. Geben Sie den Namen der gemeinsam benutzten Bibliothek im Feld *locations* an.

SQLU_USER_EXIT
Benutzerexit. Es ist keine zusätzliche Eingabe erforderlich (nur verfügbar, wenn sich der Server unter OS/2 befindet).

piUsername
Eingabe. Eine Zeichenfolge, die den Benutzernamen enthält, der beim Versuch einer Verbindung verwendet wird. Kann NULL sein.

iUsernameLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge von *piUsername* in Byte angibt. Wird auf null gesetzt, wenn kein Benutzername angegeben wird.

piPassword
Eingabe. Eine Zeichenfolge, die das Kennwort enthält, das mit dem Benutzernamen verwendet werden soll. Kann NULL sein.

iPasswordLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge von *piPassword* in Byte angibt. Wird auf null gesetzt, wenn kein Kennwort angegeben wird.

piNewLogPath
Eingabe. Eine Zeichenfolge, die den Pfad angibt, der nach Abschluss der Wiederherstellung für die Protokollierung verwendet werden soll. Wenn dieses Feld null ist, wird der Standardprotokollpfad verwendet.

iNewLogPathLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge von *piNewLogPath* in Byte angibt.

piVendorOptions
Eingabe. Wird verwendet, um Informationen von der Anwendung an die Funktionen anderer Lieferanten zu übergeben. Diese Datenstruktur muss unstrukturiert sein, d. h., Zwischenstufen werden nicht unterstützt. Beachten Sie, dass für diese Daten keine Bytefolgeumkehrung durchgeführt und die Codepage nicht überprüft wird.

iVendorOptionsSize
Eingabe. Die Länge von *piVendorOptions*, die 65535 Byte nicht überschreiten darf.

iParallelism
Eingabe. Grad der Parallelität (Anzahl Puffermanipulatoren). Der Mindestwert ist 1, der Maximalwert ist 1024.

iBufferSize
Eingabe. Sicherungspuffergröße in 4 KB großen Zuordnungseinheiten (Seiten). Die Mindestanzahl ist 8 Einheiten. Die für eine Wiederherstellung angegebene Größe muss gleich der für die Erstellung des Sicherungsimages verwendeten Puffergröße oder ein ganzzahliges Vielfaches dieses Werts ein.

iNumBuffers
Eingabe. Gibt die Anzahl der zu verwendenden Wiederherstellungspuffer an.

iCallerAction
Eingabe. Gibt die auszuführende Aktion an. Gültige Werte (definiert in `db2ApiDf.h`) sind:

DB2RESTORE_RESTORE
Die Wiederherstellungsoperation starten.

DB2RESTORE_NOINTERRUPT
Die Wiederherstellung starten. Gibt an, dass die Wiederherstellung nicht überwacht ausgeführt wird und dass Szenarios, die normalerweise Benutzereingriff erfordern, entweder versucht werden, ohne vorher an das aufrufende Programm zurückgegeben zu werden, oder einen Fehler generieren. Verwenden Sie diese Aktion z. B.

dann, wenn bekannt ist, dass alle für die Wiederherstellung erforderlichen Datenträger angehängt wurden, und Dienstprogrammeingabeaufforderungen nicht erwünscht sind.

DB2RESTORE_CONTINUE
Die Wiederherstellung fortsetzen, nachdem der Benutzer die vom Dienstprogramm angeforderten Aktionen ausgeführt hat (z. B. das Anhängen eines neuen Bandes).

DB2RESTORE_TERMINATE
Die Wiederherstellung beenden, nachdem der Benutzer eine vom Dienstprogramm angeforderte Aktion nicht ausführen konnte.

DB2RESTORE_DEVICE_TERMINATE
Eine bestimmte Einheit aus der Liste der vom Wiederherstellungsdienstprogramm verwendeten Einheiten entfernen. Wenn eine bestimmte Einheit ihre Eingabe erschöpft hat, gibt das Wiederherstellungsdienstprogramm eine Warnung an das aufrufende Programm zurück. Rufen Sie das Wiederherstellungsdienstprogramm erneut mit dieser Aktion auf, um die Einheit, die die Warnung generiert hat, aus der Liste der verwendeten Einheiten zu entfernen.

DB2RESTORE_PARM_CHK
Parameter auswerten, ohne eine Wiederherstellung auszuführen. Diese Option beendet die Datenbankverbindung nicht, nachdem der Aufruf zurückgegeben wurde. Nach einer erfolgreichen Rückgabe dieses Aufrufs wird erwartet, dass der Benutzer einen Aufruf mit `DB2RESTORE_CONTINUE` absetzt, um mit der Aktion fortzufahren.

DB2RESTORE_PARM_CHK_ONLY
Parameter auswerten, ohne eine Wiederherstellung auszuführen. Bevor dieser Aufruf zurückgegeben wird, wird die durch diesen Aufruf hergestellte Datenbankverbindung beendet, und es ist kein darauf folgender Aufruf erforderlich.

DB2RESTORE_TERMINATE_INCRE
Eine Teilwiederherstellungsoperation vor der Fertigstellung beenden.

DB2RESTORE_RESTORE_STORDEF
Erstaufruf. Erneute Definition des Tabellenbereichscontainers angefordert.

DB2RESTORE_STORDEF_NOINTERRUPT
Erstaufruf. Die Wiederherstellung wird ohne Unterbrechungen ausgeführt. Erneute Definition des Tabellenbereichscontainers angefordert.

iOptions
Eingabe. Ein Bitmap der Wiederherstellungsmerkmale. Die Optionen müssen mit dem bitweisen OR-Operator kombiniert werden, um einen Wert für *iOptions* zu erzeugen. Gültige Werte (definiert in `db2ApiDf.h`) sind:

DB2RESTORE_OFFLINE
Eine Offlinewiederherstellungsoperation ausführen.

DB2RESTORE_ONLINE
Eine Onlinewiederherstellungsoperation ausführen.

DB2RESTORE_DB
Alle Tabellenbereiche in der Datenbank wiederherstellen. Muss offline ausgeführt werden.

DB2RESTORE_TABLESPACE
Nur die im Parameter *piTablespaceList* aufgelisteten Tabellenbereiche aus dem Sicherungsimage wiederherstellen. Dies kann online oder offline ausgeführt werden.

DB2RESTORE_HISTORY
Nur die Protokolldatei wiederherstellen.

DB2RESTORE_COMPR_LIB
Gibt an, dass die Komprimierungsbibliothek wiederhergestellt werden soll. Diese Option kann nicht simultan mit einem anderen Wiederherstellungstyp verwendet werden. Wenn das Objekt im Sicherungsimage vorhanden ist, wird es im Datenbankverzeichnis wiederhergestellt. Wenn das Objekt im Sicherungsimage nicht vorhanden ist, schlägt die Wiederherstellungsoperation fehl.

DB2RESTORE_LOGS
Geben Sie diesen Parameter an, wenn nur die im Sicherungsimage vorhandene Gruppe von Protokolldateien wiederhergestellt werden soll. Wenn das Sicherungsimage keine Protokolldateien enthält, schlägt die Wiederherstellungsoperation fehl. Wird diese Option angegeben, muss auch der Parameter *piLogTarget* angegeben werden.

DB2RESTORE_INCREMENTAL
Eine manuelle kumulative Wiederherstellungsoperation ausführen.

DB2RESTORE_AUTOMATIC
Eine automatische kumulative (inkrementelle) Wiederherstellungsoperation ausführen. Muss mit DB2RESTORE_INCREMENTAL angegeben werden.

DB2RESTORE_DATALINK
Abstimmungsoperationen ausführen. Für Tabellen mit einer definierten DATALINK-Spalte muss die Option RECOVERY YES angegeben werden.

DB2RESTORE_NODATALINK
Keine Abstimmungsoperationen ausführen. Tabellen mit DATALINK-Spalten werden in den Status *DRP* (DataLink Reconcile Pending) versetzt. Für Tabellen mit einer definierten DATALINK-Spalte muss die Option RECOVERY YES angegeben werden.

DB2RESTORE_ROLLFWD
Die Datenbank nach der erfolgreichen Wiederherstellung in den Status *Aktualisierende Wiederherstellung anstehend* versetzen.

DB2RESTORE_NOROLLFWD
Die Datenbank nach der erfolgreichen Wiederherstellung nicht in den Status *Aktualisierende Wiederherstellung anstehend* versetzen. Dies kann nicht für online erstellte Sicherungen oder für Wiederherstellungen auf Tabellenbereichsebene angegeben werden. Wenn sich die Datenbank nach einer erfolgreichen Wiederherstellungsoperation im Status *Aktualisierende Wiederherstellung anstehend* befindet, muss db2Rollforward (Datenbank aktualisierend wiederherstellen) ausgeführt werden, bevor die Datenbank verwendet werden kann.

piComprLibrary
Eingabe. Gibt den Namen der externen Bibliothek an, die zum Ausführen der Dekomprimierung des Sicherungsimages verwendet werden soll, wenn das Image komprimiert ist. Der Name muss ein vollständig qualifizierter Pfad für eine Datei auf dem Server sein. Wenn der Wert ein Nullzeiger oder ein Zeiger auf eine leere Zeichenfolge ist, versucht DB2, die im Image gespeicherte Bibliothek zu verwenden. Wurde die Sicherung nicht komprimiert, wird der Wert dieses Parameters ignoriert. Wenn die angegebene Bibliothek nicht gefunden werden kann, schlägt die Wiederherstellung fehl.

piComprLibraryLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die in Byte die Länge des Namens der in `piComprLibrary` angegebenen Bibliothek angibt. Wenn kein Bibliotheksname angegeben wurde, ist dieser Parameter auf null gesetzt.

piComprOptions
Eingabe. Beschreibt einen Block binärer Daten, der an die Initialisierungsroutine in der Dekomprimierungsbibliothek übergeben wird. DB2 übergibt diese Zeichenfolge direkt vom Client an den Server, daher muss die Handhabung der Bytefolgeumkehrung oder Codepagekonvertierung von der Komprimierungsbibliothek übernommen werden. Ist '@' das erste Zeichen des Datenblocks, werden die restlichen Daten von DB2 als Name einer Datei interpretiert, die sich auf dem Server befindet. DB2 ersetzt in diesem Fall den Inhalt von `piComprOptions` und `iComprOptionsSize` durch den Inhalt bzw. die Größe dieser Datei und übergibt diesen neuen Wert stattdessen an die Initialisierungsroutine.

iComprOptionsSize
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Größe des als `piComprOptions` übergebenen Datenblocks angibt. `iComprOptionsSize` wird nur dann auf null gesetzt, wenn `piComprOptions` ein Nullzeiger ist.

piLogTarget
Eingabe. Der absolute Pfad eines vorhandenen Verzeichnisses auf dem Datenbankserver, das als Zielverzeichnis zum Extrahieren von Protokolldateien aus einem Sicherungsimage verwendet werden soll. Wenn dieser Parameter angegeben wird, werden die im Sicherungsimage vorhandenen Protokolldateien in das Zielverzeichnis extrahiert. Wird diese Option nicht angegeben, werden die im Sicherungsimage vorhandenen Protokolldateien nicht extrahiert. Wenn Sie nur die Protokolldateien aus einem Sicherungsimage extrahieren wollen, verwenden Sie den Parameter DB2RESTORE_LOGS.

iLogTargetLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die in Byte die Länge des in *piLogTarget* angegebenen Pfads angibt.

tablespaces
Ein Zeiger auf die Liste der zu sichernden Tabellenbereiche. Für C besteht diese Liste aus auf Null endenden Zeichenfolgen. Die generische Version ist eine Liste mit *db2Char*-Strukturen.

numTablespaces
Anzahl der Einträge im Parameter *tablespaces*.

locations
Ein Zeiger auf die Liste der Datenträgerpositionen. Für C besteht diese Liste aus auf Null endenden Zeichenfolgen. Die generische Version ist eine Liste mit *db2Char*-Strukturen.

numLocations
Die Anzahl der Einträge im Parameter *locations*.

locationType
Ein Zeichen, das den Datenträgertyp angibt. Gültige Werte (definiert in `sqlutil.h`) sind:

SQLU_LOCAL_MEDIA
Lokale Einheiten (Bänder, Platten, Disketten oder benannte Pipes).

SQLU_XBSA_MEDIA
XBSA-Schnittstelle.

SQLU_TSM_MEDIA
Tivoli Storage Manager.

SQLU_OTHER_MEDIA
Bibliothek eines anderen Herstellers.

SQLU_USER_EXIT
Benutzerexit (nur verfügbar, wenn sich der Server unter OS/2 befindet).

pioData
Ein Zeiger auf den Puffer für Zeichendaten.

iLength
Eingabe. Die Größe des Puffers *pioData*.

oLength
Ausgabe. Zur zukünftigen Verwendung reserviert.

Hinweise:

Für die Offlinewiederherstellung stellt dieses Dienstprogramm eine Verbindung zur Datenbank im Exklusivmodus her. Falls eine Anwendung, einschließlich der aufrufenden Anwendung, bereits eine Verbindung zur wiederherzustellenden Datenbank hat, schlägt das Dienstprogramm fehl. Die Anforderung schlägt außerdem fehl, wenn das Wiederherstellungsdienstprogramm zur Wiederherstellung verwendet wird und eine Anwendung, einschließlich der aufrufenden Anwendung, bereits mit einer Datenbank auf derselben Workstation verbunden ist. Wenn die Verbindung erfolgreich ist, sperrt die API alle übrigen Anwendungen, bis die Wiederherstellung abgeschlossen ist.

Die aktuelle Datenbankkonfigurationsdatei wird nicht durch die Sicherungskopie ersetzt, es sei denn, sie ist nicht verwendbar. Wenn die Datei ersetzt wird, wird eine Warnung zurückgegeben.

Die Datenbank bzw. der Tabellenbereich muss mit der API "db2Backup - Datenbank sichern" gesichert worden sein.

Wenn als auszuführende Aktion DB2RESTORE_NOINTERRUPT verwendet wird, wird die Wiederherstellung fortgesetzt, ohne eine Bedienerführung der Anwendung anzufordern. Wenn als auszuführende Aktion DB2RESTORE_RESTORE verwendet wird und das Dienstprogramm eine Wiederherstellung in eine vorhandene Datenbank durchführt, gibt das Dienstprogramm die Steuerung an die Anwendung

zurück und fordert mit einer entsprechenden Nachricht eine Benutzerinteraktion an. Nach Verarbeitung der Benutzerinteraktion ruft die Anwendung erneut RESTORE DATABASE auf, wobei in der auszuführenden Aktion angegeben wird, ob die Verarbeitung fortgesetzt werden soll (DB2RESTORE_CONTINUE) oder ob sie beim nächsten Aufruf beendet werden soll (DB2RESTORE_TERMINATE). Das Dienstprogramm schließt die Verarbeitung ab und gibt in der Struktur *sqlca* einen SQLCODE-Wert zurück.

Zum Schließen einer Einheit nach Beendigung der Wiederherstellung setzen Sie die auszuführende Aktion auf DB2RESTORE_DEVICE_TERMINATE. Wenn beispielsweise ein Benutzer eine Wiederherstellung von 3 Banddatenträgern durchführt, dabei 2 Bandeinheiten verwendet und eines der Bänder wiederhergestellt wurde, gibt die API mit einem SQLCODE-Wert, der das Ende des Bandes andeutet, die Steuerung zurück an die Anwendung. Die Anwendung kann den Benutzer zum Anhängen eines weiteren Bandes auffordern. Falls der Benutzer angibt, dass er keine weiteren Bänder verwenden möchte, kehrt die Anwendung mit der auszuführenden Aktion SQLUD_DEVICE_TERMINATE zur API zurück, um das Ende der externen Einheit zu signalisieren. Der Einheitentreiber wird beendet, für die übrigen, an der Wiederherstellung beteiligten Einheiten wird jedoch weiterhin die Eingabe verarbeitet, bis alle Segmente der Wiederherstellungsgruppe wiederhergestellt wurden (die Anzahl Segmente in der Wiederherstellungsgruppe wird während des Sicherungsprozesses auf der letzten externen Einheit angegeben). Diese auszuführende Aktion kann außer mit Bändern auch mit anderen (von anderen Herstellern unterstützten) Einheiten verwendet werden.

Wenn Sie vor der Rückkehr zur Anwendung eine Parameterüberprüfung durchführen wollen, setzen Sie die auszuführende Aktion auf DB2RESTORE_PARM_CHK.

Setzen Sie die auszuführende Aktion auf DB2RESTORE_RESTORE_STORDEF, wenn Sie eine umgeleitete Wiederherstellung ausführen (wird zusammen mit der API "sqlbstsc - Tabellenbereichscontainer festlegen" verwendet).

Wenn während eines kritischen Arbeitsabschnitts der Wiederherstellung einer Datenbank ein Fehler auftritt, kann der Benutzer erst dann eine erfolgreiche Verbindung zur Datenbank herstellen, nachdem eine erfolgreiche Wiederherstellung ausgeführt wurde. Diese Bedingung wird festgestellt, sobald ein Verbindungsaufbau versucht wird, und es wird eine Fehlernachricht zurückgegeben. Wenn die gesicherte Datenbank nicht für aktualisierende Wiederherstellung konfiguriert ist und eine verwendbare, aktuelle Konfigurationsdatei vorhanden ist, in der einer dieser Parameter aktiviert ist, wird der Benutzer im Anschluss an die Wiederherstellung aufgefordert, vor einer Verbindungsherstellung zur Datenbank entweder eine neue Sicherung der Datenbank zu erstellen oder die Parameter *logretain* und *userexit* zu inaktivieren.

Auch wenn die wiederhergestellte Datenbank nicht gelöscht wird (außer bei Wiederherstellung in eine nicht vorhandene Datenbank), ist sie bei einem Fehlschlagen der Wiederherstellung nicht verwendbar.

Wenn der Wiederherstellungstyp angibt, dass die Protokolldatei der Sicherung wiederhergestellt werden soll, wird die Datei über der vorhandenen Protokolldatei für die Datenbank wiederhergestellt und löscht dadurch alle an der Protokolldatei vorgenommenen Änderungen, die nach Erstellung der wiederherzustellenden Sicherung durchgeführt wurden. Falls dies nicht gewünscht wird, stellen Sie die Protokolldatei in eine neue Datenbank oder eine Testdatenbank wieder her, so dass ihr Inhalt angezeigt werden kann, ohne vorgenommene Aktualisierungen zu überschreiben.

Falls die Datenbank zum Zeitpunkt der Sicherungsoperation für aktualisierende Wiederherstellung konfiguriert war, kann die Datenbank in den Status versetzt werden, in dem sie sich vor dem Auftritt der Beschädigung befand. Setzen Sie nach erfolgreicher Ausführung von db2Restore dazu db2Rollforward ab. Wenn die Datenbank wiederherstellbar ist, nimmt sie nach Beendigung der Wiederherstellung standardmäßig den Status *Aktualisierende Wiederherstellung anstehend* an.

Wenn das Sicherungsimage der Datenbank offline erstellt wird und das aufrufende Programm die Datenbank nach der Wiederherstellung nicht aktualisierend wiederherstellen will, kann zur Wiederherstellung die Option DB2RESTORE_NOROLLFWD verwendet werden. Dies bewirkt, dass die Datenbank nach der Wiederherstellung sofort verwendet werden kann. Falls das Sicherungsimage online erstellt wird, muss das aufrufende Programm nach Beendigung der Wiederherstellung anhand der zugehörigen Protokollsätze eine aktualisierende Wiederherstellung ausführen.

Zum Wiederherstellen von Protokolldateien aus dem entsprechenden Sicherungsimage muss über die Option LOGTARGET ein vollständig qualifizierter und gültiger Pfad angegeben werden, der auf dem DB2-Server vorhanden ist. Sind diese Bedingungen erfüllt, schreibt das Wiederherstellungsdienstprogramm die Protokolldateien aus dem Image in den Zielpfad. Wenn die Option LOGTARGET während der Wiederherstellung eines Sicherungsimages angegeben wird, das keine Protokolldateien enthält, gibt die Wiederherstellung einen Fehler zurück, bevor mit der Wiederherstellung von Tabellenbereichsdaten begonnen wird. Eine Wiederherstellung schlägt ebenfalls mit einer Fehlermeldung fehl, wenn ein ungültiger oder schreibgeschützter Pfad für LOGTARGET angegeben wird.

Sind beim Absetzen des Befehls RESTORE bereits Protokolldateien im Pfad für LOGTARGET vorhanden, wird eine Warnung an den Benutzer zurückgegeben. Diese Warnung wird nicht zurückgegeben, wenn WITHOUT PROMPTING angegeben wird.

Wenn während einer Wiederherstellungsoperation, bei der die Option LOGTARGET angegeben wurde, eine Protokolldatei aus irgendeinem Grund nicht extrahiert werden kann, schlägt die Wiederherstellung fehl, und es wird ein Fehler zurückgegeben. Ist der Name einer Protokolldatei, die aus dem Sicherungsimage extrahiert wird, mit dem Namen einer bereits im Pfad für LOGTARGET vorhandenen Datei identisch, schlägt die Wiederherstellungsoperation fehl, und es wird ein Fehler zurückgegeben. Das Wiederherstellungsdienstprogramm überschreibt bereits im Pfad für LOGTARGET vorhandene Dateien nicht.

Sie haben auch die Möglichkeit, aus einem Sicherungsimage nur die darin gespeicherten Protokolldateien wiederherzustellen. Geben Sie dazu die Option LOGS zusätzlich zum Pfad für LOGTARGET an. Wenn die Option LOGS ohne einen Pfad für LOGTARGET angegeben wird, führt dies zu einem Fehler. Treten in diesem Betriebsmodus Fehler beim Wiederherstellen von Protokolldateien auf, wird die Wiederherstellung sofort beendet, und es wird ein Fehler zurückgegeben.

Bei einer automatischen Teilwiederherstellung werden nur die Protokolle aus dem Sicherungsimage abgerufen, die sich im Zielimage der Wiederherstellungsoperation befinden. Protokolle, die sich in temporären Images befinden, auf die während des Teilwiederherstellungsprozesses verwiesen wird, werden nicht aus diesen temporären Sicherungsimages extrahiert. Bei einer manuellen Teilwiederherstellung darf der Pfad für LOGTARGET nur in dem zuletzt abgesetzten Wiederherstellungsbefehl angegeben werden.

Wenn ein Sicherungsimage komprimiert ist, stellt DB2 dies fest und dekomprimiert die Daten automatisch, bevor die Wiederherstellung gestartet wird. Wird für die Anwendungsprogrammierschnittstelle `db2Restore` eine Bibliothek angegeben, wird diese zur Dekomprimierung der Daten verwendet. Andernfalls wird eine im Sicherungsimage gespeicherte Bibliothek verwendet. Ist keine Bibliothek vorhanden, können die Daten nicht dekomprimiert werden, und die Wiederherstellung schlägt fehl.

Wenn die Komprimierungsbibliothek aus einem Sicherungsimage wiederhergestellt wird (entweder explizit durch Angeben des Wiederherstellungstyps DB2RESTORE_COMPR_LIB oder implizit durch Ausführen einer normalen Wiederherstellung einer komprimierten Sicherung), muss die Wiederherstellungsoperation auf derselben Plattform und demselben Betriebssystem ausgeführt werden, auf der bzw. dem die Sicherung ausgeführt wurde. Wenn die Plattform, auf der die Sicherung erstellt wurde, nicht mit der Plattform identisch ist, auf der die Wiederherstellung ausgeführt wird, schlägt die Wiederherstellungsoperation fehl, auch wenn DB2 normalerweise plattformübergreifende Wiederherstellungen für diese beiden Systeme unterstützt.

Zugehörige Referenzen:

- „sqlemgdb - Migrate Database" im Handbuch *Administrative API Reference*
- „db2Rollforward - Datenbank aktualisierend wiederherstellen" auf Seite 154
- „SQLCA" im Handbuch *Administrative API Reference*
- „db2Backup - Datenbank sichern" auf Seite 85
- „db2CfgGet - Get Configuration Parameters" im Handbuch *Administrative API Reference*

Zugehörige Beispiele:

- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.sqC -- How to recover a database (C++)"

Wiederherstellungssitzungen - Beispiele für CLP

Beispiel 1

Es folgt ein typisches Beispielszenario für die umgeleitete, nicht inkrementelle Wiederherstellung 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 Behälter Sie erneut definieren wollen, 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 Behälterpositionen erneut definiert werden, den Befehl LIST TABLESPACE CONTAINERS ab, um sicherzustellen, dass es sich bei den Behältern der wiederhergestellten Datenbank um die in diesem Schritt angegebenen Behälter handelt.

3. Nach der erfolgreichen Durchführung der Schritte 1 und 2 setzen Sie den folgenden Befehl ab:

   ```
   db2 restore db meinedb continue
   ```

Dies ist der letzte Schritt der umgeleiteten Wiederherstellung.

4. Falls Schritt 3 fehlschlägt oder die Wiederherstellungsoperation abgebrochen wurde, kann die umgeleitete Wiederherstellung erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen.

Anmerkungen:

1. Nach der erfolgreichen Ausführung von Schritt 1 und vor der Vollendung von Schritt 3 kann die Wiederherstellungsoperation durch Absetzen des folgenden Befehls abgebrochen werden:

   ```
   db2 restore db meinedb abort
   ```

2. Falls Schritt 3 fehlschlägt oder die Wiederherstellungsoperation abgebrochen wurde, kann die umgeleitete Wiederherstellung erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen.

Beispiel 2

Es folgt ein typisches Beispielszenario für die manuelle, umgeleitete Teilwiederherstellung einer Datenbank mit dem Aliasnamen MEINEDB und den folgenden Sicherungsimages:

```
backup db meinedb
Sicherung erfolgreich. Die Zeitmarke für diese Sicherungsimagedatei ist: <zm1>

backup db meinedb incremental
Sicherung erfolgreich. Die Zeitmarke für diese Sicherungsimagedatei ist: <zm2>
```

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 Behälter 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 Behältern der wiederhergestellten Datenbank um die in diesem Schritt angegebenen Behälter handelt.

3. Nach der erfolgreichen Durchführung der Schritte 1 und 2 setzen Sie den folgenden Befehl ab:

   ```
   db2 restore db meinedb continue
   ```

4. Die übrigen Befehle zur Teilwiederherstellung können nun wie folgt abgesetzt werden:

   ```
   db2 restore db meinedb incremental taken at <zm1>
   db2 restore db meinedb incremental taken at <zm2>
   ```

   Dies ist der letzte Schritt der umgeleiteten Wiederherstellung.

Anmerkungen:

1. Nach der erfolgreichen Ausführung von Schritt 1 und vor der Vollendung von Schritt 3 kann die Wiederherstellungsoperation 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 Wiederherstellungsoperation durch Absetzen des folgenden Befehls abgebrochen werden:

   ```
   db2 restore db meinedb incremental abort
   ```

3. Falls Schritt 3 fehlschlägt oder die Wiederherstellungsoperation abgebrochen wurde, kann die umgeleitete Wiederherstellung erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen.
4. Wenn der Befehl zur Wiederherstellung in Schritt 4 fehlschlägt, kann der fehlgeschlagene Befehl erneut abgesetzt werden, um den Wiederherstellungsprozess fortzusetzen.

Beispiel 3

Es folgt ein typisches Beispielszenario für die automatische, umgeleitete Teilwiederherstellung 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 Behälter 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 Behältern der wiederhergestellten Datenbank um die in diesem Schritt angegebenen Behälter handelt.
3. Nach der erfolgreichen Durchführung der Schritte 1 und 2 setzen Sie den folgenden Befehl ab:

   ```
   db2 restore db meinedb continue
   ```

   Dies ist der letzte Schritt der umgeleiteten Wiederherstellung.

Anmerkungen:

1. Nach der erfolgreichen Ausführung von Schritt 1 und vor der Vollendung von Schritt 3 kann die Wiederherstellungsoperation durch Absetzen des folgenden Befehls abgebrochen werden:

   ```
   db2 restore db meinedb abort
   ```

2. Falls Schritt 3 fehlschlägt oder die Wiederherstellungsoperation abgebrochen wurde, kann die umgeleitete Wiederherstellung erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen, nachdem Sie den folgenden Befehl abgesetzt haben:

   ```
   db2 restore db meinedb incremental abort
   ```

Zugehörige Referenzen:

- „RESTORE DATABASE" auf Seite 102
- „LIST TABLESPACE CONTAINERS Command" im Handbuch *Command Reference*
- „SET TABLESPACE CONTAINERS Command" im Handbuch *Command Reference*

Optimieren der Leistung des Wiederherstellungsdienstprogramms

Wenn Sie eine Wiederherstellungsoperation ausführen, wählt DB2 automatisch optimale Werte für die Anzahl Puffer, die Puffergröße und die Parallelitätseinstellungen aus. Die Werte basieren auf der Menge des für das Dienstprogramm verfügbaren Zwischenspeichers, der Anzahl verfügbarer Prozessoren und der Datenbankkonfiguration. Dadurch soll die zum Ausführen einer Wiederherstellungsoperation erforderliche Zeit minimiert werden. Wenn Sie für die folgenden Parameter des Befehls RESTORE DATABASE nicht explizit Werte angeben, wählt DB2 einen Wert für die Parameter aus:

- WITH anzahl-puffer BUFFERS
- PARALLELISM n
- BUFFER puffergröße

Für Wiederherstellungsoperationen wird immer ein Vielfaches der von der Sicherungsoperation verwendeten Puffergröße verwendet. Die von den Konfigurationsparametern BACKBUFSZ und RESTBUFSZ des Datenbankmanagers angegebenen Werte werden ignoriert. Wenn Sie diese Werte verwenden wollen, müssen Sie beim Absetzen des Befehls RESTORE DATABASE explizit eine Puffergröße angeben.

Sie können auch eine der folgenden Aktionen ausführen, um die für eine Wiederherstellungsoperation erforderliche Zeit zu reduzieren:

- Erhöhen Sie die Größe der Wiederherstellungspuffer.

  Die Größe der Wiederherstellungspuffer muss eine positive ganze Zahl und ein Vielfaches der Sicherungspuffergröße sein, die bei der Sicherungsoperation 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 Anzahl der Seiten sein, die Sie für den Sicherungspuffer angegeben haben. Die Mindestanzahl beträgt 8 Seiten.
- Erhöhen Sie den Wert des Parameters PARALLELISM.

  Dadurch wird die Anzahl der Puffermanipulatoren erhöht, die verwendet werden, um während der Wiederherstellungsoperation in die Datenbank zu schreiben.

Zugehörige Konzepte:

- „Wiederherstellung - Übersicht" auf Seite 95

Zugehörige Tasks:

- „Verwenden des Wiederherstellungsprogramms" auf Seite 96

Kapitel 4. Aktualisierende Wiederherstellung

In diesem Abschnitt wird das DB2 UDB-Dienstprogramm zur aktualisierenden Wiederherstellung beschrieben, mit dem eine Datenbank durch Anwenden von Transaktionen wiederhergestellt werden kann, die in den Dateien des Datenbankwiederherstellungsprotokolls aufgezeichnet wurden.

Die folgenden Themen werden behandelt:

Aktualisierende Wiederherstellung - Übersicht

In der einfachsten Form des DB2®-Befehls ROLLFORWARD DATABASE müssen Sie lediglich den Aliasnamen der Datenbank angeben, die Sie aktualisierend wiederherstellen wollen. Beispiel:

```
db2 rollforward db sample to end of logs and stop
```

In diesem Beispiel gibt der Befehl Folgendes zurück:

```
               Status der aktualisierenden Wiederherstellung

 Aliasname der Eingabedatenbank             = sample
 Anzahl der Knoten, die Status zurückgaben  = 1

 Knotennummer                               = 0
 Aktualisierende Wiederherstellung: Status  = nicht anstehend
 Nächste zu lesende Protokolldatei          =
 Verarbeitete Protokolldateien              =  -
 Letzte festgeschriebene Transaktion        = 2001-03-11-02.39.48.000000

DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die allgemeine Vorgehensweise bei der aktualisierenden Wiederherstellung umfasst Folgendes:

1. Aufrufen des Dienstprogramms zur aktualisierenden Wiederherstellung ohne die Option STOP.

2. Aufrufen des Dienstprogramms zur aktualisierenden Wiederherstellung mit der Option QUERY STATUS

   Wenn Sie eine Wiederherstellung bis zum Ende der Protokolle angeben, kann die Option QUERY STATUS angeben, dass mindestens eine Protokolldatei fehlt, falls der zurückgegebene Zeitpunkt vor dem erwarteten Zeitpunkt liegt.

   Wenn Sie eine Wiederherstellung bis zu einem bestimmten Zeitpunkt angeben, können Sie mit der Option QUERY STATUS sicherstellen, dass die aktualisierende Wiederherstellung am richtigen Zeitpunkt beendet wird.
3. Aufrufen des Dienstprogramms zur aktualisierenden Wiederherstellung mit der Option STOP. Nach Beendigung dieser Operation können keine zusätzlichen Änderungen mehr aktualisierend wiederhergestellt werden.

Eine Datenbank muss erst erfolgreich wiederhergestellt werden (mit dem Wiederherstellungsdienstprogramm), bevor sie aktualisierend wiederhergestellt werden kann. Bei einem Tabellenbereich ist dies jedoch nicht erforderlich. Ein Tabellenbereich kann zeitweilig in den Status *Aktualisierende Wiederherstellung anstehend* versetzt werden, erfordert aber keine Wiederherstellungsoperation, um diesen Status aufzuheben (z. B. nach einem Stromausfall).

Wenn das Dienstprogramm zur aktualisierenden Wiederherstellung aufgerufen wird, geschieht Folgendes:

- Wenn sich die Datenbank im Status *Aktualisierende Wiederherstellung anstehend* befindet, wird die Datenbank aktualisierend wiederhergestellt. Wenn auch Tabellenbereiche diesen Status haben, müssen Sie nach der aktualisierenden Wiederherstellung der Datenbank das Dienstprogramm zur aktualisierenden Wiederherstellung erneut aufrufen, um diese Tabellenbereiche aktualisierend wiederherzustellen.
- Wenn die Datenbank *nicht* den Status *Aktualisierende Wiederherstellung anstehend* hat, sich jedoch Tabellenbereiche der Datenbank in diesem Status *befinden*, gilt Folgendes:
  - 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 Wiederherstellung anstehend* haben.

Die aktualisierende Wiederherstellung einer Datenbank wird offline ausgeführt. Die Datenbank steht erst dann wieder zur Verfügung, wenn die aktualisierende Wiederherstellung erfolgreich beendet wurde. Diese Operation kann jedoch nur beendet werden, wenn beim Aufrufen des Dienstprogramms die Option STOP angegeben wurde.

Die aktualisierende Wiederherstellung von Tabellenbereichen kann offline ausgeführt werden. Die Datenbank steht erst dann wieder zur Verfügung, wenn die aktualisierende Wiederherstellung 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 Wiederherstellung von Tabellenbereichen auch *online* ausführen. Wenn Sie eine aktualisierende Wiederherstellung für einen Tabellenbereich online ausführen, 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 Wiederherstellung nicht möglich. Es kann lediglich eine Wiederherstellung nach einem Systemabsturz bzw. eine Versionswiederherstellung ausgeführt werden. Archivprotokolldateien erfassen die Änderungen, die nach dem Erstellen eines Sicherungsimages an einer Datenbank vorgenommen werden. Sie können diese Protokollierung (und aktualisierende Wiederherstellung) aktivieren, indem Sie den Datenbankkonfigurationsparameter *logarchmeth1* auf einen anderen Wert als den Standardwert `OFF` setzen. Wenn Sie *logarchmeth1* auf einen anderen Wert als `OFF` setzen, erhält die Datenbank erneut den Status *Sicherung anstehend*, und Sie müssen offline ein Sicherungsimage der Datenbank erstellen, bevor Sie sie weiter verwenden können.

Anmerkung: Für jede Protokolldatei, die in einer Operation zur aktualisierenden Wiederherstellung verwendet wird, wird in der Datei des Wiederherstellungsprotokolls ein Eintrag erstellt.

Zugehörige Konzepte:

- „Wiederherstellen von Daten unter Verwendung der Datei mit den Angaben zur Speicherposition der Ladekopie" auf Seite 140
- „Wiederherstellungsprotokolle" auf Seite 37

Zugehörige Referenzen:

- „ROLLFORWARD DATABASE" auf Seite 144
- „Konfigurationsparameter für die Datenbankprotokollierung" auf Seite 42
- „logarchmeth1 - Primäre Protokollarchivierungsmethode (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*

Erforderliche Zugriffsrechte und Berechtigungen zur Verwendung des Dienstprogramms zur aktualisierenden Wiederherstellung

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf sie zuzugreifen. Berechtigungsstufen stellen eine Methode dar, Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zu gruppieren. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte. Benutzer können nur auf solche Objekte zugreifen, für die sie die entsprechende Berechtigung besitzen, d. h., für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Sie benötigen die Berechtigung SYSADM, SYSCTRL oder SYSMAINT, um das Dienstprogramm zur aktualisierenden Wiederherstellung verwenden zu können.

Verwenden des Dienstprogramms zur aktualisierenden Wiederherstellung

Vorbedingungen:

Es sollte noch keine Verbindung zu der Datenbank bestehen, die aktualisierend wiederhergestellt werden soll: Das Dienstprogramm zur aktualisierenden Wiederherstellung stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der aktualisierenden Wiederherstellung beendet wird.

Stellen Sie nur dann Tabellenbereiche wieder her, wenn die aktuell laufende Operation zur aktualisierenden Wiederherstellung abgebrochen ist; andernfalls erhalten Sie möglicherweise eine Tabellenbereichsgruppe, in der sich einige Tabellenbereiche im Status *Aktualisierende Wiederherstellung läuft* befinden und andere im Status *Aktualisierende Wiederherstellung anstehend*. Eine aktive Operation zur aktualisierenden Wiederherstellung wirkt sich nur auf Tabellenbereiche aus, die sich im Status *Aktualisierende Wiederherstellung läuft* befinden.

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln.

Einschränkungen:

Für das Dienstprogramm zur aktualisierenden Wiederherstellung gelten die folgenden Einschränkungen:

- Sie können jeweils nur eine Operation zur aktualisierenden Wiederherstellung aufrufen. Wenn Sie mehrere Tabellenbereiche wiederherstellen wollen, können Sie alle Bereiche in derselben Operation angeben.
- Wenn Sie nach der letzten Sicherungsoperation einen Tabellenbereich umbenannt haben, müssen Sie bei der aktualisierenden Wiederherstellung sicherstellen, dass der neue Name verwendet wird. Der vorherige Tabellenbereichsname wird nicht erkannt.
- Sie können eine laufende Operation zur aktualisierenden Wiederherstellung nicht abbrechen. Es können nur Operationen zur aktualisierenden Wiederherstellung abgebrochen werden, die zwar beendet wurden, für die jedoch die Option STOP nicht angegeben wurde, oder Operationen, die vor der erfolgreichen Beendigung fehlgeschlagen sind.
- Sie können eine aktualisierende Wiederherstellung 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 Wiederherstellung bis zu einem bestimmten Zeitpunkt dadurch einleiten, dass Sie lediglich die Option STOP angeben; dies ist jedoch nur dann zulässig, wenn die betroffenen Tabellenbereiche zuvor alle aus demselben Offlinesicherungsimage wiederhergestellt wurden. In diesem Fall ist keine Protokollverarbeitung erforderlich. Wenn Sie für eine Liste anderer Tabellenbereiche eine weitere Operation zur aktualisierenden Wiederherstellung starten, bevor die laufende Operation beendet oder abgebrochen wurde, wird eine Fehlernachricht (SQL4908) zurückgegeben. Rufen Sie für alle Knoten den Befehl LIST TABLESPACES auf, um festzustellen, welche Tabellenbereiche aktualisierend wiederhergestellt werden (Status *Aktualisierende Wiederherstellung läuft*) und für welche Tabellenbereiche die aktualisierende Wiederherstellung noch ausgeführt werden muss (Status *Aktualisierende Wiederherstellung anstehend*). Sie haben folgende Möglichkeiten:
  - Beenden Sie die aktuelle Operation zur aktualisierenden Wiederherstellung für alle Tabellenbereiche.
  - Beenden Sie die aktuelle Operation zur aktualisierenden Wiederherstellung für eine Untergruppe von Tabellenbereichen. (Dies ist evtl. nicht möglich, wenn die Operation zur aktualisierenden Wiederherstellung bis zu einem bestimmten Zeitpunkt ausgeführt werden soll, wofür die Beteiligung aller Knoten erforderlich ist.)
  - Brechen Sie die aktuelle Operation zur aktualisierenden Wiederherstellung ab.
- In einer Umgebung mit partitionierten Datenbanken muss das Dienstprogramm zur aktualisierenden Wiederherstellung vom Katalogknoten der Datenbank aus aufgerufen werden.

Prozedur:

Das Dienstprogramm zur aktualisierenden Wiederherstellung kann über den Befehlszeilenprozessor (CLP), das Notizbuch **Datenbank aktualisierend wiederherstellen** in der Steuerzentrale oder die Anwendungsprogrammierschnittstelle **db2Rollforward** aufgerufen werden.

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
```

Gehen Sie wie folgt vor, um das Notizbuch **Datenbank aktualisierend wiederherstellen** zu öffnen:

1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis Sie den Ordner **Datenbanken** finden.
2. Klicken Sie den Ordner **Datenbanken** an. Alle vorhandenen Datenbanken werden auf der rechten Seite des Fensters, im Inhaltsteilfenster, angezeigt.
3. Klicken Sie im Inhaltsteilfenster die gewünschte Datenbank mit Maustaste 2 an, und wählen Sie **Datenbank aktualisierend wiederherstellen** im Kontextmenü aus. Das Notizbuch **Datenbank aktualisierend wiederherstellen** wird geöffnet.

Zusatzinformationen bietet die Onlinehilfefunktion der Steuerzentrale.

Zugehörige Konzepte:

- „Administrative APIs in Embedded SQL or DB2 CLI Programs" im Handbuch *Application Development Guide: Programming Client Applications*
- „Einführung der Plug-in-Architektur für die Steuerzentrale" im Handbuch *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „db2Rollforward - Datenbank aktualisierend wiederherstellen" auf Seite 154

Aktualisierendes Wiederherstellen von Änderungen in einem Tabellenbereich

Wenn die Datenbank zur aktualisierenden Wiederherstellung aktiviert ist, können Sie Sicherungen, Wiederherstellungen und aktualisierende Wiederherstellungen nicht nur für die ganze Datenbank, sondern auch für Tabellenbereiche ausführen. Es kann sinnvoll sein, eine Wiederherstellungsstrategie für einzelne Tabellenbereiche zu implementieren, da sich dadurch möglicherweise Zeit einsparen lässt: Eine Wiederherstellung eines Teils der Datenbank dauert nicht so lange wie eine Wiederherstellung der gesamten Datenbank. Wenn zum Beispiel eine Festplatte fehlerhaft ist und nur einen Tabellenbereich enthält, kann der Tabellenbereich von einer Sicherung 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 in dieser Situation keine Verbindung zu der Datenbank herstellen. (Der Tabellenbereich mit den Systemkatalogtabellen kann unabhängig von der Datenbank wiederhergestellt werden, wenn für diesen Tabellenbereich eine Sicherung auf Tabellenbereichsebene verfügbar ist.) Außerdem bieten Sicherungen auf Tabellen-

bereichsebene die Möglichkeit, kritische Teile der Datenbank häufiger als andere Teile zu sichern, und sind weniger zeitintensiv als Sicherungen der gesamten Datenbank.

Nach der Wiederherstellung eines Tabellenbereichs befindet sich dieser immer im Status *Aktualisierende Wiederherstellung anstehend*. Um den Tabellenbereich verwendbar zu machen, müssen Sie eine aktualisierende Wiederherstellung für ihn ausführen. In den meisten Fällen haben Sie dabei die Möglichkeit, die aktualisierende Wiederherstellung bis zum Ende der Protokolldateien oder bis zu einem bestimmten Zeitpunkt auszuführen. Eine aktualisierende Wiederherstellung 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.

Wenn ein Tabellenbereich aktualisierend wiederhergestellt wird, verarbeitet DB2® alle Dateien, auch wenn sie keine Protokollsätze enthalten, die diesen Tabellenbereich betreffen. Wenn Sie möchten, dass die Protokolldateien übersprungen werden, die keine Protokollsätze für diesen Tabellenbereich enthalten, setzen Sie die Registrierdatenbankvariable DB2_COLLECT_TS_REC_INFO auf *ON*. Die Registrierdatenbankvariable muss vor der Erstellung und Verwendung der Protokolldateien gesetzt werden, um sicherzustellen, dass die zum Überspringen der Protokolldateien erforderlichen Informationen erfasst werden.

Die im Datenbankverzeichnis vorhandene Protokolldatei der Tabellenbereichsänderungen (DB2TSCHG.HIS) enthält Informationen dazu, welche Protokolle für die einzelnen Tabellenbereiche verarbeitet werden sollten. Sie können den Inhalt dieser Datei mit Hilfe des Dienstprogramms **db2logsForRfwd** anzeigen und mit dem Befehl PRUNE HISTORY Einträge aus der Datei löschen. Während einer Wiederherstellungsoperation für eine Datenbank wird die Datei DB2TSCHG.HIS aus dem Sicherungsimage wiederhergestellt und während der aktualisierenden Wiederherstellung aktualisiert. Falls für eine Protokolldatei keine Informationen verfügbar sind, wird die Datei behandelt, als ob sie für die Wiederherstellung 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 Wiederherstellungsoperation 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 Archivprotokolldateien bei einem Absturz verloren gehen und in der Datendatei keine Informationen für die Protokolle vorhanden sind, werden die Protokolldateien während der Wiederherstellungsoperation für den Tabellenbereich behandelt, als ob sie Änderungen für alle Tabellenbereiche enthielten.

Führen Sie vor der aktualisierenden Wiederherstellung eines Tabellenbereichs den Befehl LIST TABLESPACES SHOW DETAIL aus. Dieser Befehl gibt den *Mindestwiederherstellungszeitpunkt* zurück, d. h. den frühesten Zeitpunkt, bis zu dem der Tabellenbereich aktualisierend wiederhergestellt werden kann. Der Mindestwiederherstellungszeitpunkt wird aktualisiert, wenn DDL-Anweisungen (DDL - Datendefinitionssprache) für den Tabellenbereich oder für Tabellen in diesem Tabellenbereich ausgeführt werden. Der Tabellenbereich muss mindestens bis zum Mindestwiederherstellungszeitpunkt 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 Mindestwiederherstellungszeitpunkt für alle Tabellenbereiche aktualisierend wiederhergestellt werden. Setzen Sie in einer Umgebung mit partitionierten Datenbanken für alle Partitionen den Befehl LIST TABLESPACES SHOW DETAILS ab. Die Tabellenbereiche müssen bis zum frühesten Mindestwiederherstellungszeitpunkt für alle Tabellenbereiche auf allen Partitionen 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 einer Sicherung wiederhergestellt und aktualisierend wiederhergestellt werden. Es ist ratsam, nach dem Reorganisieren der Tabelle eine Sicherung 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 zu Grunde liegende Tabelle für eine gespeicherte Abfrage oder Zwischenspeichertabelle, die sich in einem anderen Tabellenbereich befindet
- Eine gespeicherte Abfragetabelle 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 gespeicherte Abfragetabelle oder Zwischenspeichertabelle am Ende der aktualisierenden Wiederherstellung in den Status *Überprüfung anstehend* versetzt. Die gespeicherte Abfragetabelle 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 Integritätsabhängigkeit am Ende der aktualisierenden Wiederherstellung in den Status *Überprüfung 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, wird sie ebenfalls zusammen mit der untergeordneten Tabelle in den Status *Überprüfung anstehend* versetzt:

- Jede untergeordnete gespeicherte Abfragetabelle für die untergeordnete Tabelle
- Jede untergeordnete Zwischenspeichertabelle für die untergeordnete Tabelle
- Jede untergeordnete Fremdschlüsseltabelle der untergeordneten Tabelle

Wenn Sie diese Tabellen aus dem Status *Überprüfung anstehend* herausnehmen wollen, müssen Sie sie vollständig verarbeiten. Wenn Sie beide Tabellenbereiche zur selben Zeit aktualisierend wiederherstellen, bleibt die Integritätsbedingung am Ende der aktualisierenden Wiederherstellung bis zu einem bestimmten Zeitpunkt aktiv.

Stellen Sie sicher, dass die aktualisierende Wiederherstellung 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 Wiederherstellung 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 Auslöser oder wird von einem Auslöser 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 Wiederherstellung 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 Wiederherstellungsprotokolls nach Wartezeitpunkten suchen und prüfen, ob sie nach dem Mindestwiederherstellungszeitpunkt liegen, um einen geeigneten Zeitpunkt für den Stopp der aktualisierenden Wiederherstellung festzulegen.

Nach Beendigung der aktualisierenden Wiederherstellung bis zu einem bestimmten Zeitpunkt wird der Tabellenbereich wieder in den Status *Sicherung anstehend* versetzt. Sie müssen eine Sicherung des Tabellenbereichs erstellen, weil alle Aktualisierungen für den Zeitraum zwischen dem Zeitpunkt, bis zu dem die aktualisierende Wiederherstellung erfolgte, und dem aktuellen Zeitpunkt entfernt wurden. Sie können den Tabellenbereich nicht mehr von einer vorherigen Sicherung der Datenbank bzw. des Tabellenbereichs bis zum aktuellen Zeitpunkt aktualisierend wiederherstellen. Das folgende Beispiel zeigt, warum die Sicherung des Tabellenbereichs erforderlich ist und wie sie verwendet wird. (Um den Tabellenbereich verfügbar zu machen, können Sie entweder die gesamte Datenbank sichern, oder nur den Tabellenbereich, der den Status *Sicherung anstehend* hat, oder eine Gruppe von Tabellenbereichen, die diesen letztgenannten Tabellenbereich enthält.)

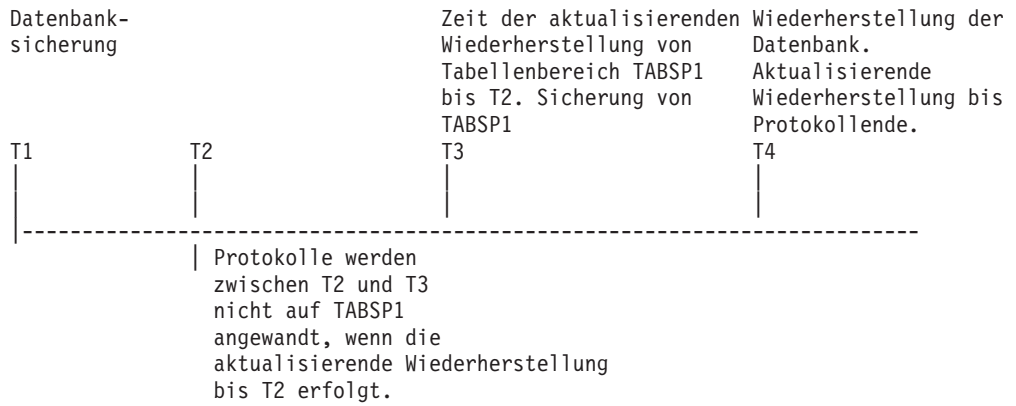


Abbildung 15. Sicherungsanforderung für Tabellenbereiche

Im vorstehenden Beispiel wird die Datenbank zum Zeitpunkt T1 gesichert. Anschließend erfolgt für den Tabellenbereich TABER1 zum Zeitpunkt T3 eine aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt (T2). Der Tabellenbereich wird nach Zeitpunkt T3 gesichert. Da sich der Tabellenbereich im Status *Sicherung anstehend* befindet, muss ein Sicherungsimage erstellt werden. Die Zeitmarke des Sicherungsimages 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 TABER1 nicht angewendet. Die Datenbank wird unter Verwendung des bei T1 erstellten Sicherungsimages zum Zeitpunkt T4 wiederhergestellt und bis zum Ende der Protokolle aktualisierend wiederhergestellt. Der Tabellenbereich TABER1 wird zum Zeitpunkt T3 in den Status *Wiederherstellung anstehend* versetzt, da der Datenbankmanager annimmt, dass zwischen T3 und T4 Operationen an TABER1 ausgeführt wurden, ohne dass die Protokolländerungen zwischen T2 und T3 auf den Tabellenbereich angewendet wurden. Wären diese Protokolländerungen tatsächlich als Teil der aktualisierenden Wiederherstellung der Datenbank angewendet worden, wäre diese Annahme falsch. Die erforderliche Sicherung eines Tabellenbereichs, die nach der aktualisierenden Wiederherstellung bis zu einem bestimmten Zeitpunkt erstellt werden muss, ermöglicht es, diesen Tabellenbereich bis nach dem Zeitpunkt einer vorherigen aktualisierenden Wiederherstellung (in diesem Beispiel T3) aktualisierend wiederherzustellen.

Wenn Sie den Tabellenbereich TABER1 beispielsweise bis zum Zeitpunkt T4 wiederherstellen wollen, würden Sie den Tabellenbereich von einer Sicherung wiederherstellen, die nach T3 erstellt wurde (entweder die erforderliche Sicherung oder eine andere), und anschließend den Tabellenbereich TABER1 bis zum Ende der Protokolle aktualisierend wiederherstellen.

Im vorangegangenen Beispiel wäre die effizienteste Vorgehensweise zur Wiederherstellung des Tabellenbereichs bis zum Zeitpunkt T4 die Ausführung der erforderlichen Schritte in folgender Reihenfolge:

1. Stellen Sie die Datenbank von einer Sicherung wieder her.
2. Stellen Sie den Tabellenbereich von einer Sicherung wieder her.
3. Stellen Sie die Datenbank aktualisierend wieder her.
4. Stellen Sie den Tabellenbereich aktualisierend wieder her.

Da Sie den Tabellenbereich vor der aktualisierenden Wiederherstellung der Datenbank von einer Sicherung wiederherstellen, werden keine Ressourcen zum Anwenden der Protokollsätze auf den Tabellenbereich verwendet, wenn die Datenbank aktualisierend wiederhergestellt wird.

Wenn Sie das Sicherungsimage von TABER1 für einen Zeitpunkt nach T3 nicht mehr finden können oder den Tabellenbereich TABER1 auf einem Stand vor oder bis T3 wiederherstellen wollen, haben Sie folgende Möglichkeiten:

- Stellen Sie den Tabellenbereich aktualisierend wieder her bis zum Zeitpunkt T3. Sie brauchen den Tabellenbereich nicht erneut wiederherzustellen, weil er von der Datenbanksicherung wiederhergestellt wurde.
- Stellen Sie den Tabellenbereich erneut von dem Datenbanksicherungsimage 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 Wiederherstellung 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 Wiederherstellung erforderlich ist. Wenn Sie einen Tabellenbereich jedoch bis zu einem bestimmten Zeitpunkt aktualisierend wiederherstellen wollen, müssen Sie ihn in jeder Datenbankpartition wiederherstellen.

Zugehörige Konzepte:

- „Wiederherstellen von Daten unter Verwendung der Datei mit den Angaben zur Speicherposition der Ladekopie” auf Seite 140

Zugehörige Referenzen:

- „ROLLFORWARD DATABASE” auf Seite 144

Wiederherstellen einer gelöschten Tabelle

Es ist möglich, dass Sie versehentlich eine Tabelle löschen, deren Daten Sie weiter benötigen. In solchen Fällen empfiehlt es sich, nach dem Löschen einer Tabelle die kritischen Tabellen wiederherstellbar zu machen.

Die Tabellendaten können durch eine Datenbankwiederherstellung mit anschließender aktualisierender Wiederherstellung bis zu einem bestimmten Zeitpunkt, der vor dem Löschzeitpunkt der Tabelle liegen muss, wiederhergestellt werden. Wenn die Datenbank umfangreich ist, kann dies zeitaufwendig sein, und die Daten der Datenbank sind während der Wiederherstellung nicht verfügbar.

Die Funktion zur Wiederherstellung gelöschter Tabellen ermöglicht es, gelöschte Tabellen über eine Wiederherstellung und anschließende aktualisierende Wiederherstellung auf Tabellenbereichsebene wiederherzustellen. Dies nimmt weniger Zeit in Anspruch, als eine Wiederherstellung der gesamten Datenbank, und die Datenbank steht den Benutzern während der Wiederherstellung der Tabelle weiterhin zur Verfügung.

Vorbedingungen:

Damit eine gelöschte Tabelle wiederherstellbar ist, muss für den Tabellenbereich, in dem sich die Tabelle befindet, der Parameter DROPPED TABLE RECOVERY aktiviert sein. Dies kann während der Erstellung des Tabellenbereichs vorgenommen werden oder durch Aufrufen der Anweisung ALTER TABLESPACE. 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 Wiederherstellung gelöschter Tabellen ist für neu erstellte Tabellenbereiche standardmäßig aktiviert.

Wenn eine Anweisung DROP TABLE für eine Tabelle ausgeführt wird, deren Tabellenbereich für eine Wiederherstellung gelöschter Tabellen aktiviert ist, wird ein zusätzlicher Eintrag in den Protokolldateien vorgenommen, der die gelöschte Tabelle angibt. Außerdem wird in der Datei des Wiederherstellungsprotokolls ein Eintrag mit Informationen erstellt, die für die erneute Erstellung der Tabelle verwendet werden können.

Einschränkungen:

Für die Datentypen, die von einer gelöschten Tabelle wiederhergestellt werden können, gibt es einige Einschränkungen. Folgende Daten können nicht wiederhergestellt werden:

- LOB- oder Langfelddaten. Die Option DROPPED TABLE RECOVERY wird für LOB-Tabellenbereiche nicht unterstützt. Wenn Sie versuchen, eine gelöschte Tabelle mit Spalten des Typs LOB oder LONG VARCHAR wiederherzustellen, werden diese Spalten in der generierten Exportdatei auf NULL gesetzt. Die Option DROPPED TABLE RECOVERY kann nur für reguläre Tabellenbereiche verwendet werden, jedoch nicht für temporäre Tabellenbereiche oder LOB-Tabellenbereiche.
- 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 die Daten in der gelöschten typisierten Tabelle.

Wenn die wiederhergestellten Daten den Typ GRAPHIC oder VARGRAPHIC haben, können die Daten mehr als eine Codepage verwenden. Zur Wiederherstellung dieser Daten müssen Sie den Dateitypänderungswert USEGRAPHICCODEPAGE der Befehle IMPORT oder LOAD angeben. Wenn Sie in diesem Fall den Befehl LOAD zur Wiederherstellung der Daten verwenden, können Sie die Leistung der Wiederherstellungsoperation erhöhen.

Prozedur:

Es kann nur jeweils eine gelöschte Tabelle wiederhergestellt werden. Gehen Sie wie folgt vor, um eine gelöschte Tabelle wiederherzustellen:

1. Geben Sie die gelöschte Tabelle an, indem Sie den Befehl LIST HISTORY DROPPED TABLE aufrufen. Die ID der gelöschten Tabelle wird in der Spalte für die Sicherungs-ID ausgegeben.
2. Führen Sie die Wiederherstellung auf Datenbank- oder Tabellenbereichsebene mit einem Sicherungsimage 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 auf allen Partitionen 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 des Namens `data` exportiert. Beispiel: `\exportverzeichnis\NODE0000\data`.
4. Führen Sie eine aktualisierende Wiederherstellung 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 Wiederherstellung bis zum Ende der Protokolldateien ausführen, so dass Aktualisierungen an anderen Tabellen im Tabellenbereich oder der Datenbank nicht verloren gehen.
5. Erstellen Sie die Tabelle mit der Anweisung CREATE TABLE aus der Datei des Wiederherstellungsprotokolls erneut.
6. Importieren Sie die Daten, die während der aktualisierenden Wiederherstellung exportiert wurden, in die Tabelle.

Die Namen der verbundenen, DATALINK-Spalten zugeordneten Dateien *können* wiederhergestellt werden. Nach dem Import der Tabellendaten muss die Tabelle mit dem DB2 Data Links Manager erneut abgeglichen werden. Die Sicherungsimages der Dateien können möglicherweise vom DB2 Data Links Manager wiederhergestellt werden, je nachdem, ob diese von Garbage Collection bereits gelöscht wurden oder nicht.

Zugehörige Referenzen:

- „ALTER TABLESPACE statement" im Handbuch *SQL Reference, Volume 2*
- „CREATE TABLE statement" im Handbuch *SQL Reference, Volume 2*
- „ROLLFORWARD DATABASE" auf Seite 144
- „LIST HISTORY" auf Seite 294

Wiederherstellen von Daten unter Verwendung der Datei mit den Angaben zur Speicherposition der Ladekopie

Anhand der Registrierdatenbankvariablen DB2LOADREC wird die Datei mit den Angaben zur Speicherposition der Ladekopie identifiziert. Diese Datei wird während der aktualisierenden Wiederherstellung zum Lokalisieren der Ladekopie verwendet. Sie enthält folgende Informationen:

- Datenträgertyp
- Anzahl der zu verwendenden Datenträgereinheiten
- Speicherposition der Ladekopie, die bei einer Ladeoperation für eine Tabelle generiert wurde
- Dateiname der Ladekopie (sofern zutreffend)

Falls die Speicherpositionsdatei nicht existiert oder darin kein übereinstimmender Eintrag gefunden wurde, werden die Informationen aus dem Protokollsatz verwendet.

Die Informationen in der Datei können überschrieben werden, bevor die aktualisierende Wiederherstellung stattfindet.

Anmerkungen:

1. In einer Umgebung mit partitionierten Datenbanken muss die Registrierdatenbankvariable DB2LOADREC unter Verwendung des Befehls **db2set** für alle Datenbankpartitionsserver festgelegt werden.
2. In einer Umgebung mit partitionierten Datenbanken muss die Datei der Ladekopie auf jedem Datenbankpartitionsserver vorhanden sein, und der Dateiname (einschließlich des Pfads) muss derselbe sein.
3. Falls ein Eintrag in der Datei, die durch die Registrierdatenbankvariable DB2LOADREC angegeben ist, ungültig ist, wird der ungültige Eintrag unter Verwendung der Informationen aus der alten Datei mit den Angaben zur Speicherposition der Ladekopie ersetzt.

Die Datei mit den Angaben zur Speicherposition enthält die folgenden Informationen. Die ersten fünf Parameter müssen gültige Werte aufweisen und werden zum Identifizieren der Ladekopie verwendet. Die gesamte Struktur wird für jede aufgezeichnete Ladekopie wiederholt. Beispiel:

```
TIMestamp      19950725182542         * Zur Ladezeit generierte Zeitmarke
SCHema         PAYROLL                * Schema der geladenen Tabelle
TABlename      EMPLOYEES              * Tabellenname
DATabasename   DBT                    * Datenbankname
DB2instance    toronto                * DB2INSTANCE
BUFfernumber   NULL                   * Anzahl Puffer für die
                                        Wiederherstellung
SESsionnumber  NULL                   * Anzahl Sitzungen für die
                                        Wiederherstellung
TYPeofmedia    L                      * Datenträgertyp - L für lokale Einheit
                                                          A für TSM
                                                          O für andere Lieferanten
LOCationnumber 3                      * Anzahl Speicherpositionen
   ENTry       /u/toronto/dbt.payroll.employes.001
   ENT         /u/toronto/dbt.payroll.employes.002
   ENT         /dev/rmt0
TIM            19950725192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            A
TIM            19940325192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            O
SHRlib         /@sys/lib/backup_vendor.a
```

Anmerkungen:

1. Die ersten drei Zeichen in jedem Schlüsselwort sind wichtig. Alle Schlüsselwörter sind in der angegebenen Reihenfolge erforderlich. Leerzeilen werden nicht akzeptiert.

2. Die Zeitmarke hat das Format *jjjjmmttssmmss* (Jahr Monat Tag Stunde Minute Sekunde).
3. Alle Felder sind verbindlich, mit Ausnahme von BUF und SES, die den Wert NULL haben können. Falls SES Null ist, wird der durch den Konfigurationsparameter *numloadrecses* angegebene Wert verwendet. Ist BUF Null, lautet der Standardwert SES+2.
4. Falls auch nur einer der der Einträge in der Datei mit den Angaben zur Speicherposition ungültig ist, werden die entsprechenden Werte aus der früheren Datei mit den Angaben zur Speicherposition der Ladekopie verwendet.
5. Es gibt folgende Datenträgertypen: lokale Einheit (`L` für Band, Platte oder Diskette), TSM (`A`) oder Datenträger anderer Lieferanten (`0`). Lautet der Typ `L`, ist die Anzahl der Speicherpositionen, gefolgt von den Einträgen zur Speicherposition, erforderlich. Lautet der Typ `A`, ist keine weitere Eingabe erforderlich. Lautet der Typ `0`, ist der Name der gemeinsam benutzten Bibliothek erforderlich.
6. Der Parameter SHRlib zeigt auf eine Bibliothek, die zum Speichern der Ladekopiedaten dient.
7. Wenn Sie eine Ladeoperation aufrufen und hierbei die Option COPY NO oder NONRECOVERABLE angeben und nach Abschluss der Operation keine Sicherungskopie der Datenbank oder der betroffenen Tabellenbereiche erstellen, können Sie die Datenbank oder die Tabellenbereiche nicht für einen Zeitpunkt wiederherstellen, der nach der Ladeoperation liegt. Das heißt, dass Sie keine aktualisierende Wiederherstellung ausführen können, um die Datenbank oder die Tabellenbereiche auf dem Stand wiederherzustellen, der nach der Ladeoperation vorlag. Sie können die Datenbank oder die Tabellenbereiche lediglich für einen Zeitpunkt vor der Ladeoperation wiederherstellen.

Für den Fall, dass Sie eine bestimmte Ladekopie verwenden wollen, können Sie in der Datei des Wiederherstellungsprotokolls für die Datenbank die Zeitmarke für diese spezifische Ladeoperation ermitteln. In einer Umgebung mit partitionierten Datenbanken ist die Datei des Wiederherstellungsprotokolls für jede Datenbankpartition lokal.

Zugehörige Referenzen:

• Anhang F, „Tivoli Storage Manager", auf Seite 365

Synchronisieren der Systemuhren in einem System 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 Wiederherstellung 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.

Zur Sicherstellung, dass die Zeitmarken der Protokollsätze die Reihenfolge von Transaktionen in einem System mit partitionierten Datenbanken richtig wiedergeben, verwendet DB2® die Systemuhr 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 fälschlicherweise auf den 7. November 1999 gesetzt wird, obwohl das Jahr 1997 ist, und nehmen Sie weiter an, dass der Fehler korrigiert wurde, *nachdem* eine Transaktion zur Aktualisierung in der Partition auf diesem Datenbankpartitionsserver festgeschrieben wurde. Wenn die Datenbank ständig verwendet und regelmäßig aktualisiert wird, bleibt jeder Zeitpunkt zwischen dem 7. November 1997 und dem 7. November 1999 durch die aktualisierende Wiederherstellung praktisch unerreichbar. Wenn die Festschreibung auf Datenbankpartitionsserver A erfolgt ist, wird die Zeitmarke im Datenbankprotokoll auf 1999 gesetzt und die Uhr des Protokolls bleibt auf dem 7. November 1999 stehen, bis die Systemuhr diesen Zeitpunkt erreicht. Wenn Sie versuchen, eine aktualisierende Wiederherstellung bis zu einem Zeitpunkt innerhalb dieses Zeitrahmens durchzuführen, stoppt die Operation an der ersten Zeitmarke, die über den angegebenen Stoppzeitpunkt, 7. November 1997, hinausgeht.

Obwohl DB2 die Aktualisierung der Systemuhr nicht steuern kann, verringert der Konfigurationsparameter *max_time_diff* des Datenbankmanagers die Möglichkeit, dass diese Art von Problem auftritt:

- Die konfigurierbaren Werte für diesen Parameter reichen von 1 Minute bis zu 24 Stunden.
- Wenn die erste Verbindungsanforderung an einen Nichtkatalogknoten erfolgt, sendet der Datenbankpartitionsserver seine Zeit an den Katalogknoten für die Datenbank. Der Katalogknoten überprüft, ob die Zeit auf dem Knoten, der die Verbindung anfordert, und seine 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.

Zugehörige Referenzen:

- „max_time_diff - Maximale Zeitdifferenz zwischen Knoten" im Handbuch *Systemverwaltung: Optimierung*

Konvertieren von Client/Server-Zeitmarken

In diesem Abschnitt wird die Generierung von Zeitmarken in einer Client/Server-Umgebung erläutert:

- Wenn Sie für eine aktualisierende Wiederherstellung eine Ortszeit angeben, werden alle Nachrichten ebenfalls in Ortszeit zurückgegeben.

Anmerkung: Alle Zeitangaben werden auf dem Server und (in Umgebungen mit partitionierten Datenbanken) auf dem Katalogknoten konvertiert.

- Die Zeichenfolge der Zeitmarke wird auf dem Server in GMT konvertiert, so dass die Zeitmarke die Zeitzone des Servers und nicht 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 aufgrund der Sommerzeit nahe am Zeitwechsel liegt, muss klar sein, ob die Stoppzeit vor oder nach dem Zeitwechsel liegt, damit sie korrekt angegeben wird.

Zugehörige Konzepte:

- „Aktualisierende Wiederherstellung - Übersicht" auf Seite 129
- „Synchronisieren der Systemuhren in einem System mit partitionierten Datenbanken" auf Seite 142

ROLLFORWARD DATABASE

Stellt durch Anwendung von in den Datenbankprotokolldateien aufgezeichneten Transaktionen eine Datenbank wieder her. Wird nach der Wiederherstellung eines Datenbank- oder Tabellenbereichssicherungsimages aufgerufen bzw. wenn die Datenbank aufgrund eines Datenträgerfehlers Tabellenbereiche in den Offlinestatus versetzt. Die Datenbank muss wiederherstellbar sein, (d. h., der Datenbankkonfigurationsparameter *logarchmeth1* oder *logarchmeth2* muss auf einen anderen Wert als `OFF` gesetzt sein), bevor die Datenbank aktualisierend wiederhergestellt werden kann.

Geltungsbereich:

In einer Umgebung mit partitionierten Datenbanken kann dieser Befehl nur von der Katalogtabellenpartition aus aufgerufen werden. Eine aktualisierende Wiederherstellungsoperation für Datenbanken oder Tabellenbereiche bis zu einem bestimmten Zeitpunkt wirkt sich auf alle Partitionen aus, die in der Datei `db2nodes.cfg` aufgelistet sind. Eine aktualisierende Wiederherstellungsoperation für Datenbanken oder Tabellenbereiche bis zum Ende der Protokolle wirkt sich auf die Partitionen aus, die angegeben werden. Wenn keine Partitionen angegeben werden, wirkt sich die Operation auf alle Partitionen aus, die in der Datei `db2nodes.cfg` aufgelistet sind. Wenn für eine bestimmte Partition keine aktualisierende Wiederherstellung erforderlich ist, wird diese Partition ignoriert.

Berechtigung:

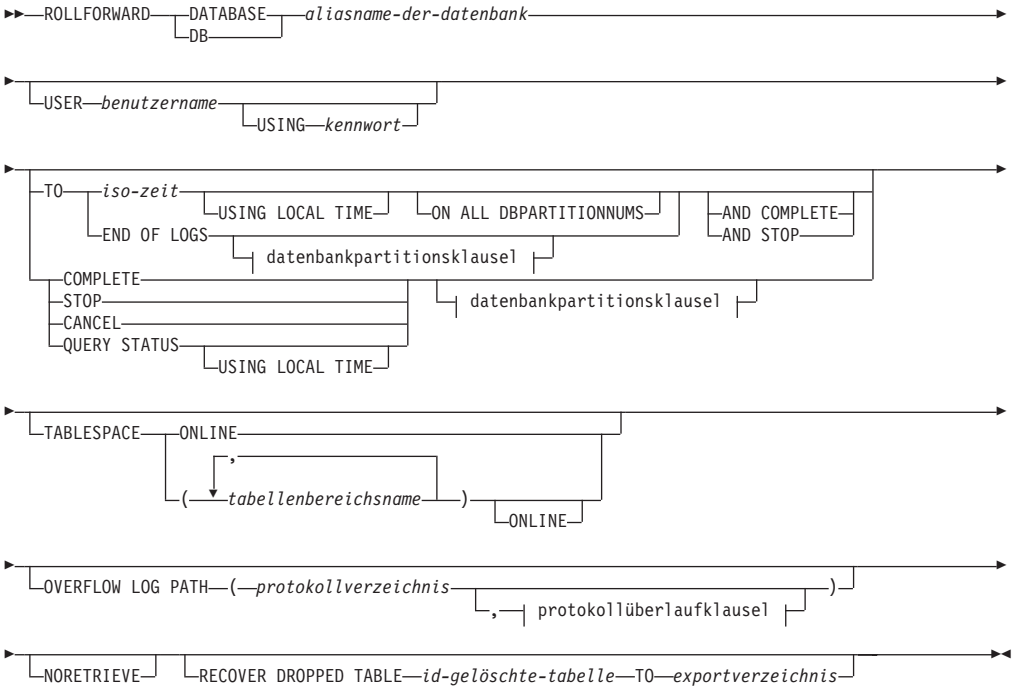
Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

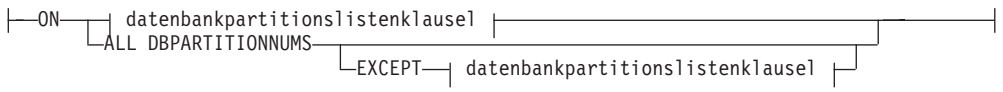
Erforderliche Verbindung:

Keine. Dieser Befehl stellt eine Datenbankverbindung her.

Befehlssyntax:



Datenbankpartitionsklausel:



Datenbankpartitionslistenklausel:

Protokollüberlaufklausel:

Befehlsparameter:

DATABASE aliasname-der-datenbank
Der Aliasname der Datenbank, die aktualisierend wiederhergestellt werden soll.

USER benutzername
Der Benutzername, unter dem die Datenbank aktualisierend wiederhergestellt werden soll.

USING kennwort
Das Kennwort, das verwendet wird, um den Benutzernamen zu authentifizieren. Wenn kein Kennwort angegeben wird, wird der Benutzer zur Eingabe des Kennworts aufgefordert.

TO

iso-zeit
Der Zeitpunkt, bis zu dem alle festgeschriebenen Transaktionen aktualisierend wiederhergestellt werden sollen (einschließlich der Transaktion, die genau zu diesem Zeitpunkt festgeschrieben wurde, und aller vorher festgeschriebenen Transaktionen). Dieser Wert wird als Zeitmarke angegeben, eine 7-teilige Zeichenfolge, die eine Kombination aus Datum und Zeit angibt. Das Format ist *jjjj-mm-tt-hh.mm.ss.nnnnnn* (Jahr, Monat, Tag, Stunde, Minuten, Sekunden, Mikrosekunden) und wird in Weltzeit (Coordinated Universal Time, UTC) ausgedrückt. Die Weltzeit hilft zu verhindern, dass verschiedenen Protokollen dieselbe Zeitmarke zugeordnet wird (z. B. wegen einer Zeitänderung aufgrund der Sommerzeit). Die Zeitmarke in einem Sicherungsimage basiert auf der Ortszeit, zu der die Sicherungsoperation gestartet wurde. Das Sonderregister CURRENT TIMEZONE gibt den Unterschied zwischen UTC und der Ortszeit auf dem Anwendungsserver an. Der Unterschied wird als Zeitdifferenz dargestellt (eine Dezimalzahl, in der die ersten beiden Ziffern die Anzahl Stunden, die nächsten beiden Ziffern die Anzahl Minuten und die letzten beiden Ziffern die Anzahl Sekunden darstellen). Indem CURRENT TIMEZONE von einer Ortszeit abgezogen wird, wird diese Ortszeit in Weltzeit konvertiert.

USING LOCAL TIME
Ermöglicht dem Benutzer die aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt in Ortszeit anstatt in WEZ. Dies erleichtert Benutzern die aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt auf ihren lokalen Maschinen und vermeidet potenzielle Benutzerfehler aufgrund der Umsetzung von Ortszeit in WEZ.

Anmerkungen:
1. Wenn der Benutzer für eine aktualisierende Wiederherstellung eine Ortszeit angibt, werden alle Nachrichten ebenfalls in Ortszeit an den Benutzer zurückgegeben. Alle Zeitangaben werden auf dem Server und (in Umgebungen mit partitionierten Datenbanken) auf der Katalogdatenbankpartition konvertiert.
2. Die Zeichenfolge der Zeitmarke wird auf dem Server in WEZ konvertiert, so dass die Zeitmarke die Zeitzone des Servers und nicht des Clients wiedergibt. Wenn sich der Client in einer anderen Zeitzone befindet als der Server, sollte die Ortszeit des Servers verwendet werden. Dies unterscheidet sich von der Ortszeitoption der Steuerzentrale, die die Ortszeit des Clients verwendet.
3. Wenn die Zeichenfolge der Zeitmarke aufgrund der Sommerzeit nahe am Zeitwechsel liegt, muss klar sein, ob die Stoppzeit vor oder nach dem Zeitwechsel liegt, damit sie korrekt angegeben wird.

END OF LOGS
Gibt an, dass alle festgeschriebenen Transaktionen aus allen Onlinearchivprotokolldateien angewendet werden sollen, die in dem durch den Datenbankkonfigurationsparameter *logpath* angegebenen Pfad enthalten sind.

ALL DBPARTITIONNUMS
Gibt an, dass Transaktionen auf allen in der Datei `db2nodes.cfg` angegebenen Partitionen aktualisierend wiederhergestellt werden sollen. Dies ist die Standardeinstellung, falls keine Datenbankpartitionsklausel angegeben ist.

EXCEPT
Gibt an, dass Transaktionen auf allen in der Datei `db2nodes.cfg` angegebenen Partitionen aktualisierend wiederhergestellt werden sollen, mit Ausnahme der Partitionen in der Datenbankpartitionsliste.

ON DBPARTITIONNUM / ON DBPARTITIONNUMS
Stellt die Datenbank in einer Datenbankpartitionsgruppe aktualisierend wieder her.

db-partitionsnummer1
Gibt eine Datenbankpartitionsnummer in der Datenbankpartitionsliste an.

db-partitionsnummer2
Gibt die zweite Datenbankpartitionsnummer an, so dass alle Partitionen von *db-partitionsnummer1* bis einschließlich *db-partitionsnummer2* in die Datenbankpartitionsliste aufgenommen werden.

COMPLETE / STOP
Stoppt die aktualisierende Wiederherstellung von Protokollsätzen, und beendet den Prozess zur aktualisierenden Wiederherstellung, indem alle unvollständigen Transaktionen rückgängig gemacht werden und der Status *Aktualisierende Wiederherstellung anstehend* der Datenbank inaktiviert wird. Dadurch erhalten Sie Zugriff auf die Datenbank oder auf Tabellenbereiche, die aktualisierend wiederhergestellt werden. Diese Schlüsselwörter sind gleichbedeutend. Geben Sie eins von beiden an, aber nicht beide gleichzeitig. Mit dem Schlüsselwort AND können Sie mehrere Operationen gleichzeitig angeben, z. B. `db2 rollforward db sample to end of logs and complete`.

Anmerkung: Wenn Sie Tabellenbereiche bis zu einem Zeitpunkt aktualisierend wiederherstellen, werden die Tabellenbereiche in den Status *Sicherung anstehend* versetzt.

CANCEL
Bricht die aktualisierende Wiederherstellungsoperation ab. Dadurch wird die Datenbank oder mindestens ein Tabellenbereich auf allen Partitionen, auf denen die aktualisierende Wiederherstellung gestartet wurde, in den Status *Wiederherstellung anstehend* versetzt:
- Wenn keine aktualisierende Wiederherstellungsoperation für eine Datenbank ausgeführt wird, die Datenbank sich also im Status *Aktualisierende Wiederherstellung anstehend* befindet, versetzt diese Option die Datenbank in den Status *Wiederherstellung anstehend*.
- Wenn keine aktualisierende Wiederherstellungsoperation für einen Tabellenbereich ausgeführt wird, die Tabellenbereiche sich also im Status *Aktualisierende Wiederherstellung anstehend* befinden, muss eine Tabellenbereichsliste angegeben werden. Alle Tabellenbereiche in der Liste werden in den Status *Wiederherstellung anstehend* versetzt.

- Wenn eine aktualisierende Tabellenbereichswiederherstellung *ausgeführt* wird, sich also mindestens ein Tabellenbereich im Status *Aktualisierende Wiederherstellung läuft* befindet, werden alle Tabellenbereiche, die sich im Status *Aktualisierende Wiederherstellung läuft* befinden, in den Status *Wiederherstellung anstehend* versetzt. Wenn eine Tabellenbereichsliste angegeben wird, muss sie alle Tabellenbereiche enthalten, die sich im Status *Aktualisierende Wiederherstellung läuft* befinden. Alle Tabellenbereiche auf der Liste werden in den Status *Wiederherstellung anstehend* versetzt.
- Wenn eine aktualisierende Wiederherstellung bis zu einem Zeitpunkt ausgeführt wird, werden alle übergebenen Tabellenbereichsnamen ignoriert und alle Tabellenbereiche, die sich im Status *Aktualisierende Wiederherstellung läuft* befinden, werden in den Status *Wiederherstellung anstehend* versetzt.
- Wenn eine aktualisierende Wiederherstellung bis zum Ende der Protokolle mit einer Tabellenbereichsliste ausgeführt wird, werden nur die aufgelisteten Tabellenbereiche in den Status *Wiederherstellung anstehend* versetzt.

Diese Option kann nicht verwendet werden, um eine aktualisierende Wiederherstellungsoperation abzubrechen, die *tatsächlich ausgeführt wird*. Die Option kann nur verwendet werden, um eine aktualisierende Wiederherstellungsoperation abzubrechen, die läuft, aber zurzeit nicht ausgeführt wird. Eine aktualisierende Wiederherstellungsoperation kann unter folgenden Umständen laufen, ohne ausgeführt zu werden:

- Sie wurde abnormal beendet.
- Die Option STOP wurde nicht angegeben.
- Sie ist aufgrund eines Fehlers fehlgeschlagen. Einige Fehler, wie z. B. die aktualisierende Wiederherstellung durch eine nicht wiederherstellbare Ladeoperation, können einen Tabellenbereich in den Status *Wiederherstellung anstehend* versetzen.

Anmerkung: Verwenden Sie diese Option mit Bedacht und nur dann, wenn die laufende aktualisierende Wiederherstellungsoperation nicht beendet werden kann, weil einige der Tabellenbereiche in den Status *Aktualisierende Wiederherstellung anstehend* oder *Wiederherstellung anstehend* versetzt worden sind. Zeigen Sie im Zweifelsfall mit dem Befehl LIST TABLESPACES die Tabellenbereiche an, die sich im Status *Aktualisierende Wiederherstellung läuft* oder *Aktualisierende Wiederherstellung anstehend* befinden.

QUERY STATUS

Listet die Protokolldateien, die der Datenbankmanager aktualisierend wiederhergestellt hat, die nächste erforderliche Archivierungsdatei und die Zeitmarke (in UTC) der letzten, seit dem Start der Verarbeitung der aktualisierenden Wiederherstellung festgeschriebenen Transaktion auf. In einer Umgebung mit partitionierten Datenbanken werden diese Statusinformationen für alle Partitionen zurückgegeben. Die zurückgegebenen Informationen umfassen die folgenden Felder:

Datenbankpartitionsnummer

Status der aktualisierenden Wiederherstellung
Es gibt die folgenden Status: *Aktualisierende Wiederherstellung anstehend* für Datenbank oder Tabellenbereich, *Aktualisierende Wiederherstellung läuft* für Datenbank oder Tabellenbereich, *Aktualisierende Wiederherstellung stoppt* für Datenbank oder Tabellenbereich oder *Nicht anstehend*.

Nächste zu lesende Protokolldatei
Eine Zeichenfolge, die den Namen der nächsten erforderlichen Protokolldatei enthält. Verwenden Sie diese Informationen in einer Umgebung mit partitionierten Datenbanken, wenn das Dienstprogramm zur aktualisierenden Wiederherstellung mit einem Rückkehrcode fehlschlägt, der anzeigt, dass eine Protokolldatei fehlt oder dass eine Abweichung der Protokolldaten aufgetreten ist.

Verarbeitete Protokolldateien
Eine Zeichenfolge, die die Namen der verarbeiteten Protokolldateien enthält, die nicht mehr für die Wiederherstellung erforderlich sind und aus dem Verzeichnis gelöscht werden können. Wenn z. B. die älteste nicht festgeschriebene Transaktion in der Protokolldatei *x* beginnt, umfasst der Bereich der veralteten Protokolldateien *x* nicht. Der Bereich endet bei *x* - 1.

Letzte festgeschriebene Transaktion
Eine Zeichenfolge, die eine Zeitmarke in ISO-Format enthält (*jjjj-mm-tt-hh.mm.ss*). Diese Zeitmarke markiert die letzte festgeschriebene Transaktion nach der Fertigstellung der aktualisierenden Wiederherstellung. Die Zeitmarke wird auf die Datenbank angewendet. Bei der aktualisierenden Tabellenbereichswiederherstellung ist dieser Wert die Zeitmarke der letzten in der Datenbank festgeschriebenen Transaktion.

Anmerkung: Der Standardwert ist QUERY STATUS, wenn die Klauseln TO, STOP, COMPLETE oder CANCEL nicht angegeben werden. Wenn TO, STOP oder COMPLETE angegeben wurden, werden nach erfolgreicher Beendigung des Befehls Statusinformationen angezeigt. Wenn einzelne Tabellenbereiche angegeben werden, werden sie ignoriert. Die Statusanforderung gilt nicht nur für angegebene Tabellenbereiche.

TABLESPACE
Dieses Schlüsselwort wird für die aktualisierende Wiederherstellung auf Tabellenbereichsebene angegeben.

tabellenbereichsname
Verbindlich für eine aktualisierende Wiederherstellung auf Tabellenbereichsebene bis zu einem Zeitpunkt. Ermöglicht die Angabe einer Untermenge der Tabellenbereiche, die bis zum Ende der Protokolle aktualisierend wiederhergestellt werden sollen. In einer Umgebung mit partitionierten Datenbanken muss nicht jeder Tabellenbereich in der Liste auf allen Partitionen vorhanden sein, die aktualisierend wiederhergestellt werden. Wenn er vorhanden *ist*, muss er sich im korrekten Status befinden.

ONLINE
Dieses Schlüsselwort wird angegeben, damit die aktualisierende Wiederherstellung auf Tabellenbereichsebene online ausgeführt werden kann. Das heißt, dass andere Agenten eine Verbindung herstellen dürfen, während die aktualisierende Wiederherstellung läuft.

OVERFLOW LOG PATH protokollverzeichnis
Gibt einen alternativen Protokollpfad an, der während der Wiederherstellung nach archivierten Protokollen durchsucht wird. Verwenden Sie diesen Parameter, wenn Protokolldateien an eine andere als die im Datenbankkonfigurationsparameter *logpath* angegebene Position versetzt wurden. In einer Umgebung mit partitionierten Datenbanken ist dies der (vollständig quali-

fizierte) Standardüberlaufprotokollpfad *für alle Partitionen*. Für Datenbanken mit einer Partition kann ein relativer Überlaufprotokollpfad angegeben werden.

Anmerkung: Der Befehlsparameter OVERFLOW LOG PATH überschreibt den Wert (falls vorhanden) des Datenbankkonfigurationsparameters OVERFLOWLOGPATH.

protokollverzeichnis ON DBPARTITIONNUM
In einer Umgebung mit partitionierten Datenbanken kann ein anderer Protokollpfad den Standardüberlaufprotokollpfad für eine bestimmte Partition überschreiben.

NORETRIEVE
Ermöglicht dem Benutzer zu steuern, welche Protokolldateien auf der Bereitschaftsmaschine aktualisierend wiederhergestellt werden, indem der Benutzer das Abrufen archivierter Protokolle inaktivieren kann. Dies bietet folgende Vorteile:

- Durch diese Steuerung der aktualisierend wiederherzustellenden Protokolldateien kann sichergestellt werden, dass sich die Bereitschaftsmaschine zeitlich `X` Stunden hinter der Produktionsmaschine befindet, so dass sich Benutzeraktionen nicht auf beide Systeme auswirken.
- Wenn das fehlertolerante System (d h. das Bereitschaftssystem) keinen Zugriff auf das Archiv hat (wenn beispielsweise TSM das Archiv ist, erlaubt es nur der Ursprungsmaschine, die Dateien abzurufen).
- Es ist möglich, dass das Produktionssystem eine Datei archiviert und das fehlertolerante System währenddessen dieselbe Datei abruft und nur eine unvollständige Protokolldatei erhält. Dieses Problem kann mit NORETRIEVE behoben werden.

RECOVER DROPPED TABLE id-gelöschte-tabelle
Stellt eine gelöschte Tabelle während der aktualisierenden Wiederherstellungsoperation wieder her. Die Tabellen-ID kann mit dem Befehl LIST HISTORY abgerufen werden.

TO exportverzeichnis
Gibt ein Verzeichnis an, in das Dateien geschrieben werden sollen, die die Tabellendaten enthalten. Alle Datenbankpartitionen müssen auf das Verzeichnis zugreifen können.

Beispiele:

Beispiel 1

Mit dem Befehl ROLLFORWARD DATABASE können Sie mehrere Operationen gleichzeitig spezifizieren, wobei diese durch das Schlüsselwort `AND` getrennt sind. Sie benötigen z. B. die folgenden Einzelbefehle, um eine aktualisierende Wiederherstellung 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, dass Sie überprüfen, ob die aktualisierende Wiederherstellung wie erwartet fortgeschritten ist, bevor Sie sie stoppen und möglicherweise Protokolle fehlen. Dies ist besonders wichtig, wenn

während der aktualisierenden Wiederherstellung ein beschädigtes Protokoll gefunden wird und dieses Protokoll als „Protokollende" interpretiert wird. In solchen Fällen könnte eine unbeschädigte Sicherungskopie dieses Protokolls verwendet werden, um die aktualisierende Wiederherstellung mit weiteren Protokollen fortzusetzen.

Beispiel 2

Führen Sie eine aktualisierende Wiederherstellung bis zum Protokollende aus (zwei Tabellenbereiche wurden wiederhergestellt):

```
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 Tabellenbereichswiederherstellung bis zum Protokollende erforderlich. Tabellenbereichsnamen sind nicht erforderlich. Wenn keine Bereiche angegeben sind, werden alle Tabellenbereiche eingeschlossen, die eine aktualisierende Wiederherstellung erfordern. Wenn nur eine Untermenge dieser Tabellenbereiche aktualisierend wiederhergestellt werden sollen, müssen die Namen der Tabellenbereiche angegeben werden.

Beispiel 3

Nachdem drei Tabellenbereiche wiederhergestellt worden sind, stellen Sie wie folgt einen Bereich aktualisierend bis zum Ende der Protokolle wieder her und die anderen beiden Bereiche bis zu einem Zeitpunkt. Beide Aktionen sollen online ausgeführt werden.

```
db2 rollforward db sample to end of logs tablespace(TBS1) online

db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS2, TBS3) online
```

Beachten Sie, dass zwei aktualisierende Wiederherstellungsoperationen nicht gleichzeitig ausgeführt werden können. Sie können den zweiten Befehl erst aufrufen, nachdem die erste aktualisierende Wiederherstellungsoperation erfolgreich beendet wurde.

Beispiel 4

Nachdem die Datenbank wiederhergestellt wurde, führen Sie wie folgt eine aktualisierende Wiederherstellung bis zu einem Zeitpunkt aus, wobei Sie OVERFLOW LOG PATH zur Angabe des Verzeichnisses verwenden, in dem der Benutzerexit Archivprotokolldateien speichert:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   overflow log path (/logs)
```

Beispiel 5 (Umgebungen mit partitionierten Datenbanken)

Es gibt drei Datenbankpartitionen: 0, 1 und 2. Der Tabellenbereich TBS1 ist auf allen Partitionen definiert, der Tabellenbereich TBS2 nur auf den Partitionen 0 und 2. Nachdem Sie die Datenbank auf Datenbankpartition 1 und TBS1 auf den Datenbankpartitionen 0 und 2 wiederhergestellt haben, stellen Sie die Datenbank auf Datenbankpartition 1 wie folgt aktualisierend wieder her:

```
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 ″knotenliste″ ist jedoch mindestens ein Tabellenbereich offline.“) zurück.

```
db2 rollforward db sample to end of logs
```

Durch diesen Befehl wird TBS1 auf den Datenbankpartitionen 0 und 2 aktualisierend wiederhergestellt. In diesem Fall ist die Klausel TABLESPACE(TBS1) optional.

Beispiel 6 (Umgebungen mit partitionierten Datenbanken)

Nachdem Sie Tabellenbereich TBS1 nur auf den Datenbankpartitionen 0 und 2 wiederhergestellt haben, stellen Sie TBS1 wie folgt auf 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 Wiederherstellung bereit ist. Der Befehl gibt SQL4906N aus.

```
db2 rollforward db sample to end of logs on dbpartitionnums (0, 2)
   tablespace(TBS1)
```

Dieser Befehl wird erfolgreich beendet.

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

Dieser Befehl schlägt fehl, weil TBS1 auf Datenbankpartition 1 nicht für eine aktualisierende Wiederherstellung bereit ist. Alle Teile müssen gemeinsam aktualisierend wiederhergestellt werden.

Anmerkung: Bei der aktualisierenden Tabellenbereichswiederherstellung bis zu einem Zeitpunkt wird die Datenbankpartitionsklausel nicht akzeptiert. Die aktualisierende Wiederherstellungsoperation muss auf allen Datenbankpartitionen ausgeführt werden, auf denen sich der Tabellenbereich befindet.

Geben Sie nach der Wiederherstellung von TBS1 auf Datenbankpartition 1 Folgendes ein:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

Dieser Befehl wird erfolgreich beendet.

Beispiel 7 (Umgebung mit partitionierten Datenbanken)

Nachdem Sie einen Tabellenbereich auf allen Datenbankpartitionen wiederhergestellt haben, stellen Sie ihn wie folgt aktualisierend bis zum Zeitpunkt PIT2 wieder her, aber geben Sie nicht `AND STOP` an. Die aktualisierende Wiederherstellungsoperation läuft immer noch. Brechen Sie sie wie folgt ab, und führen Sie eine aktualisierende Wiederherstellung bis zum Zeitpunkt PIT1 aus:

```
db2 rollforward db sample to pit2 tablespace(TBS1)
db2 rollforward db sample cancel tablespace(TBS1)

** TBS1 auf allen Datenbankpartitionen wiederherstellen **
```

```
db2 rollforward db sample to pit1 tablespace(TBS1)
db2 rollforward db sample stop tablespace(TBS1)
```

Beispiel 8 (Umgebungen mit partitionierten Datenbanken)

Stellen Sie wie folgt einen Tabellenbereich wieder her, der sich auf 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, auf denen sich die Tabellenbereiche befinden, müssen nicht angegeben werden. Das Dienstprogramm verwendet standardmäßig die Datei db2nodes.cfg.

Beispiel 9 (Umgebung mit partitionierten Datenbanken)

Stellen Sie wie folgt sechs kleine Tabellenbereiche aktualisierend wieder her, die sich auf einer Datenbankpartitionsgruppe mit nur einer Partition befinden (auf 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.

Hinweise:

Wenn Sie eine Wiederherstellung von einem Image ausführen, das während einer Onlinesicherungsoperation erstellt wurde, muss der für die aktualisierende Wiederherstellung angegebene Zeitpunkt nach dem Zeitpunkt liegen, zu dem die Onlinesicherungsoperation beendet wurde. Wenn die aktualisierende Wiederherstellung gestoppt wird, bevor sie diesen Punkt überschreitet, behält sie den Status *Aktualisierende Wiederherstellung anstehend*. Wenn die aktualisierende Wiederherstellung eines Tabellenbereichs gerade ausgeführt wird, behält er den Status *Aktualisierende Wiederherstellung läuft*.

Wenn mindestens ein Tabellenbereich bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt wird, muss die aktualisierende Wiederherstellung mindestens bis zu der Mindestwiederherstellungszeit fortgesetzt werden, d. h. bis zur letzten Aktualisierung der Systemkataloge für diesen Tabellenbereich oder dessen Tabellen. Die Mindestwiederherstellungszeit (angegeben in Weltzeit) für einen Tabellenbereich kann mit dem Befehl LIST TABLESPACES SHOW DETAIL abgerufen werden.

Eine aktualisierende Wiederherstellung von Datenbanken könnte eine Wiederherstellung mit Bandeinheiten erfordern. Falls ein anderes Band angefordert wird, kann der Benutzer mit einer der folgenden Optionen antworten:

c Fortsetzen. Verwendung der Einheit fortsetzen, die die Warnung generiert hat (zum Beispiel, wenn ein neues Band eingelegt wurde).

d Einheit beenden. Die Verwendung der Einheit beenden, die die Warnung generiert hat (zum Beispiel, wenn keine Bänder mehr vorhanden sind).

t Beenden. Alle Einheiten beenden.

Wenn das Dienstprogramm zur aktualisierenden Wiederherstellung das nächste benötigte Protokoll nicht finden kann, wird der Protokollname im SQL-Kommunikationsbereich zurückgegeben, und die aktualisierende Wiederherstellung stoppt. Wenn keine weiteren Protokolle verfügbar sind, verwenden Sie die Option STOP, um die aktualisierende Wiederherstellung zu beenden. Unvollständige Transaktionen werden rückgängig gemacht, um sicherzustellen, dass die Datenbank oder der Tabellenbereich in einem konsistenten Status bleibt.

Kompatibilität:

Zur Kompatibilität mit älteren Versionen als Version 8 ist Folgendes möglich:

- Das Schlüsselwort NODE kann DBPARTITIONNUM ersetzen.
- Das Schlüsselwort NODES kann DBPARTITIONNUMS ersetzen.

Zugehörige Referenzen:

- „BACKUP DATABASE" auf Seite 79
- „RESTORE DATABASE" auf Seite 102

db2Rollforward - Datenbank aktualisierend wiederherstellen

Stellt durch Anwendung von in den Datenbankprotokolldateien aufgezeichneten Transaktionen eine Datenbank wieder her. Wird nach der Wiederherstellung einer Datenbank- oder Tabellenbereichssicherung aufgerufen bzw. wenn die Datenbank aufgrund eines Datenträgerfehlers Tabellenbereiche in den Offlinestatus versetzt. Die Datenbank muss wiederherstellbar sein, (d. h., der Datenbankkonfigurationsparameter *logarchmeth1* muss auf `on` gesetzt sein), bevor die Datenbank aktualisierend wiederhergestellt werden kann.

Geltungsbereich:

In einer Umgebung mit partitionierten Datenbanken kann diese API nur von der Katalogtabellenpartition aus aufgerufen werden. Ein Aufruf für eine Wiederherstellung einer Datenbank oder eines Tabellenbereichs bis zu einem bestimmten Zeitpunkt wirkt sich auf alle Datenbankpartitionsserver aus, die in der Datei `db2nodes.cfg` aufgelistet sind. Ein Aufruf für eine Wiederherstellung einer Datenbank oder eines Tabellenbereichs bis zum Ende der Protokolle wirkt sich auf die angegebenen Datenbankpartitionsserver aus. Falls keine Datenbankpartitionsserver angegeben sind, wirkt sich der Aufruf auf alle Datenbankpartitionsserver aus, die in der Datei `db2nodes.cfg` aufgelistet sind. Falls für einen bestimmten Datenbankpartitionsserver keine aktualisierende Wiederherstellung erforderlich ist, wird dieser Datenbankpartitionsserver ignoriert.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Erforderliche Verbindung:

Keine. Diese API stellt eine Datenbankverbindung her.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2Rollforward */
/* ... */
SQL_API_RC SQL_API_FN
db2Rollforward (
  db2Uint32 versionNumber,
  void *pDB2RollforwardStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2RollforwardStruct
{
  struct db2RfwdInputStruct   *piRfwdInput;
  struct db2RfwdOutputStruct  *poRfwdOutput;
} db2RollforwardStruct;

typedef SQL_STRUCTURE db2RfwdInputStruct
{
  sqluint32                   iVersion;
  char                        *piDbAlias;
  db2Uint32                   iCallerAction;
  char                        *piStopTime;
  char                        *piUserName;
   char                *piPassword;
  char                        *piOverflowLogPath;
  db2Uint32                   iNumChngLgOvrflw;
  struct sqlurf_newlogpath    *piChngLogOvrflw;
  db2Uint32                   iConnectMode;
  struct sqlu_tablespace_bkrst_list *piTablespaceList;
  db2int32                    iAllNodeFlag;
  db2int32                    iNumNodes;
  SQL_PDB_NODE_TYPE           *piNodeList;
  db2int32                    iNumNodeInfo;
  char                        *piDroppedTblID;
  char                        *piExportDir;
  db2Uint32                   iRollforwardFlags;
} db2RfwdInputStruct;

typedef SQL_STRUCTURE db2RfwdOutputStruct
{
  char                        *poApplicationId;
  sqlint32                    *poNumReplies;
  struct sqlurf_info          *poNodeInfo;
} db2RfwdOutputStruct;

typedef SQL_STRUCTURE sqlurf_newlogpath
{
  SQL_PDB_NODE_TYPE           nodenum;
   unsigned short             pathlen;
   char                       logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
} sqlurf_newlogpath;

typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
  long                         num_entry;
  struct sqlu_tablespace_entry *tablespace;
} sqlu_tablespace_bkrst_list;

typedef SQL_STRUCTURE sqlu_tablespace_entry
{
  sqluint32                   reserve_len;
  char                        tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
```

```
  char                          filler[1];
} sqlu_tablespace_entry;

typedef SQL_STRUCTURE sqlurf_info
{
   SQL_PDB_NODE_TYPE nodenum;
   sqlint32        state;
   unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
} sqlurf_info;
/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2Rollforward */
/* ... */
SQL_API_RC SQL_API_FN
db2gRollforward (
  db2Uint32 versionNumber,
  void *pDB2gRollforwardStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2gRollforwardStruct
{
  struct db2gRfwdInputStruct  *piRfwdInput;
  struct db2RfwdOutputStruct  *poRfwdOutput;
} db2gRollforwardStruct;

SQL_STRUCTURE db2gRfwdInputStruct
{
  db2Uint32                   iDbAliasLen;
  db2Uint32                   iStopTimeLen;
   db2Uint32 iUserNameLen;
   db2Uint32           iPasswordLen;
  db2Uint32                   iOvrflwLogPathLen;
  db2Uint32                   iDroppedTblIDLen;
  db2Uint32                   iExportDirLen;
  sqluint32                   iVersion;
  char                        *piDbAlias;
  db2Uint32                   iCallerAction;
  char                        *piStopTime;
  char                        *piUserName;
   char                *piPassword;
  char                        *piOverflowLogPath;
  db2Uint32                   iNumChngLgOvrflw;
  struct sqlurf_newlogpath    *piChngLogOvrflw;
  db2Uint32                   iConnectMode;
  struct sqlu_tablespace_bkrst_list *piTablespaceList;
  db2int32                    iAllNodeFlag;
  db2int32                    iNumNodes;
  SQL_PDB_NODE_TYPE           *piNodeList;
  db2int32                    iNumNodeInfo;
  char                        *piDroppedTblID;
  char                        *piExportDir;
  db2Uint32                   iRollforwardFlags;
} db2gRfwdInputStruct;

typedef SQL_STRUCTURE db2RfwdOutputStruct
{
  char                        *poApplicationId;
  sqlint32                    *poNumReplies;
  struct sqlurf_info          *poNodeInfo;
} db2RfwdOutputStruct;
```

```
typedef SQL_STRUCTURE sqlurf_newlogpath
{
  SQL_PDB_NODE_TYPE           nodenum;
   unsigned short             pathlen;
   char                       logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
} sqlurf_newlogpath;

typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
  long                         num_entry;
  struct sqlu_tablespace_entry *tablespace;
} sqlu_tablespace_bkrst_list;

typedef SQL_STRUCTURE sqlu_tablespace_entry
{
  sqluint32                   reserve_len;
  char                        tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
  char                        filler[1];
} sqlu_tablespace_entry;

typedef SQL_STRUCTURE sqlurf_info
{
   SQL_PDB_NODE_TYPE nodenum;
   sqlint32        state;
   unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
} sqlurf_info;
/* ... */
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter übergeben wird.

pDB2RollforwardStruct
Eingabe. Ein Zeiger auf die Struktur *db2RollforwardStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

piRfwdInput
Eingabe. Ein Zeiger auf die Struktur *db2RfwdInputStruct*.

poRfwdOutput
Ausgabe. Ein Zeiger auf die Struktur *db2RfwdOutputStruct*.

iDbAliasLen
Eingabe. Gibt die Länge des Datenbankaliasnamens in Byte an.

iStopTimeLen
Eingabe. Gibt die Länge des Stoppzeitparameters in Byte an. Wird auf null gesetzt, wenn keine Stoppzeit angegeben wird.

iUserNameLen
Eingabe. Gibt die Länge des Benutzernamens in Byte an. Wird auf null gesetzt, wenn kein Benutzername angegeben wird.

iPasswordLen
Eingabe. Gibt die Länge des Kennworts in Byte an. Wird auf null gesetzt, wenn kein Kennwort angegeben wird.

iOverflowLogPathLen
Eingabe. Gibt die Länge des Überlaufprotokollpfads in Byte an. Wird auf null gesetzt, wenn kein Überlaufprotokollpfad angegeben wird.

iVersion
Eingabe. Die Versions-ID des Parameters für eine aktualisierende Wiederherstellung. Sie ist als SQLUM_RFWD_VERSION definiert.

piDbAlias
Eingabe. Eine Zeichenfolge, die den Datenbankaliasnamen enthält. Dies ist der im Systemdatenbankverzeichnis katalogisierte Aliasname.

iCallerAction
Eingabe. Gibt die auszuführende Aktion an. Gültige Werte (definiert in db2ApiDf.h) sind:

DB2ROLLFORWARD_ROLLFWD
Eine aktualisierende Wiederherstellung bis zu dem in *piStopTime* angegebenen Zeitpunkt ausführen. Für eine aktualisierende Wiederherstellung der Datenbank bleibt die Datenbank im Status *Aktualisierende Wiederherstellung anstehend*. Für eine aktualisierende Wiederherstellung von Tabellenbereichen bis zu einem bestimmten Zeitpunkt bleiben die Tabellenbereiche im Status *Aktualisierende Wiederherstellung läuft*.

DB2ROLLFORWARD_STOP
Aktualisierende Wiederherstellung beenden. Es werden keine neuen Protokollsätze verarbeitet, und nicht festgeschriebene Transaktionen werden zurückgesetzt. Der Status *Aktualisierende Wiederherstellung anstehend* der Datenbank oder des Tabellenbereichs wird inaktiviert. Der Parameter DB2ROLLFORWARD_RFWD_COMPLETE ist gleichbedeutend.

DB2ROLLFORWARD_RFWD_STOP
Eine aktualisierende Wiederherstellung bis zu dem in *piStopTime* angegebenen Zeitpunkt ausführen und die aktualisierende Wiederherstellung beenden. Der Status *Aktualisierende Wiederherstellung anstehend* der Datenbank oder des Tabellenbereichs wird inaktiviert. Der Parameter DB2ROLLFORWARD_RFWD_COMPLETE ist gleichbedeutend.

DB2ROLLFORWARD_QUERY
Die Werte abfragen für *nextarclog*, *firstarcdel*, *lastarcdel* und *lastcommit*. Den Datenbankstatus und eine Knotenzahl zurückgeben.

DB2ROLLFORWARD_PARM_CHECK
Parameter auswerten, ohne die aktualisierende Wiederherstellung auszuführen.

DB2ROLLFORWARD_CANCEL
Die zurzeit ausgeführte aktualisierende Wiederherstellungsoperation abbrechen. Die Datenbank oder der Tabellenbereich wird in den Status *Wiederherstellung anstehend* versetzt.

Anmerkung: Diese Option kann nicht verwendet werden, während die aktualisierende Wiederherstellung tatsächlich läuft. Sie kann verwendet werden, wenn die aktualisierende Wiederherstellung angehalten ist (d. h. auf ein STOP wartet) oder wenn während der

aktualisierenden Wiederherstellung ein Systemfehler aufgetreten ist. Die Option sollte mit Bedacht verwendet werden.

Eine aktualisierende Wiederherstellung von Datenbanken könnte eine Wiederherstellung mit Bandeinheiten erfordern. Die API zur aktualisierenden Wiederherstellung gibt eine Warnung zurück, wenn Benutzereingriff auf einer Einheit erforderlich ist. Die API kann mit einer der folgenden drei Aktionen erneut aufgerufen werden:

DB2ROLLFORWARD_LOADREC_CONT
Verwendung der Einheit fortsetzen, die die Warnung generiert hat (zum Beispiel, wenn ein neues Band eingelegt wurde).

DB2ROLLFORWARD_DEVICE_TERM
Verwendung der Einheit beenden, die die Warnung generiert hat (zum Beispiel, wenn keine Bänder mehr vorhanden sind).

DB2ROLLFORWARD_LOAD_REC_TERM
Alle von der Wiederherstellung verwendeten Einheiten beenden.

piStopTime
Eingabe. Eine Zeichenfolge, die eine Zeitmarke in ISO-Format enthält. Die Datenbankwiederherstellung wird gestoppt, wenn diese Zeitmarke überschritten wird. Geben Sie SQLUM_INFINITY_TIMESTAMP an, um so weit wie möglich aktualisierend wiederherzustellen. Der Parameter kann für `DB2ROLLFORWARD_QUERY`, `DB2ROLLFORWARD_PARM_CHECK` und alle Wiederherstellungsaktionen (`DB2ROLLFORWARD_LOADREC_`*xxx*) des aufrufenden Programms NULL sein.

piUserName
Eingabe. Eine Zeichenfolge, die den Benutzernamen der Anwendung enthält. Kann NULL sein.

piPassword
Eingabe. Eine Zeichenfolge, die das Kennwort für den angegebenen Benutzernamen (wenn vorhanden) enthält. Kann NULL sein.

piOverflowLogPath
Eingabe. Dieser Parameter wird verwendet, um einen alternativ zu verwendenden Protokollpfad anzugeben. Zusätzlich zu den aktiven Protokolldateien müssen Archivprotokolldateien (durch den Benutzer) in den durch *logpath* angegebenen Protokollpfad versetzt werden, bevor sie von diesem Dienstprogramm verwendet werden können. Das könnte ein Problem sein, wenn der Benutzer nicht genügend Speicherbereich im durch *logpath* angegebenen Protokollpfad hat. Aus diesem Grund wird der Überlaufprotokollpfad angegeben. Während der aktualisierenden Wiederherstellung werden die erforderlichen Protokolldateien zuerst im durch *logpath* angegebenen Protokollpfad und danach im Überlaufprotokollpfad gesucht. Die erforderlichen Protokolldateien für die aktualisierende Wiederherstellung von Tabellenbereichen können entweder im durch *logpath* angegebenen Protokollpfad oder in den Überlaufprotokollpfad gestellt werden. Wenn das aufrufende Programm keinen Überlaufprotokollpfad angibt, ist der Standardwert der durch *logpath* angegebene Pfad. In einer Umgebung mit partitionierten Datenbanken muss der Überlaufprotokollpfad ein gültiger, vollständig qualifizierter Pfad sein. Der Standardpfad ist der Standardüberlaufprotokollpfad für jeden Knoten. In einer Umgebung mit einer einzelnen Datenbankpartition kann der Überlaufprotokollpfad relativ sein, wenn der Server lokal ist.

iNumChngLgOvrflw
Eingabe. Nur Umgebungen mit partitionierten Datenbanken. Die Anzahl geänderter Überlaufprotokollpfade. Diese neuen Protokollpfade überschreiben nur den Standardüberlaufprotokollpfad für den angegebenen Datenbankpartitionsserver.

piChngLogOvrflw
Eingabe. Nur Umgebungen mit partitionierten Datenbanken. Ein Zeiger auf eine Struktur, die die vollständig qualifizierten Namen der geänderten Überlaufprotokollpfade enthält. Diese neuen Protokollpfade überschreiben nur den Standardüberlaufprotokollpfad für den angegebenen Datenbankpartitionsserver.

iConnectMode
Eingabe. Gültige Werte (definiert in `db2ApiDf.h`) sind:

DB2ROLLFORWARD_OFFLINE
Aktualisierende Offlinewiederherstellung. Dieser Wert muss für eine aktualisierende Datenbankwiederherstellung angegeben werden.

DB2ROLLFORWARD_ONLINE
Aktualisierende Onlinewiederherstellung.

piTablespaceList
Eingabe. Ein Zeiger auf eine Struktur, die die Namen der bis zum Protokollende oder einem bestimmten Zeitpunkt aktualisierend wiederherzustellenden Tabellenbereiche enthält. Wenn keine Angabe gemacht wird, werden die Tabellenbereiche ausgewählt, für die eine aktualisierende Wiederherstellung erforderlich ist.

iAllNodeFlag
Eingabe. Nur Umgebungen mit partitionierten Datenbanken. Gibt an, ob die aktualisierende Wiederherstellungsoperation auf alle in `db2nodes.cfg` definierten Datenbankpartitionsserver angewendet werden soll. Gültige Werte sind:

DB2_NODE_LIST
Auf Datenbankpartitionsserver in einer Liste anwenden, die in *piNodeList* übergeben wird.

DB2_ALL_NODES
Auf alle Datenbankpartitionsserver anwenden. *piNodeList* sollte NULL sein. Dies ist der Standardwert.

DB2_ALL_EXCEPT
Auf alle Datenbankpartitionsserver anwenden außer auf die in einer Liste aufgeführten Datenbankpartitionsserver, die in *piNodeList* übergeben wird.

DB2_CAT_NODE_ONLY
Nur auf die Katalogtabellenpartition anwenden. *piNodeList* sollte NULL sein.

iNumNodes
Eingabe. Gibt die Anzahl Datenbankpartitionsserver im Bereich *piNodeList* an.

piNodeList
Eingabe. Ein Zeiger auf einen Bereich mit Nummern von Datenbankpartitionsservern, auf denen die aktualisierende Wiederherstellung ausgeführt werden soll.

iNumNodeInfo
Eingabe. Definiert die Größe des Ausgabeparameters *poNodeInfo*, der groß genug sein muss, um Statusinformationen für jede aktualisierend wiederherzustellende Datenbankpartition zu enthalten. In einer Umgebung mit einer einzelnen Datenbankpartition sollte dieser Parameter auf 1 gesetzt werden. Der Wert dieses Parameters sollte gleich der Anzahl Datenbankpartitionsserver sein, für die diese API aufgerufen wird.

piDroppedTblID
Eingabe. Eine Zeichenfolge, die die ID der gelöschten Tabelle enthält, deren Wiederherstellung versucht wird.

piExportDir
Eingabe. Das Verzeichnis, in das die gelöschten Tabellendaten exportiert werden.

RollforwardFlags
Eingabe. Gibt die Markierungen für aktualisierende Wiederherstellung an. Gültige Werte (definiert in `db2ApiDf.h`):

DB2ROLLFORWARD_EMPTY_FLAG
Keine Markierungen angegeben.

DB2ROLLFORWARD_LOCAL_TIME
Ermöglicht dem Benutzer die aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt in Ortszeit anstatt in WEZ. Dies erleichtert Benutzern die aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt auf ihren lokalen Maschinen und vermeidet potenzielle Benutzerfehler aufgrund der Umsetzung von Ortszeit in WEZ.

DB2ROLLFORWARD_NO_RETRIEVE
Steuert, welche Protokolldateien auf der Bereitschaftsmaschine aktualisierend wiederhergestellt werden, indem der Benutzer das Abrufen archivierter Protokolle inaktivieren kann. Durch diese Steuerung der aktualisierend wiederherzustellenden Protokolle kann sichergestellt werden, dass sich die Bereitschaftsmaschine zeitlich `X` Stunden hinter der Produktionsmaschine befindet, so dass sich Benutzeraktionen nicht auf beide Systeme auswirken. Diese Option ist nützlich, wenn das fehlertolerante System (d h. das Bereitschaftssystem) keinen Zugriff auf das Archiv hat. Wenn beispielsweise TSM das Archiv ist, erlaubt es nur der Ursprungsmaschine, die Dateien abzurufen. Diese Option schließt ebenfalls aus, dass das fehlertolerante System eine unvollständige Protokolldatei abruft, während das Produktionssystem eine Datei archiviert und das fehlertolerante System dieselbe Datei abruft.

poApplicationId
Ausgabe. Die Anwendungs-ID.

poNumReplies
Ausgabe. Die Anzahl erhaltener Antworten.

poNodeInfo
Ausgabe. Antwortinformationen der Datenbankpartition.

nodenum
Knotennummer.

pathlen
Die Länge des neuen Protokollpfads.

logpath
Der neue Überlaufprotokollpfad.

num_entry
Anzahl Einträge in der Liste, auf die im Feld *tablespace* gezeigt wird.

tablespace
Zeigt auf die Struktur *sqlu_tablepsace_entry*.

reserve_len
Länge der im Feld *tablespace_entry* angegebenen Zeichenfolge. Für Sprachen außer C.

tablespace_entry
Name des Tabellenbereichs.

state Statusinformation.

nextarclog
Ein Puffer, der den zurückgegebenen Namen der nächsten erforderlichen Archivprotokolldatei enthält. Wenn eine andere Aktion des aufrufenden Programms als `DB2ROLLFORWARD_QUERY` angegeben wird, weist der in diesem Feld zurückgegebene Wert darauf hin, dass beim Zugriff auf die Datei ein Fehler aufgetreten ist. Mögliche Ursachen:

- Die Datei wurde weder im Datenbankprotokollverzeichnis noch in dem Pfad gefunden, der vom Parameter für den Überlaufprotokollpfad angegeben ist.
- Die Protokollarchivierungsmethode konnte die archivierte Datei nicht zurückgeben.

firstarcdel
Ein Puffer, der den zurückgegebenen Namenn der ersten Archivprotokolldatei enthält, die nicht mehr zur Wiederherstellung benötigt wird. Diese Datei und alle Dateien bis einschließlich lastarcdel können versetzt werden, um Plattenspeicher freizugeben.

Wenn beispielsweise die in firstarcdel und lastarcdel zurückgegebenen Werte S0000001.LOG und S0000005.LOG lauten, können die folgenden Dateien versetzt werden:

- `S0000001.LOG`
- `S0000002.LOG`
- `S0000003.LOG`
- `S0000004.LOG`
- `S0000005.LOG`

lastarcdel
Ein Puffer, der den zurückgegebenen Namen der letzten Archivprotokolldatei enthält, die aus dem Datenbankprotokollverzeichnis entfernt werden kann.

lastcommit
Eine Zeichenfolge, die eine Zeitmarke in ISO-Format enthält. Dieser Wert ist die Zeitmarke der letzten Transaktion, die festgeschrieben wurde, nachdem die aktualisierende Wiederherstellung endete.

Hinweise:

Der Datenbankmanager verwendet die in den archivierten und den aktiven Protokolldateien enthaltenen Informationen zur Rekonstruktion der Transaktionen, die seit der letzten Sicherung einer Datenbank für die Datenbank ausgeführt wurden.

Die Aktionen, die bei Aufruf dieser API ausgeführt werden, hängen von der Markierung *rollforward_pending* der Datenbank vor dem Aufruf ab. Diese Markierung kann mit Hilfe der API "db2CfgGet - Konfigurationsparameter abrufen" abgerufen werden. Die Markierung *rollforward_pending* ist auf `DATABASE` gesetzt, wenn sich die Datenbank im Status *Aktualisierende Wiederherstellung anstehend* befindet. Sie ist auf `TABLESPACE` gesetzt, wenn mindestens ein Tabellenbereich den Status `SQLB_ROLLFORWARD_PENDING` (Aktualisierende Wiederherstellung anstehend) oder `SQLB_ROLLFORWARD_IN_PROGRESS` (Aktualisierende Wiederherstellung läuft) hat. Die Markierung *rollforward_pending* ist auf `NO` gesetzt, wenn weder die Datenbank noch einer der Tabellenbereiche aktualisierend wiederhergestellt werden muss.

Wenn sich die Datenbank beim Aufruf der API im Status *Aktualisierende Wiederherstellung anstehend* befindet, wird sie aktualisierend wiederhergestellt. Tabellenbereiche erhalten nach einer erfolgreichen aktualisierenden Wiederherstellung der Datenbank wieder den normalen Status, außer es werden Tabellenbereiche aufgrund eines abnormalen Status in den Offlinestatus versetzt. Wenn die Markierung *rollforward_pending* auf `TABLESPACE` gesetzt ist, werden nur die Tabellenbereiche aktualisierend wiederhergestellt, die sich im Status *Aktualisierende Wiederherstellung anstehend* befinden oder die mit Namen angefordert werden.

Anmerkung: Wenn die aktualisierende Wiederherstellung der Tabellenbereiche abnormal beendet wird, werden Tabellenbereiche, deren aktualisierende Wiederherstellung gerade lief, in den Status `SQLB_ROLLFORWARD_IN_PROGRESS` (Aktualisierende Wiederherstellung läuft) versetzt. Beim nächsten Aufruf von ROLLFORWARD DATABASE werden nur die Tabellenbereiche verarbeitet, die den Status `SQLB_ROLLFORWARD_IN_PROGRESS` (Aktualisierende Wiederherstellung läuft) haben. Wenn die Gruppe der ausgewählten Tabellenbereichsnamen nicht alle Tabellenbereich mit dem Status `SQLB_ROLLFORWARD_IN_PROGRESS` (Aktualisierende Wiederherstellung läuft) enthält, werden die nicht erforderlichen Tabellenbereiche in den Status `SQLB_RESTORE_PENDING` (Wiederherstellung anstehend) versetzt.

Wenn sich die Datenbank nicht im Status *Aktualisierende Wiederherstellung anstehend* befindet und kein Zeitpunkt angegeben wird, werden alle Tabellenbereiche, die den Status *Aktualisierende Wiederherstellung läuft* haben, bis zum Ende der Protokolle aktualisierend wiederhergestellt. Wenn keine Tabellenbereiche den Status *Aktualisierende Wiederherstellung läuft* haben, werden alle Tabellenbereiche, die sich im Status *Aktualisierende Wiederherstellung anstehend* befinden, bis zum Ende der Protokolle aktualisierend wiederhergestellt.

Diese API beginnt beim Lesen der Protokolldateien mit der Protokolldatei, die mit dem Sicherungsimage übereinstimmt. Der Name dieser Protokolldatei kann ermittelt werden, indem vor der aktualisierenden Wiederherstellung von Protokolldateien diese API mit einer auszuführenden Aktion des Typs `DB2ROLLFORWARD_QUERY` aufgerufen wird.

Die in den Protokolldateien aufgezeichneten Transaktionen werden erneut auf die Datenbank angewendet. Das Protokoll wird soweit zeitlich vorwärts verarbeitet, wie Informationen verfügbar sind, bzw. bis zu dem vom Stoppzeitparameter angegebenen Zeitpunkt.

Die Wiederherstellung wird gestoppt, wenn eines der folgenden Ereignisse auftritt:

- Es werden keine weiteren Protokolldateien gefunden.
- Eine in der Protokolldatei vorhandene Zeitmarke liegt zeitlich nach der vom Stoppzeitparameter angegebenen Zeitmarke für die Beendigung.
- Während des Lesens der Protokolldatei tritt ein Fehler auf.

Einige Transaktionen können möglicherweise nicht wiederhergestellt werden. Der in *lascommit* zurückgegebene Wert gibt die Zeitmarke der letzten festgeschriebenen Transaktion an, die für die Datenbank ausgeführt wurde.

Wenn die Datenbank auf Grund eines Anwendungs- oder Benutzerfehlers wiederhergestellt werden muss, kann der Benutzer in *piStopTime* einen Zeitmarkenwert angeben, der andeutet, dass die Wiederherstellung vor dem Zeitpunkt gestoppt werden soll, zu dem der Fehler auftrat. Dies ist nur für eine aktualisierende Wiederherstellung der gesamten Datenbank möglich sowie für eine aktualisierende Wiederherstellung von Tabellenbereichen bis zu einem bestimmten Zeitpunkt. Auf gleiche Weise kann die Wiederherstellung auch vor dem Auftreten eines Protokolllesefehlers gestoppt werden, dessen Zeitpunkt bei einer früheren, fehlgeschlagenen Wiederherstellungsoperation ermittelt wurde.

Wenn die Markierung *rollforward_recovery* auf `DATABASE` gesetzt ist, steht die Datenbank erst nach Beendigung der aktualisierenden Wiederherstellung wieder zur Verfügung. Die Wiederherstellung kann durch Aufrufen der API mit der aufrufenden Aktion `DB2ROLLFORWARD_STOP` oder `DB2ROLLFORWARD_RFWRD_STOP` beendet werden, um die Datenbank aus dem Status *Aktualisierende Wiederherstellung anstehend* zu nehmen. Wenn die Markierung *rollforward_recovery* auf `TABLESPACE` gesetzt ist, steht die Datenbank zur Verfügung. Die Tabellenbereiche, die den Status `SQLB_ROLLFORWARD_PENDING` (Aktualisierende Wiederherstellung anstehend) bzw. `SQLB_ROLLFORWARD_IN_PROGRESS` (Aktualisierende Wiederherstellung läuft) haben, stehen jedoch erst dann wieder zur Verfügung, wenn die API zur Ausführung einer aktualisierenden Wiederherstellung der Tabellenbereiche aufgerufen wurde. Wenn Tabellenbereiche bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt werden, erhalten sie nach einer erfolgreichen Wiederherstellung den Status *Sicherung anstehend*.

Wenn die Option *RollforwardFlags* auf `DB2ROLLFORWARD_LOCAL_TIME` gesetzt wird, werden alle Nachrichten in Ortszeit an den Benutzer zurückgegeben. In Umgebungen mit partitionierten Datenbanken werden alle Zeitangaben auf dem Server und auf der Katalogtabellenpartition konvertiert. Die Zeichenfolge der Zeitmarke wird auf dem Server in WEZ konvertiert, so dass die Zeitmarke die Zeitzone des Servers und nicht des Clients wiedergibt. Wenn sich der Client in einer anderen Zeitzone befindet als der Server, sollte die Ortszeit des Servers verwendet werden. Dies unterscheidet sich von der Ortszeitoption der Steuerzentrale, die die Ortszeit des Clients verwendet. Wenn die Zeichenfolge der Zeitmarke aufgrund der Sommerzeit nahe am Zeitwechsel liegt, muss klar sein, ob die Stoppzeit vor oder nach dem Zeitwechsel liegt, damit sie korrekt angegeben wird.

Zugehörige Referenzen:

- „SQLCA" im Handbuch *Administrative API Reference*
- „db2Restore - Datenbank wiederherstellen" auf Seite 113

Zugehörige Beispiele:

- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.sqC -- How to recover a database (C++)"

Sitzungen zur aktualisierenden Wiederherstellung - Beispiele für CLP

Beispiel 1

Mit dem Befehl ROLLFORWARD DATABASE können Sie mehrere Operationen gleichzeitig spezifizieren, wobei diese durch das Schlüsselwort AND getrennt sind. Sie benötigen z. B. die folgenden Einzelbefehle, um eine aktualisierende Wiederherstellung 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, dass Sie überprüfen, ob die aktualisierende Wiederherstellung wie erwartet fortgeschritten ist, bevor Sie sie stoppen und möglicherweise Protokolle fehlen. Dies ist besonders wichtig, wenn während der aktualisierenden Wiederherstellung ein beschädigtes Protokoll gefunden wird und dieses Protokoll als „Protokollende" interpretiert wird. In solchen Fällen könnte eine unbeschädigte Sicherungskopie dieses Protokolls verwendet werden, um die aktualisierende Wiederherstellung mit weiteren Protokollen fortzusetzen.

Beispiel 2

Führen Sie eine aktualisierende Wiederherstellung bis zum Protokollende aus (zwei Tabellenbereiche wurden wiederhergestellt):

```
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 Tabellenbereichswiederherstellung bis zum Protokollende erforderlich. Tabellenbereichsnamen sind nicht erforderlich. Wenn keine Bereiche angegeben sind, werden alle Tabellenbereiche eingeschlossen, die eine aktualisierende Wiederherstellung erfordern. Wenn nur eine Untermenge dieser Tabellenbereiche aktualisierend wiederhergestellt werden sollen, müssen die Namen der Tabellenbereiche angegeben werden.

Beispiel 3

Nachdem drei Tabellenbereiche wiederhergestellt worden sind, stellen Sie wie folgt einen Bereich aktualisierend bis zum Ende der Protokolle wieder her und die anderen beiden Bereiche bis zu einem Zeitpunkt. Beide Aktionen sollen online ausgeführt werden.

```
db2 rollforward db sample to end of logs tablespace(TBS1) online

db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS2, TBS3) online
```

Beachten Sie, dass zwei aktualisierende Wiederherstellungsoperationen nicht gleichzeitig ausgeführt werden können. Sie können den zweiten Befehl erst aufrufen, nachdem die erste aktualisierende Wiederherstellungsoperation erfolgreich beendet wurde.

Beispiel 4

Nachdem die Datenbank wiederhergestellt wurde, führen Sie wie folgt eine aktualisierende Wiederherstellung bis zu einem Zeitpunkt aus, wobei Sie OVERFLOW LOG PATH zur Angabe des Verzeichnisses verwenden, in dem der Benutzerexit Archivprotokolldateien speichert:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   overflow log path (/logs)
```

Beispiel 5 (Umgebungen mit partitionierten Datenbanken)

Es gibt folgende drei Knoten: 0, 1 und 2. Der Tabellenbereich TBS1 ist auf allen Knoten definiert, der Tabellenbereich TBS2 nur auf den Knoten 0 und 2. Nachdem Sie die Datenbank auf Knoten 1 und TBS1 auf den Knoten 0 und 2 wiederhergestellt haben, stellen Sie die Datenbank auf Knoten 1 wie folgt aktualisierend wieder her:

```
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 auf den Knoten 0 und 2 aktualisierend wiederhergestellt. In diesem Fall ist die Klausel TABLESPACE(TBS1) optional.

Beispiel 6 (Umgebungen mit partitionierten Datenbanken)

Nachdem Sie Tabellenbereich TBS1 nur auf den Knoten 0 und 2 wiederhergestellt haben, stellen Sie TBS1 wie folgt auf den Knoten 0 und 2 aktualisierend wieder her:

```
db2 rollforward db sample to end of logs
```

Knoten 1 wird ignoriert.

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

Dieser Befehl schlägt fehl, weil TBS1 auf Knoten 1 nicht für eine aktualisierende Wiederherstellung bereit ist. Der Befehl gibt SQL4906N aus.

```
db2 rollforward db sample to end of logs on nodes (0, 2) tablespace(TBS1)
```

Dieser Befehl wird erfolgreich beendet.

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

Dieser Befehl schlägt fehl, weil TBS1 auf Knoten 1 nicht für eine aktualisierende Wiederherstellung bereit ist. Alle Teile müssen gemeinsam aktualisierend wiederhergestellt werden.

Anmerkung: Bei der aktualisierenden Tabellenbereichswiederherstellung bis zu einem Zeitpunkt wird die Knotenklausel nicht akzeptiert. Die aktualisierende Wiederherstellungsoperation muss auf allen Knoten ausgeführt werden, auf denen sich der Tabellenbereich befindet.

Geben Sie nach der Wiederherstellung von TBS1 auf Knoten 1 Folgendes ein:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

Dieser Befehl wird erfolgreich beendet.

Beispiel 7 (Umgebungen mit partitionierten Datenbanken)

Nachdem Sie einen Tabellenbereich auf allen Knoten wiederhergestellt haben, stellen Sie ihn wie folgt aktualisierend bis zum Zeitpunkt PIT2 wieder her, aber geben Sie nicht `AND STOP` an. Die aktualisierende Wiederherstellungsoperation läuft immer noch. Brechen Sie sie wie folgt ab, und führen Sie eine aktualisierende Wiederherstellung bis zum Zeitpunkt PIT1 aus:

```
db2 rollforward db sample to pit2 tablespace(TBS1)
db2 rollforward db sample cancel tablespace(TBS1)

** TBS1 auf allen Knoten wiederherstellen **

db2 rollforward db sample to pit1 tablespace(TBS1)
db2 rollforward db sample stop tablespace(TBS1)
```

Beispiel 8 (Umgebungen mit partitionierten Datenbanken)

Stellen Sie wie folgt einen Tabellenbereich wieder her, der sich auf den in der Datei `db2nodes.cfg` aufgelisteten acht Knoten (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 Knoten, auf denen sich der Tabellenbereich befindet, müssen nicht angegeben werden. Das Dienstprogramm verwendet standardmäßig die Datei `db2nodes.cfg`.

Beispiel 9 (Umgebung mit partitionierten Datenbanken)

Stellen Sie wie folgt sechs kleine Tabellenbereiche aktualisierend wieder her, die sich auf einem einzelnen Knoten (auf Knoten 6) einer Datenbankpartitionsgruppe befinden:

```
db2 rollforward database dwtest to end of logs on node (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

Diese Operation bis zum Protokollende (nicht bis zu einem Zeitpunkt) wird erfolgreich beendet.

Kapitel 5. Datenbankwiederherstellung

In diesem Abschnitt wird das DB2 UDB-Wiederherstellungsdienstprogramm beschrieben, das auf Basis der in der Datei des Wiederherstellungsprotokolls vorhandenen Informationen die erforderlichen Operationen zum Wiederherstellen und aktualisierenden Wiederherstellen einer Datenbank bis zu einem bestimmten Zeitpunkt ausführt.

Die folgenden Themen werden behandelt:

Wiederherstellung - Übersicht

Das Wiederherstellungsdienstprogramm führt auf Basis der in der Datei des Wiederherstellungsprotokolls 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 geeignetste Sicherungsimage aus und führt die Wiederherstellungsoperationen aus.

Das Wiederherstellungsdienstprogramm bietet keine Unterstützung für die folgenden Optionen des Befehls RESTORE DATABASE:

- TABLESPACE tabellenbereichsname. Wiederherstellungsoperationen für Tabellenbereiche werden nicht unterstützt.
- INCREMENTAL. Teilwiederherstellungsoperationen 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 Wiederherstellungsoperation 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 Wiederherstellung anstehend* versetzt werden soll, nachdem sie erfolgreich wiederhergestellt worden ist.
- WITHOUT DATALINK. Es kann nicht angegeben werden, dass kein Datenabgleich der verknüpften Dateien ausgeführt werden soll.
- PARALLELISM n. Es kann kein Grad der Parallelität für die Wiederherstellungsoperation angegeben werden.

- WITHOUT PROMPTING. Es kann nicht angegeben werden, dass die Wiederherstellungsoperation nicht überwacht ausgeführt werden soll.

Zugehörige Konzepte:

- „Konvertieren von Client/Server-Zeitmarken" auf Seite 143

Zugehörige Tasks:

- „Verwenden des Wiederherstellungsdienstprogramms" auf Seite 170

Zugehörige Referenzen:

- „RESTORE DATABASE" auf Seite 102
- „Konfigurationsparameter für die Datenbankprotokollierung" auf Seite 42

Erforderliche Zugriffsrechte und Berechtigungen zur Verwendung des Wiederherstellungsdienstprogramms

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf sie zuzugreifen. Berechtigungsstufen stellen eine Methode dar, Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zu gruppieren. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte. Benutzer können nur auf solche Objekte zugreifen, für die sie die entsprechende Berechtigung besitzen, d. h., für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Sie benötigen die Berechtigung SYSADM, SYSCTRL oder SYSMAINT, um das Wiederherstellungsdienstprogramm verwenden zu können.

Verwenden des Wiederherstellungsdienstprogramms

Vorbedingungen:

Es sollte noch keine Verbindung zu der wiederherzustellenden Datenbank bestehen: Das Datenbankwiederherstellungsdienstprogramm stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der Wiederherstellungsoperation beendet wird.

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln.

Prozedur:

Das Wiederherstellungsdienstprogramm kann über den Befehlszeilenprozessor (CLP) oder die Anwendungsprogrammierschnittstelle **db2Recover** aufgerufen werden.

Das folgende Beispiel zeigt, wie Sie den Befehl RECOVER DATABASE über den CLP verwenden:

```
db2 recover db sample
```

Anmerkung: In einer Umgebung mit partitionierten Datenbanken muss das Wiederherstellungsdienstprogramm vom Katalogknoten der Datenbank aus aufgerufen werden.

Zugehörige Konzepte:

- „Wiederherstellung - Übersicht" auf Seite 169

Zugehörige Referenzen:

- „db2Recover - Datenbank wiederherstellen" auf Seite 177
- „RECOVER DATABASE" auf Seite 171

Konvertieren von Client/Server-Zeitmarken

In diesem Abschnitt wird die Generierung von Zeitmarken in einer Client/Server-Umgebung erläutert:

- Wenn Sie für eine aktualisierende Wiederherstellung eine Ortszeit angeben, werden alle Nachrichten ebenfalls in Ortszeit zurückgegeben.

  Anmerkung: Alle Zeitangaben werden auf dem Server und (in Umgebungen mit partitionierten Datenbanken) auf dem Katalogknoten konvertiert.
- Die Zeichenfolge der Zeitmarke wird auf dem Server in GMT konvertiert, so dass die Zeitmarke die Zeitzone des Servers und nicht 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 aufgrund der Sommerzeit nahe am Zeitwechsel liegt, muss klar sein, ob die Stoppzeit vor oder nach dem Zeitwechsel liegt, damit sie korrekt angegeben wird.

Zugehörige Konzepte:

- „Aktualisierende Wiederherstellung - Übersicht" auf Seite 129
- „Synchronisieren der Systemuhren in einem System mit partitionierten Datenbanken" auf Seite 142

RECOVER DATABASE

Stellt eine Datenbank bis zu einem bestimmten Zeitpunkt oder bis zum Ende der Protokolle aktualisierend wiederher.

Geltungsbereich:

In einer Umgebung mit partitionierten Datenbanken kann dieser Befehl nur von der Katalogtabellenpartition aus aufgerufen werden. Eine Wiederherstellungsoperation für Datenbanken bis zu einem bestimmten Zeitpunkt wirkt sich auf alle Partitionen aus, die in der Datei `db2nodes.cfg` aufgelistet sind. Eine Wiederherstellungsoperation für Datenbanken bis zum Ende der Protokolle wirkt sich auf die Partitionen aus, die angegeben werden. Wenn keine Partitionen angegeben werden, wirkt sich die Operation auf alle Partitionen aus, die in der Datei `db2nodes.cfg` aufgelistet sind.

Berechtigung:

Zur Wiederherstellung einer vorhandenen Datenbank eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

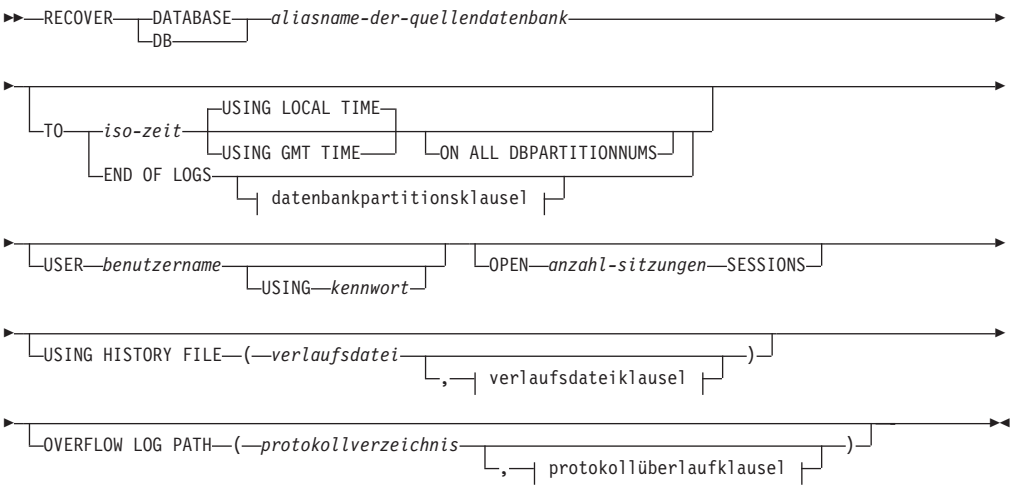
Zur Wiederherstellung in eine neue Datenbank eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*

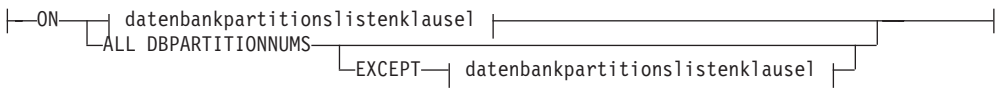
Erforderliche Verbindung:

Zur Wiederherstellung einer vorhandenen Datenbank ist eine Datenbankverbindung erforderlich. Dieser Befehl stellt automatisch eine Verbindung zur angegebenen Datenbank her und beendet diese Verbindung, sobald die Wiederherstellungsoperation abgeschlossen ist. Zur Wiederherstellung in eine neue Datenbank sind eine Exemplar- und eine Datenbankverbindung erforderlich. Die Verbindung zu einem Exemplar ist erforderlich, um die Datenbank zu erstellen.

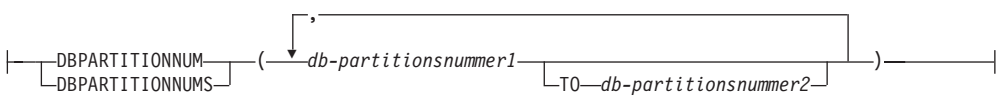
Befehlssyntax:



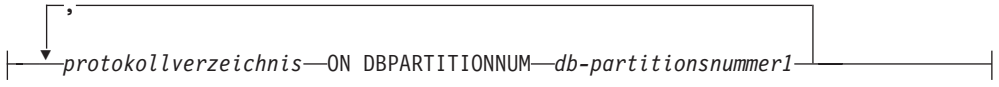
Datenbankpartitionsklausel:



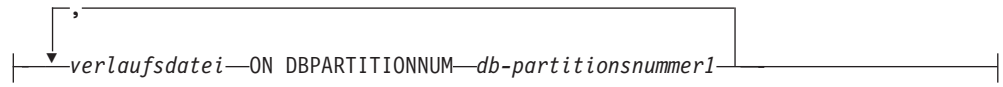
Datenbankpartitionslistenklausel:



Protokollüberlaufklausel:



verlaufsdateiklausel:



Befehlsparameter:

DATABASE *aliasname-der-quellendatenbank*
Der Aliasname der Datenbank, die wiederhergestellt werden soll.

USER *benutzername*
Der Benutzername, unter dem die Datenbank wiederhergestellt werden soll.

USING *kennwort*
Gibt das Kennwort an, mit dem der Benutzername authentifiziert wird. Wenn kein Kennwort angegeben wird, wird der Benutzer zur Eingabe des Kennworts aufgefordert.

TO

iso-zeit Der Zeitpunkt, bis zu dem alle festgeschriebenen Transaktionen wiederhergestellt werden sollen (einschließlich der Transaktion, die genau zu diesem Zeitpunkt festgeschrieben wurde, und aller vorher festgeschriebenen Transaktionen).

Dieser Wert wird als Zeitmarke angegeben, eine 7-teilige Zeichenfolge, die eine Kombination aus Datum und Zeit angibt. Das Format ist *jjjj-mm-tt-hh.mm.ss.nnnnnn* (Jahr, Monat, Tag, Stunde, Minuten, Sekunden, Mikrosekunden) und wird in Weltzeit (Coordinated Universal Time, UTC) ausgedrückt. Die Weltzeit hilft zu verhindern, dass verschiedenen Protokollen dieselbe Zeitmarke zugeordnet wird (z. B. wegen einer Zeitänderung aufgrund der Sommerzeit). Die Zeitmarke in einem Sicherungsimage basiert auf der Ortszeit, zu der die Sicherungsoperation gestartet wurde. Das Sonderregister CURRENT TIMEZONE gibt den Unterschied zwischen UTC und der Ortszeit auf dem Anwendungsserver an. Der Unterschied wird als Zeitdifferenz dargestellt (eine Dezimalzahl, in der die ersten beiden Ziffern die Anzahl Stunden, die nächsten beiden Ziffern die Anzahl Minuten und die letzten beiden Ziffern die Anzahl Sekunden darstellen). Indem CURRENT TIMEZONE von einer Ortszeit abgezogen wird, wird diese Ortszeit in Weltzeit konvertiert.

USING LOCAL TIME
Gibt den Zeitpunkt an, bis zu dem wiederhergestellt werden soll. Diese Option ermöglicht dem Benutzer die Wiederherstellung bis zu einem bestimmten Zeitpunkt in Ortszeit anstatt in WEZ. Dies erleichtert Benutzern die Wiederherstellung bis zu einem bestimmten Zeitpunkt auf ihren lokalen Maschinen und vermeidet potenzielle Benutzerfehler aufgrund der Umsetzung von Ortszeit in WEZ.

Anmerkungen:

1. Wenn der Benutzer für eine Wiederherstellung eine Ortszeit angibt, werden alle Nachrichten ebenfalls in Ortszeit an den Benutzer zurückgegeben. Alle Zeitangaben werden auf dem Server und (in Umgebungen mit partitionierten Datenbanken) auf der Katalogdatenbankpartition konvertiert.

2. Die Zeichenfolge der Zeitmarke wird auf dem Server in WEZ konvertiert, so dass die Zeitmarke die Zeitzone des Servers und nicht des Clients wiedergibt. Wenn sich der Client in einer anderen Zeitzone befindet als der Server, sollte die Ortszeit des Servers verwendet werden. Dies unterscheidet sich von der Ortszeitoption der Steuerzentrale, die die Ortszeit des Clients verwendet.
3. Wenn die Zeichenfolge der Zeitmarke aufgrund der Sommerzeit nahe am Zeitwechsel liegt, muss klar sein, ob die Stoppzeit vor oder nach dem Zeitwechsel liegt, damit sie korrekt angegeben wird.

USING GMT TIME
Gibt den Zeitpunkt an, bis zu dem wiederhergestellt werden soll.

END OF LOGS
Gibt an, dass alle festgeschriebenen Transaktionen aus allen Onlinearchivprotokolldateien angewendet werden sollen, die in dem durch den Datenbankkonfigurationsparameter *logpath* angegebenen Pfad enthalten sind.

ON ALL DBPARTITIONNUMS
Gibt an, dass Transaktionen auf allen in der Datei `db2nodes.cfg` angegebenen Partitionen aktualisierend wiederhergestellt werden sollen. Dies ist die Standardeinstellung, falls keine Datenbankpartitionsklausel angegeben ist.

EXCEPT
Gibt an, dass Transaktionen auf allen in der Datei `db2nodes.cfg` angegebenen Partitionen aktualisierend wiederhergestellt werden sollen, mit Ausnahme der Partitionen in der Datenbankpartitionsliste.

ON DBPARTITIONNUM / ON DBPARTITIONNUMS
Stellt die Datenbank in einer Datenbankpartitionsgruppe aktualisierend wieder her.

db-partitionsnummer1
Gibt eine Datenbankpartitionsnummer in der Datenbankpartitionsliste an.

db-partitionsnummer2
Gibt die zweite Datenbankpartitionsnummer an, so dass alle Partitionen von *db-partitionsnummer1* bis einschließlich *db-partitionsnummer2* in die Datenbankpartitionsliste aufgenommen werden.

OPEN *anzahl-sitzungen* **SESSIONS**
Gibt die Anzahl E/A-Sitzungen an, die mit Tivoli Storage Manager (TSM) oder einem Programm eines anderen Lieferanten verwendet werden sollen.

USING HISTORY FILE *verlaufsdatei*

verlaufsdatei **ON DBPARTITIONNUM**
Ermöglicht in einer Umgebung mit partitionierten Datenbanken das Verwenden einer anderen Verlaufsdatei.

OVERFLOW LOG PATH *protokollverzeichnis*
Gibt einen alternativen Protokollpfad an, der während der Wiederherstellung nach archivierten Protokollen durchsucht wird. Verwenden Sie diesen Parameter, wenn Protokolldateien an eine andere als die im Datenbankkonfigurationsparameter *logpath* angegebene Position versetzt wurden. In einer Umgebung mit partitionierten Datenbanken ist dies der (vollständig quali-

fizierte) Standardüberlaufprotokollpfad *für alle Partitionen*. Für Datenbanken mit einer Partition kann ein relativer Überlaufprotokollpfad angegeben werden.

Anmerkung: Der Befehlsparameter OVERFLOW LOG PATH überschreibt den Wert (falls vorhanden) des Datenbankkonfigurationsparameters *overflowlogpath*.

protokollverzeichnis **ON DBPARTITIONNUM**
In einer Umgebung mit partitionierten Datenbanken kann ein anderer Protokollpfad den Standardüberlaufprotokollpfad für eine bestimmte Partition überschreiben.

Beispiele:

In einer Umgebung mit einer einzelnen Datenbankpartition, in der die wiederherzustellende Datenbank zurzeit vorhanden ist und daher die aktuellste Version der Verlaufsdatei in *dftdbpath* zur Verfügung steht:

1. Setzen Sie den folgenden Befehl ab, um das neueste Sicherungsimage zu verwenden und die Datenbank bis zum Ende der Protokolle unter Verwendung aller Standardwerte aktualisierend wiederherzustellen:
   ```
   RECOVER DB SAMPLE
   ```
2. Setzen Sie den folgenden Befehl ab, um die Datenbank bis zu einem bestimmten Zeitpunkt wiederherzustellen. Es wird das aktuellste verfügbare Image wiederhergestellt, und die Protokolle werden angewendet, bis der angegebene Zeitpunkt erreicht ist.
   ```
   RECOVER DB SAMPLE TO  2001-12-31-04:00:00
   ```
3. Setzen Sie den folgenden Befehl ab, um die Datenbank mit einer gespeicherten Version der Protokolldatei wiederherzustellen. Wenn der Benutzer beispielsweise bis zu einem sehr weit zurückliegenden Zeitpunkt wiederherstellen muss, der nicht mehr in der aktuellen Verlaufsdatei enthalten ist, muss der Benutzer eine Version der Verlaufsdatei aus diesem Zeitraum zur Verfügung stellen. Wenn der Benutzer eine Verlaufsdatei aus diesem Zeitraum gespeichert hat, kann die Wiederherstellung anhand dieser Version ausgeführt werden.
   ```
   RECOVER DB SAMPLE TO  1999-12-31-04:00:00
      USING HISTORY FILE (/home/user/old1999files/db2rhist.asc)
   ```

In einer Umgebung mit einer einzelnen Datenbankpartition, in der die wiederherzustellende Datenbank nicht vorhanden ist, müssen Sie die Klausel USING HISTORY FILE verwenden, um auf eine Verlaufsdatei zu zeigen.

1. Wenn Sie keine Sicherungen der Verlaufsdatei erstellt haben, sodass die im Sicherungsimage vorhandene Kopie die einzige verfügbare Version ist, wird empfohlen, den Befehl RESTORE und anschließend den Befehl ROLLFORWARD abzusetzen. Damit Sie RECOVER verwenden können, müssen Sie jedoch zuerst die Verlaufsdatei aus dem Image an eine beliebige Position extrahieren, beispielsweise `/home/user/oldfiles/db2rhist.asc`, bevor Sie diesen Befehl absetzen. (Diese Version der Verlaufsdatei enthält keine Informationen zu Verlaufsdateien, die für die aktualisierende Wiederherstellung erforderlich sind, sodass diese Verlaufsdatei nicht für eine Wiederherstellung verwendet werden kann.)
   ```
   RECOVER DB SAMPLE TO END OF LOGS
      USING HISTORY FILE (/home/user/fromimage/db2rhist.asc)
   ```

2. Wenn Sie regelmäßig oder häufig Sicherungskopien der Verlaufsdatei erstellen, sollte mit der Klausel USING HISTORY auf diese Version der Verlaufsdatei gezeigt werden. Für die Datei /home/user/myfiles/db2rhist.asc beispielsweise setzen Sie den folgenden Befehl ab:

   ```
   RECOVER DB SAMPLE TO PIT
      USING HISTORY FILE (/home/user/myfiles/db2rhist.asc)
   ```

   (In diesem Fall können Sie eine beliebige Kopie der Verlaufsdatei verwenden. Es muss nicht unbedingt die neueste Version sein, so lange die Datei eine Sicherung enthält, die vor dem angeforderten Zeitpunkt erstellt wurde.)

In einer Umgebung mit partitionierten Datenbanken, in der die Datenbank auf allen Datenbankpartitionen vorhanden ist und die neueste Verlaufsdatei auf allen Datenbankpartitionen im *dftdbpath*-Pfad zur Verfügung steht:

1. Setzen Sie den folgenden Befehl ab, um die Datenbank auf allen Knoten bis zu einem bestimmten Zeitpunkt wiederherzustellen. DB2 überprüft, ob der Zeitpunkt auf allen Knoten erreicht werden kann, bevor mit dem Ausführen von Wiederherstellungsoperationen begonnen wird.

   ```
   RECOVER DB SAMPLE TO  2001:12:31:04:00:00
   ```

2. Setzen Sie den folgenden Befehl ab, um die Datenbank auf allen Knoten bis zu diesem Zeitpunkt wiederherzustellen. DB2 überprüft, ob der Zeitpunkt auf allen Knoten erreicht werden kann, bevor mit dem Ausführen von Wiederherstellungsoperationen begonnen wird. Die Operation RECOVER auf allen Knoten ist identisch mit RECOVER in einer Umgebung mit einer einzelnen Datenbankpartition.

   ```
   RECOVER DB SAMPLE TO END OF LOGS
   ```

3. Auch wenn sich die neueste Version der Verlaufsdatei im *dftdbpath*-Pfad befindet, wollen Sie möglicherweise mehrere verschiedene Verlaufsdateien verwenden. Wenn nichts Anderes angegeben, verwendet jede Partition die Verlaufsdatei, die sich lokal an der Position */home/user/oldfiles/db2rhist.asc* befindet. Eine Ausnahme bilden dabei die Knoten 2 und 4. Knoten 2 verwendet die Datei */home/user/node2files/db2rhist.asc*, und Knoten 4 verwendet */home/user/node4files/db2rhist.asc*.

   ```
   RECOVER DB SAMPLE TO  1999:12:31:04:00:00
      USING HISTORY FILE (/home/user/oldfiles/db2rhist.asc,
         /home/user/node2files/db2rhist.asc ON DBPARTITIONNUM 2,
         /home/user/node4files/db2rhist.asc ON DBPARTITIONNUM 4)
   ```

4. Es ist möglich, nur eine Untergruppe von Knoten statt alle Knoten wiederherzustellen. In diesem Fall kann jedoch keine Wiederherstellung bis zu einem bestimmten Zeitpunkt ausgeführt werden, sondern die Wiederherstellung muss bis zum Ende der Protokolle ausgeführt werden.

   ```
   RECOVER DB SAMPLE TO END OF LOGS ON DBPARTITIONNUMS(2 TO 4, 7, 9)
   ```

In einer Umgebung mit partitionierten Datenbanken, in der die Datenbank nicht vorhanden ist:

1. Wenn Sie keine Sicherungen der Verlaufsdatei erstellt haben, sodass die im Sicherungsimage vorhandene Kopie die einzige verfügbare Version ist, wird empfohlen, den Befehl RESTORE und anschließend den Befehl ROLLFORWARD abzusetzen. Damit Sie RECOVER verwenden können, müssen Sie jedoch zuerst die Verlaufsdatei aus dem Image an eine beliebige Position extrahieren, beispielsweise /home/user/oldfiles/db2rhist.asc, bevor Sie diesen Befehl absetzen. (Diese Version der Verlaufsdatei enthält keine Informationen zu Verlaufsdateien, die für die aktualisierende Wiederherstellung erforderlich sind, sodass diese Verlaufsdatei nicht für die Wiederherstellung verwendet werden kann.)

```
RECOVER DB SAMPLE TO PIT
   USING HISTORY FILE (/home/user/fromimage/db2rhist.asc)
```

2. Wenn Sie regelmäßig oder häufig Sicherungskopien der Verlaufsdatei erstellen, sollte mit der Klausel USING HISTORY auf diese Version der Verlaufsdatei gezeigt werden. Für die Datei `/home/user/myfiles/db2rhist.asc` beispielsweise setzen Sie den folgenden Befehl ab:

```
RECOVER DB SAMPLE TO END OF LOGS
   USING HISTORY FILE (/home/user/myfiles/db2rhist.asc)
```

Hinweise:

- Eine Wiederherstellung von Datenbanken könnte eine Wiederherstellung mit Bandeinheiten erfordern. Falls ein anderes Band angefordert wird, kann der Benutzer mit einer der folgenden Optionen antworten:
  - **c** Fortsetzen. Verwendung der Einheit fortsetzen, die die Warnung generiert hat (zum Beispiel, wenn ein neues Band eingelegt wurde).
  - **d** Einheit beenden. Verwendung der Einheit beenden, die die Warnung generiert hat (zum Beispiel, wenn keine Bänder mehr vorhanden sind).
  - **t** Beenden. Alle Einheiten beenden.
- Wenn während des Zurückschreibungsabschnitts der Wiederherstellungsoperation ein Fehler auftritt, können Sie den Befehl RECOVER DATABASE erneut absetzen. Wenn die Zurückschreibungsoperation erfolgreich ausgeführt wurde, aber während der aktualisierenden Wiederherstellung ein Fehler auftrat, können Sie den Befehl ROLLFORWARD DATABASE absetzen, da es nicht erforderlich (und zeitintensiv) ist, die gesamte Wiederherstellungsoperation zu wiederholen.
- Wenn in einer Umgebung mit partitionierten Datenbanken während des Zurückschreibungsabschnitts der Wiederherstellungsoperation ein Fehler auftritt, kann es sich dabei auch nur um einen Fehler auf einer einzelnen Datenbankpartition handeln. Anstatt den Befehl RECOVER DATABASE erneut abzusetzen, der die Datenbanken auf allen Datenbankpartitionen wiederherstellt, ist es effizienter, für die fehlgeschlagene Datenbankpartition den Befehl RESTORE DATABASE und anschließend den Befehl ROLLFORWARD DATABASE zu verwenden.

db2Recover - Datenbank wiederherstellen

Stellt eine Datenbank bis zu einem bestimmten Zeitpunkt oder bis zum Ende der Protokolle aktualisierend wiederher.

Geltungsbereich:

In einer Umgebung mit partitionierten Datenbanken kann diese API nur von der Katalogtabellenpartition aus aufgerufen werden. Falls keine Datenbankpartitionsserver angegeben sind, wirkt sich der Aufruf auf alle Datenbankpartitionsserver aus, die in der Datei `db2nodes.cfg` aufgelistet sind. Wenn ein bestimmter Zeitpunkt angegeben wird, wirkt sich die API auf alle Datenbankpartitionen aus.

Berechtigung:

Zur Wiederherstellung einer vorhandenen Datenbank eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Zur Wiederherstellung in eine neue Datenbank eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*

Erforderliche Verbindung:

Zur Wiederherstellung einer vorhandenen Datenbank ist eine Datenbankverbindung erforderlich. Diese API stellt automatisch eine Verbindung zur angegebenen Datenbank her und beendet diese Verbindung, sobald die Wiederherstellungsoperation abgeschlossen ist. Zur Wiederherstellung in eine neue Datenbank ist eine Exemplar- und eine Datenbankverbindung erforderlich. Die Verbindung zu einem Exemplar ist erforderlich, um die Datenbank zu erstellen.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2Recover */
/* ... */
SQL_API_RC SQL_API_FN
  db2Recover (
  db2Uint32 versionNumber,
       void * pDB2RecovStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RecoverStruct
{
  char                        *piSourceDBAlias;
  char                        *piUsername;
  char                        *piPassword;
  db2Uint32                   iRecoverCallerAction;
   db2Uint32 iOptions;
  sqlint32                    *poNumReplies;
  struct sqlurf_info          *poNodeInfo;
  char                        *piStopTime;
  char                        *piOverflowLogPath;
  db2Uint32                   iNumChngLgOvrflw;
  struct sqlurf_newlogpath    *piChngLogOvrflw;
  db2int32                    iAllNodeFlag;
  db2int32                    iNumNodes;
   SQL_PDB_NODE_TYPE                   *piNodeList;
  db2int32                    iNumNodeInfo;
  db2Uint32                   iRollforwardFlags;
  char                        *piHistoryFile;
  db2Uint32                   iNumChngHistoryFile;
  struct sqlu_histFile        *piChngHistoryFile;
} db2RecoverStruct;
/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2gRecover */
/* ... */
SQL_API_RC SQL_API_FN
  db2gRecover (
  db2Uint32 versionNumber,
       void * pDB2gRecoverStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRecoverStruct
```

```
{
  char                        *piSourceDBAlias;
  db2Uint32                   iSourceDBAliasLen;
  char                        *piUserName;
   db2Uint32 iUserNameLen;
  char                        *piPassword;
   db2Uint32 iPasswordLen;
  db2Uint32                   iRecoverCallerAction;
   db2Uint32 iOptions;
  sqlint32                    *poNumReplies;
  struct sqlurf_info          *poNodeInfo;
  char                        *piStopTime;
  db2Uint32                   iStopTimeLen;
  char                        *piOverflowLogPath;
  db2Uint32                   iOverflowLogPathLen;
  db2Uint32                   iNumChngLgOvrflw;
  struct sqlurf_newlogpath    *piChngLogOvrflw;
  db2int32                    iAllNodeFlag;
  db2int32                    iNumNodes;
   SQL_PDB_NODE_TYPE                  *piNodeList;
  db2int32                    iNumNodeInfo;
  db2Uint32                   iRollforwardFlags;
  char                        *piHistoryFile;
  db2Uint32                   iHistoryFileLen;
  db2Uint32                   iNumChngHistoryFile;
  struct sqlu_histFile        *piChngHistoryFile;
} db2gRecoverStruct;
/* ... */
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2RecoverStruct*, übergeben wird.

pDB2RecoverStruct
Eingabe. Ein Zeiger auf die Struktur *db2RecoverStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

piSourceDBAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der wiederherzustellenden Datenbank enthält.

iSourceDBAliasLen
Länge von *piSourceDBAlias* in Byte.

piUserName
Eingabe. Eine Zeichenfolge, die den Benutzernamen enthält, der beim Versuch einer Verbindung verwendet wird. Kann NULL sein.

iUserNameLen
Länge von *piUsername* in Byte.

piPassword
Eingabe. Eine Zeichenfolge, die das Kennwort enthält, das mit dem Benutzernamen verwendet werden soll. Kann NULL sein.

iPasswordLen
Länge von *piPassword* in Byte.

iRecoverCallerAction
Eingabe. Gültige Werte:

DB2RESTORE_NOINTERRUPT
Startet die Wiederherstellungsoperation. Gibt an, dass die Wiederherstellung nicht überwacht ausgeführt wird und dass Szenarios, die normalerweise Benutzereingriff erfordern, entweder versucht werden, ohne vorher an das aufrufende Programm zurückgegeben zu werden, oder einen Fehler generieren. Verwenden Sie diese Aktion z. B. dann, wenn bekannt ist, dass alle für die Wiederherstellung erforderlichen Datenträger angehängt wurden, und Dienstprogrammeingabeaufforderungen nicht erwünscht sind.

DB2RESTORE_CONTINUE
Setzt die Wiederherstellung fort, nachdem der Benutzer die vom Dienstprogramm angeforderten Aktionen ausgeführt hat (z. B. das Anhängen eines neuen Bandes).

DB2RESTORE_TERMINATE
Beendet die Wiederherstellungsoperation, nachdem der Benutzer eine vom Dienstprogramm angeforderte Aktion nicht ausführen konnte.

DB2RESTORE_DEVICE_TERMINATE
Entfernt eine bestimmte Einheit aus der Liste der vom Wiederherstellungsdienstprogramm verwendeten Einheiten. Wenn eine bestimmte Einheit ihre Eingabe erschöpft hat, gibt das Wiederherstellungsdienstprogramm eine Warnung an das aufrufende Programm zurück. Rufen Sie das Wiederherstellungsdienstprogramm erneut mit dieser Aktion auf, um die Einheit, die die Warnung generiert hat, aus der Liste der verwendeten Einheiten zu entfernen.

DB2RESTORE_PARM_CHK
Parameter auswerten, ohne eine Wiederherstellungsoperation auszuführen. Diese Option beendet die Datenbankverbindung nicht, nachdem der Aufruf zurückgegeben wurde. Nach einer erfolgreichen Rückgabe dieses Aufrufs wird erwartet, dass der Benutzer einen Aufruf mit `DB2RESTORE_CONTINUE` absetzt, um mit der Aktion fortzufahren.

DB2RESTORE_PARM_CHK_ONLY
Parameter auswerten, ohne eine Wiederherstellungsoperation auszuführen. Bevor dieser Aufruf zurückgegeben wird, wird die durch diesen Aufruf hergestellte Datenbankverbindung beendet, und es ist kein darauf folgender Aufruf erforderlich.

DB2RESTORE_TERMINATE_INCRE
Beendet eine Teilwiederherstellungsoperation vor der Fertigstellung.

DB2ROLLFORWARD_LOADREC_CONT
Verwendung der Einheit fortsetzen, die die Warnung generiert hat (zum Beispiel, wenn ein neues Band eingelegt wurde).

DB2ROLLFORWARD_DEVICE_TERM
Verwendung der Einheit beenden, die die Warnung generiert hat (zum Beispiel, wenn keine Bänder mehr vorhanden sind).

DB2ROLLFORWARD_LOAD_REC_TERM
Alle von der Wiederherstellung verwendeten Einheiten beenden.

iOptions
Eingabe. Gültige Werte:

DB2RECOVER_EMPTY_FLAG
Keine Markierungen angegeben.

DB2RECOVER_LOCAL_TIME
Gibt an, dass der für die Stoppzeit *piStopTime* angegebene Wert in Ortszeit und nicht in WEZ angegeben ist. Dies ist die Standardeinstellung.

DB2RECOVER_GMT_TIME
Gibt an, dass der für die Stoppzeit *piStopTime* angegebene Wert in WEZ (Westeuropäische Zeit) angegeben ist.

poNumReplies
Ausgabe. Die Anzahl erhaltener Antworten.

poNodeInfo
Ausgabe. Antwortinformationen der Datenbankpartition.

piStopTime
Eingabe. Eine Zeichenfolge, die eine Zeitmarke in ISO-Format enthält. Die Datenbankwiederherstellung wird gestoppt, wenn diese Zeitmarke überschritten wird. Geben Sie SQLUM_INFINITY_TIMESTAMP an, um so weit wie möglich aktualisierend wiederherzustellen. Der Parameter kann für DB2ROLLFORWARD_QUERY, DB2ROLLFORWARD_PARM_CHECK und alle Wiederherstellungsaktionen (DB2ROLLFORWARD_LOADREC_) des aufrufenden Programms NULL sein.

iStopTimeLen
Länge von *piStopTime* in Byte.

piOverflowLogPath
Eingabe. Dieser Parameter wird verwendet, um einen alternativ zu verwendenden Protokollpfad anzugeben. Zusätzlich zu den aktiven Protokolldateien müssen Archivprotokolldateien (durch den Benutzer) an die durch *logpath* angegebene Position versetzt werden, bevor sie von diesem Dienstprogramm verwendet werden können. Dies könnte ein Problem darstellen, wenn der Benutzer nicht über genügend Speicherbereich im Protokollpfad verfügt. Aus diesem Grund wird der Überlaufprotokollpfad angegeben. Während der aktualisierenden Wiederherstellung werden die erforderlichen Protokolldateien zuerst im Protokollpfad und danach im Überlaufprotokollpfad gesucht. Die erforderlichen Protokolldateien für die aktualisierende Wiederherstellung von Tabellenbereichen können entweder in den Protokollpfad oder in den Überlaufprotokollpfad gestellt werden. Wenn das aufrufende Programm keinen Überlaufprotokollpfad angibt, ist der Standardwert der angegebene Pfad. In einer Umgebung mit partitionierten Datenbanken muss der Überlaufprotokollpfad ein gültiger, vollständig qualifizierter Pfad sein. Der Standardpfad ist der Standardüberlaufprotokollpfad für jede Datenbankpartition. In einer Umgebung mit einer einzelnen Datenbankpartition kann der Überlaufprotokollpfad relativ sein, wenn der Server lokal ist.

iOverflowLogPathLen
Länge von *piOverflowLogPath* in Byte.

iNumChngLgOvrflw
Eingabe. Nur Umgebungen mit partitionierten Datenbanken. Die Anzahl geänderter Überlaufprotokollpfade. Diese neuen Protokollpfade überschreiben nur den Standardüberlaufprotokollpfad für den angegebenen Datenbankpartitionsserver.

piChngLogOvrflw
Eingabe. Nur Umgebungen mit partitionierten Datenbanken. Ein Zeiger auf eine Struktur, die die vollständig qualifizierten Namen der geänderten Überlaufprotokollpfade enthält. Diese neuen Protokollpfade überschreiben nur den Standardüberlaufprotokollpfad für den angegebenen Datenbankpartitionsserver.

iAllNodeFlag
Eingabe. Nur Umgebungen mit partitionierten Datenbanken. Gibt an, ob die aktualisierende Wiederherstellungsoperation auf alle in `db2nodes.cfg` definierten Datenbankpartitionsserver angewendet werden soll. Gültige Werte:

DB2_NODE_LIST
Auf Datenbankpartitionsserver in einer Liste anwenden, die in *piNodeList* übergeben wird.

DB2_ALL_NODES
Auf alle Datenbankpartitionsserver anwenden. *piNodeList* sollte NULL sein. Dies ist der Standardwert.

DB2_ALL_EXCEPT
Auf alle Datenbankpartitionsserver anwenden außer auf die in einer Liste aufgeführten Datenbankpartitionsserver, die in *piNodeList* übergeben wird.

DB2_CAT_NODE_ONLY
Nur auf die Katalogtabellenpartition anwenden. *piNodeList* sollte NULL sein.

iNumNodes
Eingabe. Gibt die Anzahl Datenbankpartitionsserver im Bereich *piNodeList* an.

piNodeList
Eingabe. Ein Zeiger auf einen Bereich mit Nummern von Datenbankpartitionsservern, auf denen die aktualisierende Wiederherstellung ausgeführt werden soll.

iNumNodeInfo
Eingabe. Definiert die Größe des Ausgabeparameters *poNodeInfo*, der groß genug sein muss, um Statusinformationen für jede aktualisierend wiederherzustellende Datenbankpartition zu enthalten. In einer Umgebung mit einer einzelnen Datenbankpartition sollte dieser Parameter auf 1 gesetzt werden. Der Wert dieses Parameters sollte gleich der Anzahl Datenbankpartitionsserver sein, für die diese API aufgerufen wird.

RollforwardFlags
Eingabe. Gibt die Markierungen für aktualisierende Wiederherstellung an. Gültige Werte (definiert in `db2ApiDf.h`):

DB2ROLLFORWARD_EMPTY_FLAG
Keine Markierungen angegeben.

DB2ROLLFORWARD_LOCAL_TIME
Ermöglicht dem Benutzer die aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt in Ortszeit anstatt in WEZ. Dies erleichtert Benutzern die aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt auf ihren lokalen Maschinen und vermeidet potenzielle Benutzerfehler aufgrund der Umsetzung von Ortszeit in WEZ.

piHistoryFile
Verlaufsdatei.

iHistoryFileLen
Länge von *piHistoryFile* in Byte.

iNumChngHistoryFile
Anzahl Verlaufsdateien in der Liste.

piChngHistoryFile
Eine Liste der Verlaufsdateien.

Zugehörige Referenzen:

- „RECOVER DATABASE" auf Seite 171

Teil 2. Hohe Verfügbarkeit

Kapitel 6. Einführung in die Unterstützung der hohen Verfügbarkeit und der Funktionsübernahme

Erfolgreiche e-business Unternehmen hängen von der ständigen Verfügbarkeit von Transaktionsverarbeitungssystemen ab, welche ihrerseits von Datenbankverwaltungssystemen gesteuert werden, z. B. von DB2. Diese Systeme müssen rund um die Uhr und an allen Wochentagen verfügbar sein („24 x 7"). In diesem Abschnitt wird Folgendes behandelt:

- „Hohe Verfügbarkeit"
- „Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank" auf Seite 191
- „Fehlermonitorfunktion für UNIX-Systeme" auf Seite 196
- „Befehl db2fm - DB2-Fehlermonitor" auf Seite 198

Hohe Verfügbarkeit

Mit *hoher Verfügbarkeit* (HA - High Availability) wird ausgedrückt, dass Systeme aus der Sicht von Kunden praktisch immer aktiv und verfügbar sind. Dafür müssen die folgenden Voraussetzungen erfüllt sein:

- Transaktionen müssen effizient verarbeitet werden, ohne dass sich die Leistung in Zeiten von Spitzenbelastungen nennenswert vermindert (oder es gar zu einem völligen Leistungsausfall kommt). In einer Umgebung mit partitionierten Datenbanken kann DB2® sowohl die partitionsinterne Parallelität als auch die partitionsübergreifende Parallelität nutzen, um Transaktionen effizient zu verarbeiten. Die *partitionsinterne Parallelität* kann in einer SMP-Umgebung eingesetzt werden, um verschiedene Komponenten einer komplexen SQL-Anweisung gleichzeitig zu verarbeiten. Die *partitionsübergreifende Parallelität* in einer Umgebung mit partitionierten Datenbanken bedeutet dagegen, dass eine Abfrage auf allen beteiligten Knoten gleichzeitig verarbeitet wird, wobei jeder Knoten eine Untermenge von Tabellenzeilen verarbeitet.
- Die Systeme müssen nach einem Hardware- oder Softwarefehler oder einem Störfall schnell wiederherstellbar sein. DB2 verfügt über ein erweitertes System für fortlaufende Prüfpunktverfahren und eine Funktion für parallele Wiederherstellung, die eine sehr schnelle Wiederherstellung nach einem Systemabsturz ermöglichen.

  Die schnelle Wiederherstellung hängt auch vom Vorhandensein einer getesteten Sicherungs- und Wiederherstellungsstrategie ab.
- Software für die Unternehmensdatenbanken muss für die Transaktionsverarbeitung ständig aktiv und verfügbar sein. Dazu muss bei einem Ausfall des Datenbankmanagers ein anderer Datenbankmanager dessen Aufgaben übernehmen können. Dies wird als Funktionsübernahme bezeichnet. Die *Funktionsübernahme* ermöglicht bei einem Hardwarefehler eine automatische Übertragung der Arbeitsbelastung von einem System auf ein anderes System.

Die Funktionsübernahmefunktionalität können Sie mit der Datenbankreplikationsfunktion HADR (High Availability Disaster Recovery) implementieren. HADR ist eine Hochverfügbarkeitslösung und bietet Schutz vor Datenverlust, indem Änderungen aus einer Quellendatenbank, der so genannten Primärdatenbank, in eine Zieldatenbank, der so genannten Bereitschaftsdatenbank, repliziert werden.

Zur Funktionsübernahme kann auch auf einer anderen Maschine eine Datenbankkopie gepflegt werden, die ständig die Protokolldateien aktualisierend wiederherstellt. Die *Protokollübertragung* ist das Kopieren ganzer Protokolldateien auf eine Bereitschaftsmaschine, entweder von einer Archivierungseinheit aus oder mit Hilfe eines Benutzerexitprogramms, das für die primäre Datenbank ausgeführt wird. Mit dieser Methode wird die primäre Datenbank auf der Bereitschaftsmaschine wiederhergestellt, wobei das DB2-Wiederherstellungsdienstprogramm oder die Funktion zur Erstellung einer Spiegeldatenbank verwendet wird. Sie können auch die Unterstützung für ausgesetzte E/A verwenden, um die neue Datenbank schnell zu initialisieren. Die Bereitschaftsdatenbank der Bereitschaftsmaschine stellt die Protokolldateien ständig aktualisierend wiederher. Wenn die primäre Datenbank ausfällt, werden alle übrigen Protokolldateien auf die Bereitschaftsmaschine kopiert. Nach einer aktualisierenden Wiederherstellung bis zum Ende der Protokolle und bis zur Stoppoperation werden alle Clients mit der Bereitschaftsdatenbank der Bereitschaftsmaschine verbunden.

Die Unterstützung der Funktionsübernahme erhalten Sie auch über plattformspezifische Software, die Sie auf dem System installieren können. Beispiel:

- Tivoli® System Automation für Linux

  Weitere Informationen zu Tivoli System Automation finden Sie im White Paper „Highly Available DB2 Universal Database™ using Tivoli System Automation for Linux", das auf der Website „DB2 UDB and DB2 Connect Online Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar ist.
- High Availability Cluster Multi-Processing, Enhanced Scalability für AIX®

  Weitere Informationen zu HACMP/ES finden Sie im White Paper „IBM® DB2 Universal Database™ Enterprise Edition for AIX® and HACMP/ES", das auf der Website „DB2 UDB and DB2 Connect Online Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar ist.
- Microsoft® Cluster Server für Windows®-Betriebssysteme

  Informationen zu Microsoft Cluster Server finden Sie im folgenden White Paper, das auf der Website „DB2 UDB and DB2 Connect Online Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar ist: „Implementing IBM DB2 Universal Database V8.1 Enterprise Server Edition with Microsoft Cluster Server".
- Sun™ Cluster oder VERITAS Cluster Server für die Solaris™-Betriebsumgebung

  Informationen zu Sun Cluster 3.0 finden Sie im White Paper „DB2 Universal Database and High Availability on Sun Cluster 3.X", das auf der Website „DB2 UDB and DB2 Connect Online Support" unter http://www.ibm.com/software/data/pubs/papers/ 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

  Weitere Informationen zu HP MC/ServiceGuard finden Sie in dem White Paper, in dem die Implementierung von IBM DB2 ESE V8.1 mit HP MC/ServiceGuard besprochen wird. Auf dieses White Paper können Sie über die Website „IBM DB2 Information Management Products for HP" unter http://www-306.ibm.com/software/data/hp/ zugreifen.

Strategien zur Funktionsübernahme basieren normalerweise auf Clustern aus mehreren Systemen. Ein *Cluster* ist eine Gruppe verbundener Systeme, die nach außen hin wie ein einzelnes System zusammenarbeiten. Jede physische Maschine in

einem Cluster enthält mindestens einen logischen Knoten. Durch das Clustering können Server sich gegenseitig gegen Ausfälle schützen, indem sie die Arbeitsbelastung des ausgefallenen Servers übernehmen.

Die IP-Adressübernahme (IP-Übernahme) ist die Fähigkeit, bei einem Ausfall eines Servers eine Server-IP-Adresse von einer Maschine auf eine andere zu übertragen. Aus der Sicht der Clientanwendung erscheinen die Maschinen zu unterschiedlichen Zeiten wie ein und derselbe Server.

In der Software für Funktionsübernahme können *Überwachungssignale* oder *Keepalive-Pakete* zwischen Systemen verwendet werden, um die Verfügbarkeit zu bestätigen. Überwachungssignale bedeuten, dass Systemservices zwischen allen Knoten eines Clusters ständig kommunizieren. Wenn kein Überwachungssignal erkannt wird, beginnt die Funktionsübernahme durch ein Ausweichsystem. Endbenutzer bemerken gewöhnlich nicht, dass ein System ausgefallen ist.

Die zwei häufigsten Strategien für Funktionsübernahme auf dem Markt sind der *Bereitschaftsmodus (Idle Standby)* und der *Modus der gegenseitigen Übernahme (Mutual Takeover)*, obwohl die Konfigurationen, die zu diesen Modi gehören, je nach Lieferant auch anders bezeichnet werden können:

Bereitschaftsmodus (Idle Standby)
Bei dieser Konfiguration führt ein System ein DB2-Exemplar aus; das zweite System befindet sich „im Bereitschaftsmodus" (Idle Standby) und ist bereit, das Exemplar zu übernehmen, falls das Betriebssystem oder die Hardware für das erste System ausfällt. Auf die Gesamtleitung des Systems wirkt sich dies nicht aus, da das Ausweichsystem erst bei Bedarf eingesetzt wird.

Gegenseitige Übernahme (Mutual Takeover)
Bei dieser Konfiguration ist jedes System als Ausweichsystem für ein anderes zugeordnet. Dies kann sich auf die Gesamtleistung des Systems auswirken, da das Ausweichsystem nach einer Funktionsübernahme zusätzliche Arbeit leisten muss: Es muss die eigenen Aufgaben erfüllen und zusätzlich die Aufgaben des ausgefallenen Systems.

Mit Strategien zur Funktionsübernahme können die Funktionen eines Exemplars, einer Partition oder mehrerer logischer Knoten übernommen werden.

Zugehörige Konzepte:
- „Parallelität" im Handbuch *Systemverwaltung: Konzept*
- „Entwickeln einer Sicherungs- und Wiederherstellungsstrategie" auf Seite 3
- „Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank" auf Seite 191
- „Clusterunterstützung für die Solaris-Betriebsumgebung" auf Seite 251
- „Unterstützung für Sun Cluster 3.0" auf Seite 254
- „Unterstützung für VERITAS Cluster Server" auf Seite 257
- „Unterstützung für Microsoft Cluster Server" auf Seite 247
- „Unterstützung für High Availability Cluster Multi-Processing (HACMP)" auf Seite 241
- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201

Hohe Verfügbarkeit durch Protokollübertragung

Die Protokollübertragung ist das Kopieren ganzer Protokolldateien auf eine Bereitschaftsmaschine, entweder von einer Archivierungseinheit aus oder mit Hilfe 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 und 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 haben wie die Produktionsdatenbank. Mit dieser Methode wird die primäre Datenbank auf der Bereitschaftsmaschine wiederhergestellt, wobei das Wiederherstellungsdienstprogramm oder die Funktion zur Erstellung einer Spiegeldatenbank verwendet wird.

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, NVRAM-Cache (lokal und fern) zu verwenden, um die Leistungsauswirkungen einer Wiederherstellungssituation nach einem Katastrophenfall zu minimieren.

Anmerkungen:

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 Wiederherstellung anstehend* genommen wurde. Wenn auf dem Primärserver Indizes wiederhergestellt werden, wird empfohlen, den Bereitschaftsserver erneut mit dem Primärserver zu synchronisieren.
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 *Wiederherstellung anstehend* versetzt wird.
4. Eine Bereitschaftsmaschine kann auf zwei Weisen initialisiert werden:
   a. Durch Wiederherstellen auf die Maschine von einem Sicherungsimage.
   b. Durch Erstellen einer geteilten Spiegeldatenbank des Produktionssystems und Absetzen des Befehls **db2inidb** mit der Option STANDBY.

Sie können den Befehl ROLLFORWARD erst dann auf dem fehlertoleranten System absetzen, wenn die Bereitschaftsmaschine initialisiert wurde.

5. Nicht protokollierte Operationen werden in der Bereitschaftsdatenbank nicht wiederholt. Daher wird empfohlen, dass Sie die Bereitschaftsdatenbank nach solchen Operationen erneut synchronisieren. Dies erreichen Sie durch Onlineteilung einer Spiegeldatenbank und Unterstützung der ausgesetzten E/A.

Zugehörige Konzepte:

- „Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank" auf Seite 191
- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201

Zugehörige Tasks:

- „Verwenden einer geteilten Spiegeldatenbank als Bereitschaftsdatenbank" auf Seite 194

Zugehörige Referenzen:

- Anhang G, „Benutzerexit zur Datenbankwiederherstellung", auf Seite 369

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

Ausgesetzte E/A unterstützt die ständige Systemverfügbarkeit, indem für die Bearbeitung der Onlineteilung von Spiegeldatenbanken eine vollständige Implementierung bereitgestellt wird, d. h. zum Teilen von Spiegeldatenbanken muss das System nicht heruntergefahren werden. Eine *geteilte Spiegeldatenbank* ist eine „Momentaufnahme" der Datenbank, die erstellt wird, indem die Festplatten mit den Daten gespiegelt werden; wenn eine Kopie benötigt wird, wird die Spiegeldatenbank geteilt. *Plattenspiegelung* ist das Schreiben aller Daten auf zwei separate Festplatten, wobei eine Platte das Spiegelbild der anderen ist. Das *Teilen* (Splitting) einer Spiegeldatenbank ist der Prozess des Trennens der primären und der sekundären Kopien der Datenbank.

Wenn Sie vermeiden möchten, eine große Datenbank mit dem DB2®-Sicherungsdienstprogramm zu sichern, können Sie mit der Funktion für ausgesetzte E/A und der Funktion zur Erstellung einer Spiegeldatenbank von einem gespiegelten Image Kopien erstellen. Außerdem erreichen Sie mit dieser Methode Folgendes:

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

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

- Als Klondatenbank
- Als Bereitschaftsdatenbank
- Als Sicherungsimage

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 Partitionen gleichzeitig aussetzen. Sie können die E/A für eine Untergruppe von mindestens einer Partition aussetzen, um zum Ausführen von Offlinesicherungen geteilte Spiegeldatenbanken zu erstellen. Wenn in dieser Untergruppe der Katalogknoten enthalten ist, muss das Aussetzen auf dieser Partition zuletzt erfolgen.

In einer Umgebung mit partitionierten Datenbanken müssen Sie den Befehl **db2inidb** auf jeder Partition ausführen, bevor Sie das geteilte Image einer beliebigen Partition verwenden können. Sie können das Tool auf allen Partitionen 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 Partitionen gleichzeitig ausführen. Für jede Partition muss eine eigene Konfigurationsdatei angegeben werden, die den Wert NODENUM der zu ändernden Partition enthält. Wenn z. B. der Name der Datenbank geändert wird, sind alle Partitionen betroffen, und der Befehl `db2relocatedb` muss auf jeder Partition mit einer eigenen Konfigurationsdatei ausgeführt werden. Wenn Behälter versetzt werden, die zu einer einzelnen Datenbankpartition gehören, muss der Befehl **db2relocatedb** nur einmal auf dieser Partition ausgeführt werden.

Anmerkung: Stellen Sie sicher, dass die geteilte Spiegeldatenbank alle Behälter und Verzeichnisse enthält, aus denen die Datenbank besteht, einschließlich des Datenträgerverzeichnisses.

Zugehörige Referenzen:

- „db2inidb - Spiegeldatenbank initialisieren" auf Seite 285

Onlineteilung einer Spiegeldatenbank

Verwenden einer geteilten Spiegeldatenbank zum Klonen einer Datenbank

Verwenden Sie zum Erstellen einer Klondatenbank die folgende Prozedur. 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.

Einschränkungen:

Sie können die geklonte Datenbank weder sichern, noch ihr Image auf dem Originalsystem wiederherstellen oder es mit Hilfe der Protokolldateien, die auf dem Originalsystem generiert wurden, aktualisierend wiederherstellen.

Prozedur:

Gehen Sie wie folgt vor, um eine Datenbank zu klonen:

1. Setzen Sie für die primäre Datenbank die E/A aus:

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

2. Verwenden Sie geeignete Befehle auf Betriebssystemebene, um die Spiegeldatenbank(en) aus der primären Datenbank zu erstellen.

Anmerkung: Stellen Sie sicher, dass Sie das gesamte Datenbankverzeichnis einschließlich des Datenträgerverzeichnisses kopieren. Sie müssen auch das Protokollverzeichnis und alle Behälterverzeichnisse kopieren, die außerhalb des Datenbankverzeichnisses vorhanden sind.

3. Nehmen Sie den E/A-Betrieb auf der primären Datenbank wieder auf:
   ```
   db2 set write resume for database
   ```
4. Katalogisieren Sie die gespiegelte Datenbank im sekundären System.

   Anmerkung: Eine gespiegelte Datenbank kann standardmäßig nicht in demselben System vorhanden sein wie die Primärdatenbank. Sie muss sich in einem sekundären System befinden, das dieselbe Verzeichnisstruktur aufweist und denselben Exemplarnamen wie die Primärdatenbank verwendet. Falls sich die gespiegelte Datenbank auf demselben System wie die Primärdatenbank befinden muss, können Sie dies mit Hilfe des Dienstprogramms **db2relocatedb** oder der Option RELOCATE USING im Befehl **db2inidb** erreichen.
5. Starten Sie das Datenbankexemplar auf dem sekundären System:
   ```
   db2start
   ```
6. 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 `d2inidb` an, um die Klondatenbank zu verlagern:
   ```
   db2inidb aliasname-der-datenbank as snapshot relocate using relocatedbcfg.txt
   ```

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

   Anmerkungen:
   a. Mit diesem Befehl werden Transaktionen rückgängig gemacht, die während des Teilens gerade verarbeitet werden, und es wird eine neue Protokollkettenreihenfolge begonnen, so dass Protokolle aus der primären Datenbank nicht für die Klondatenbank wiederholt werden können.
   b. Das Datenbankverzeichnis (einschließlich des Datenträgerverzeichnisses), das Protokollverzeichnis und die Behälterverzeichnisse müssen an die gewünschte Position versetzt werden, bevor Sie die Option RELOCATE USING verwenden.

Zugehörige Konzepte:

- „Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank" auf Seite 191

Zugehörige Tasks:

- „Verwenden einer geteilten Spiegeldatenbank als Bereitschaftsdatenbank" auf Seite 194
- „Verwenden einer geteilten Spiegeldatenbank als Sicherungsimage" auf Seite 195

Zugehörige Referenzen:

- „db2relocatedb - Relocate Database Command" im Handbuch *Command Reference*
- „SET WRITE Command" im Handbuch *Command Reference*

Verwenden einer geteilten Spiegeldatenbank als Bereitschaftsdatenbank

Verwenden Sie die folgende Prozedur, um eine geteilte Spiegeldatenbank einer Datenbank zu erstellen, die als Bereitschaftsdatenbank verwendet werden soll. Wenn in der Primärdatenbank ein Fehler auftritt und eine Wiederherstellung nach einem Systemabsturz erforderlich wird, können Sie die Bereitschaftsdatenbank die Rolle der Primärdatenbank übernehmen lassen.

Prozedur:

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

1. Setzen Sie für die primäre Datenbank die E/A aus:
   ```
   db2 set write suspend for database
   ```
2. Verwenden Sie geeignete Befehle auf Betriebssystemebene, um die Spiegeldatenbank(en) aus der primären Datenbank zu erstellen.

   Anmerkung: Stellen Sie sicher, dass Sie das gesamte Datenbankverzeichnis einschließlich des Datenträgerverzeichnisses kopieren. Sie müssen auch das Protokollverzeichnis und alle Behälterverzeichnisse kopieren, die außerhalb des Datenbankverzeichnisses vorhanden sind.
3. Nehmen Sie den E/A-Betrieb auf der primären Datenbank wieder auf:
   ```
   db2 set write resume for database
   ```
4. Katalogisieren sie die gespiegelte Datenbank im sekundären System.

   Anmerkung: Eine gespiegelte Datenbank kann standardmäßig nicht in demselben System vorhanden sein wie die Primärdatenbank. Sie muss sich in einem sekundären System befinden, das dieselbe Verzeichnisstruktur aufweist und denselben Exemplarnamen als Primärdatenbank verwendet. Falls sich die gespiegelte Datenbank auf demselben System wie die Primärdatenbank befinden muss, können Sie dies mit Hilfe des Dienstprogramms **db2relocatedb** oder der Option RELOCATE USING des Befehls **db2inidb** erreichen.
5. Starten Sie das Datenbankexemplar auf dem sekundären System:
   ```
   db2start
   ```
6. Initialisieren Sie die gespiegelte Datenbank auf dem sekundären System, indem Sie sie in den Status für anstehende aktualisierende Wiederherstellung versetzen:
   ```
   db2inidb aliasname-der-datenbank as standby
   ```

   Falls erforderlich, geben Sie die Option RELOCATE USING des Befehls `d2inidb` an, um die Bereitschaftsdatenbank zu verlagern:
   ```
   db2inidb aliasname-der-datenbank as standby relocate using relocatedbcfg.txt
   ```

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

   Anmerkungen:

   a. Wenn Sie nur über DMS-Tabellenbereiche (vom Datenbankmanager verwaltete Tabellenbereiche) verfügen, können Sie eine vollständige

Datenbanksicherung erstellen, damit der Systemaufwand für eine Sicherung auf der Produktionsdatenbank entfällt.

b. Das Datenbankverzeichnis (einschließlich des Datenträgerverzeichnisses), das Protokollverzeichnis und die Behälterverzeichnisse müssen an die gewünschte Position versetzt werden, bevor Sie die Option RELOCATE USING verwenden.

7. Definieren Sie ein Benutzerexitprogramm, um die Protokolldateien vom primären System abzurufen.
8. Stellen Sie die Datenbank bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt aktualisierend wieder her.
9. 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.
10. Wenn Sie die Bereitschaftsdatenbank in den Onlinestatus versetzen wollen, setzen Sie den Befehl ROLLFORWARD mit der Option STOP ab.

Anmerkung: Die Protokolle der primären Datenbank können nicht mehr auf die gespiegelte Datenbank angewandt werden, nachdem für sie der Status für anstehende aktualisierende Wiederherstellung aufgehoben wurde.

Zugehörige Konzepte:

- „Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank" auf Seite 191

Zugehörige Tasks:

- „Verwenden einer geteilten Spiegeldatenbank zum Klonen einer Datenbank" auf Seite 192
- „Verwenden einer geteilten Spiegeldatenbank als Sicherungsimage" auf Seite 195

Zugehörige Referenzen:

- „db2inidb - Spiegeldatenbank initialisieren" auf Seite 285
- „db2relocatedb - Relocate Database Command" im Handbuch *Command Reference*
- „SET WRITE Command" im Handbuch *Command Reference*

Verwenden einer geteilten Spiegeldatenbank als Sicherungsimage

Verwenden Sie die folgende Prozedur, um eine geteilte Spiegeldatenbank einer Primärdatenbank zu erstellen, die als Sicherungsimage verwendet werden soll. Diese Prozedur kann an Stelle von Datenbanksicherungsoperationen für die Primärdatenbank verwendet werden.

Prozedur:

Gehen Sie wie folgt vor, um eine geteilte Spiegeldatenbank als „Sicherungsimage" zu verwenden:

1. Setzen Sie für die primäre Datenbank die E/A aus:

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

2. Verwenden Sie geeignete Befehle auf Betriebssystemebene, um die Spiegeldatenbank(en) aus der primären Datenbank zu erstellen.

3. Nehmen Sie den E/A-Betrieb auf der primären Datenbank wieder auf:
   ```
   db2 set write resume for database
   ```
4. Auf dem primären System tritt ein Fehler auf, der eine Wiederherstellung aus einer Sicherung erforderlich macht.
5. Stoppen Sie das primäre Datenbankexemplar:
   ```
   db2stop
   ```
6. Verwenden Sie Befehle auf Betriebssystemebene, um die geteilten Daten auf das primäre System zu kopieren. **Kopieren Sie nicht die geteilten Protokolldateien**, da die primären Protokolle für die aktualisierende Wiederherstellung benötigt werden.
7. Starten Sie das primäre Datenbankexemplar:
   ```
   db2start
   ```
8. Initialisieren Sie die primäre Datenbank:
   ```
   db2inidb aliasname-der-datenbank as mirror
   ```
9. Stellen Sie die primäre Datenbank bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt aktualisierend wieder her.

Zugehörige Konzepte:

- „Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank" auf Seite 191

Zugehörige Tasks:

- „Verwenden einer geteilten Spiegeldatenbank zum Klonen einer Datenbank" auf Seite 192
- „Verwenden einer geteilten Spiegeldatenbank als Bereitschaftsdatenbank" auf Seite 194

Zugehörige Referenzen:

- „db2inidb - Spiegeldatenbank initialisieren" auf Seite 285

Fehlermonitorfunktion für UNIX-Systeme

Auf UNIX®-Systemen erhöht die Fehlermonitorfunktion (Fault Monitor Facility) die Verfügbarkeit der DB2®-Umgebungen ohne Cluster durch eine Reihe von zusammenarbeitenden Prozessen, die sicherstellen, dass DB2 aktiv ist. Dabei überwacht der Dämon *init* den Fehlermonitorkoordinator (Fault Monitor Coordinator, FMC), der FMC überwacht die Fehlermonitore, und die Fehlermonitore überwachen DB2. Der FMC ist der Prozess der Fehlermonitorfunktion, der während der UNIX-Bootfolge gestartet wird. Der Dämon *init* startet den FMC und startet ihn erneut, falls der FMC abnormal beendet wird. Der FMC startet für jedes DB2-Exemplar einen Fehlermonitor. Jeder Fehlermonitor wird als Dämonprozess ausgeführt und hat dieselben Benutzerzugriffsrechte wie das DB2-Exemplar. 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 ein DB2-Exemplar. Wenn das DB2-Exemplar vorzeitig beendet wird, wird es vom Fehlermonitor erneut gestartet.

Anmerkungen:

1. Wenn Sie ein Clusterprodukt mit hoher Verfügbarkeit verwenden (z. B. HACMP oder MSCS), muss die Fehlermonitorfunktion inaktiviert werden, da das Starten und Stoppen des Exemplars vom Clusterprodukt gesteuert wird.

2. Die Fehlermonitorfunktion kann nur durch Absetzen des Befehls **db2stop** inaktiviert werden. Wenn ein DB2-Exemplar auf andere Weise beendet wird, wird es von der Fehlermonitorfunktion erneut gestartet.

Registrierungsdatei für Fehlermonitore

Beim Starten des Fehlermonitordämons wird auf jeder physischen Maschine für jedes Exemplar eine Registrierungsdatei für Fehlermonitore erstellt. Die Werte in dieser Datei geben das Verhalten der Fehlermonitore an. Die Datei befindet sich im Verzeichnis /sqllib/ und hat den Namen fm.<maschinenname>.reg. Diese Datei kann mit Hilfe des Befehls **db2fm** geändert werden. Sie enthält folgende Einträge:

```
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 = <exemplarname>@<maschinenname>
```

Dabei gilt Folgendes:

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 dann Aktionen ausführen, wenn die Parameter FM_ON und FM_ACTIVE 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 DB2 beendet wird. Der Standardwert ist YES.

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

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

STATUS_TIMEOUT
Gibt die Zeitspanne an, innerhalb der der Fehlermonitor den Status des Services 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 Services an. Der Standardwert ist 20 Sekunden.

RESTART_RETRIES
Gibt an, wie oft der Fehlermonitor versuchen wird, nach einem fehlgeschlagenen Aufruf den Status des überwachten Services 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 Hinweisnachricht sendet. Der Standardwert ist <exemplarname>@<maschinenname>.

Diese Datei kann mit Hilfe des Befehls **db2fm** geändert werden. Beispiel:

Wenn Sie den Wert für START_TIMEOUT für das Exemplar DB2INST1 auf 100 Sekunden setzen wollen, setzen Sie folgenden Befehl ab:

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

Wenn Sie den Wert für STOP_TIMEOUT für das Exemplar DB2INST1 auf 200 Sekunden setzen wollen, setzen Sie folgenden Befehl ab:

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

Wenn Sie für das Exemplar DB2INST1 den Wert für START_TIMEOUT auf 100 Sekunden setzen wollen und Wert für STOP_TIMEOUT auf 200 Sekunden, setzen Sie folgenden Befehl ab:

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

Wenn Sie für das Exemplar DB2INST1 die Überwachung durch Fehlermonitore aktivieren wollen, setzen Sie folgenden Befehl ab:

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

Wenn Sie für das Exemplar DB2INST1 die Überwachung durch Fehlermonitore inaktivieren wollen, setzen Sie folgenden Befehl ab:

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

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

Zugehörige Referenzen:

- „Befehl db2fm - DB2-Fehlermonitor" auf Seite 198

Befehl db2fm - DB2-Fehlermonitor

Steuert den DB2-Fehlermonitordämon. Mit **db2fm** können Sie den Fehlermonitor konfigurieren.

Dieser Befehl ist nur auf Plattformen auf UNIX-Basis verfügbar.

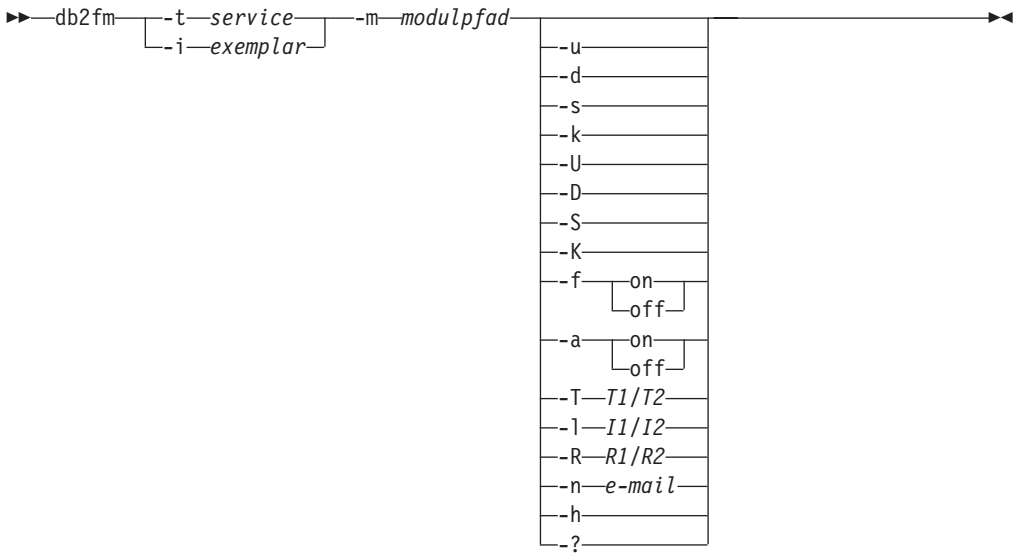
Berechtigung:

Eine Berechtigung für das Exemplar, für das Sie den Befehl ausführen.

Erforderliche Verbindung:

Keine.

Befehlssyntax:



Befehlsparameter:

-m *modulpfad*
Definiert den vollständigen Pfad der gemeinsam benutzten Bibliothek des Fehlermonitors für das überwachte Produkt. Der Standardwert lautet $INSTANCEHOME/sqllib/lib/libdb2gcf.

-t *service*
Gibt den eindeutigen Textdeskriptor für einen Service an.

-i *exemplar*
Definiert das Exemplar des Services.

-u Aktiviert den Service.

-U Aktiviert den Fehlermonitordämon.

-d Inaktiviert den Service.

-D Inaktiviert den Fehlermonitordämon.

-k Bricht den Service ab.

-K Bricht den Fehlermonitordämon ab.

-s Gibt den Status des Services zurück.

-S Gibt den Status des Fehlermonitordämons zurück.

Anmerkung: Der Service oder der Fehlermonitor kann einen der folgenden Status haben:

- Nicht ordnungsgemäß installiert
- INSTALLED PROPERLY aber NOT ALIVE
- ALIVE aber NOT AVAILABLE (Wartung)
- AVAILABLE
- UNKNOWN

-f *on | off*
Aktiviert oder inaktiviert den Fehlermonitor.

Anmerkung: Wenn diese Option auf *off* gesetzt ist, wird der Fehlermonitordämon nicht gestartet bzw. wird der Dämon beendet, falls er aktiv war.

-a *on | off*
Aktiviert oder inaktiviert die Fehlerüberwachung.

Anmerkung: Wenn diese Option auf *off* gesetzt ist, führt der Fehlermonitor keine aktive Überwachung durch, d. h. falls der Service inaktiv wird, versucht der Monitor nicht, ihn wieder zu aktivieren.

-T *T1/T2*
Überschreibt das Zeitlimit für die Start- und Stoppzeit.

Zum Beispiel:

- -T 15/10 aktualisiert beide Zeitlimitwerte entsprechend
- -T 15 aktualisiert das Zeitlimit für die Startzeit auf 15 Sekunden
- -T /10 aktualisiert das Zeitlimit für die Stoppzeit auf 10 Sekunden

-I *I1/I2*
Legt das Statusintervall und Zeitlimit entsprechend fest.

-R *R1/R2*
Legt die Anzahl Wiederholungen für die Statusmethode und Aktion fest, bevor keine weiteren Wiederholungsversuche mehr erfolgen.

-n *e-mail*
Legt die E-Mail-Adresse für Benachrichtigungen bei Ereignissen fest.

-h Gibt die Syntax aus.

-? Gibt die Syntax aus.

Kapitel 7. HADR (High Availability Disaster Recovery)

HADR (High Availability Disaster Recovery) - Übersicht

DB2® Universal Databases (DB2 UDB) HADR (High Availability Disaster Recovery) ist eine Datenbankreplikationsfunktion, die eine Hochverfügbarkeitslösung sowohl für teilweise als auch vollständige Ausfälle eines Standorts zur Verfügung stellt. HADR bietet Schutz vor Datenverlust durch Replizieren von Datenänderungen aus einer Quellendatenbank, der so genannten Primärdatenbank, in eine Zieldatenbank, der so genannten Bereitschaftsdatenbank. Eine Datenbank, die HADR nicht verwendet, wird als Standarddatenbank bezeichnet.

HADR ist besonders geeignet, wenn Sie die meisten oder alle Ihre Datenbanken schützen müssen oder wenn Sie DDL-Operationen ausführen, die automatisch in der Bereitschaftsdatenbank repliziert werden müssen.

Anwendungen können nur auf die aktuelle Primärdatenbank zugreifen. Aktualisierungen an der Bereitschaftsdatenbank erfolgen durch aktualisierendes Wiederherstellen von Protokolldaten, die auf der Primärdatenbank generiert und an die Bereitschaftsdatenbank übertragen werden.

Ein teilweiser Ausfall eines Standorts kann durch einen Hardware-, Software- (DB2 oder Betriebssystem) oder Netzausfall verursacht werden. Ohne HADR muss bei einem teilweisen Ausfall eines Standorts der Datenbankverwaltungssystemserver (DBMS - Database Management System) oder die Maschine, auf der sich die Datenbank befindet, erneut gestartet werden. Die für den Neustart der Datenbank und der Maschine, auf der sich die Datenbank befindet, erforderliche Zeit ist unvorhersehbar. Es kann einige Minuten in Anspruch nehmen, bevor die Datenbank wieder in einen konsistenten Zustand versetzt und verfügbar ist. Mit HADR kann die Bereitschaftsdatenbank innerhalb von Sekunden eingesetzt werden. Außerdem können Sie die automatische Clientweiterleitung oder Wiederholungslogik in der Anwendung nutzen, um die Clients, die die ursprüngliche Primärdatenbank verwendeten, an die Bereitschaftsdatenbank (die neue Primärdatenbank) umzuleiten.

Ein vollständiger Ausfall eines Standorts kann im Fall einer Katastrophe auftreten, beispielsweise wenn ein Standort durch einen Brand zerstört wird. Da HADR zur Kommunikation zwischen der Primärdatenbank und der Bereitschaftsdatenbank TCP/IP verwendet, können sich die Datenbanken an verschiedenen Standorten befinden. Ihre Primärdatenbank kann beispielsweise am Standort Ihrer Zentrale untergebracht sein, und die Bereitschaftsdatenbank kann sich in einer Vertriebsstelle in einer anderen Stadt befinden. Im Katastrophenfall wird 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 sichern und in ihren Status als Primärdatenbank zurückversetzen. Dies wird als Zurücksetzung bezeichnet.

Mit HADR können Sie die gewünschte Ebene für den Schutz vor potenziellem Datenverlust auswählen, indem Sie einen der drei Synchronisationsmodi angeben: synchron, fast synchron oder asynchron.

Wenn der ausgefallene primäre Server wieder instand gesetzt ist, kann er dem HADR-Paar als Bereitschaftsdatenbank hinzugefügt werden, sofern eine Konsistenz der beiden Datenbankkopien erzielt werden kann. Nachdem die ursprüngliche Primärdatenbank dem HADR-Paar als Bereitschaftsdatenbank hinzugefügt wurde, können Sie die Rollen der Datenbanken wieder tauschen, sodass die ursprüngliche Primärdatenbank erneut als aktuelle Primärdatenbank genutzt werden kann.

HADR ist nur eine von verschiedenen Replikationslösungen, die mit der DB2-Produktfamilie angeboten werden. Version 8.2 von DB2 Information Integrator und DB2 UDB enthalten SQL Replication- und Q Replication-Lösungen, die in einigen Konfigurationen ebenfalls verwendet werden können, um hohe Verfügbarkeit zu gewährleisten. Diese Funktionen 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, und sie können in Umgebungen mit partitionierten Datenbanken verwendet werden.

Zugehörige Konzepte:
- „Hohe Verfügbarkeit durch Protokollübertragung" auf Seite 190
- „Automatische Clientweiterleitung und HADR" auf Seite 214
- „Synchronisationsmodi für HADR" auf Seite 211
- „Systemvoraussetzungen für HADR" auf Seite 202
- „Einschränkungen für HADR" auf Seite 204
- „Datenbankkonfiguration für HADR" auf Seite 204
- „Vergleich zwischen der Q Replication und HADR" im Handbuch *IBM DB2 Information Integrator Replikation und Event-Publishing - Einführung*

Systemvoraussetzungen für HADR

Zum Erzielen einer optimalen Leistung mit HADR stellen Sie sicher, dass Ihr System den folgenden Anforderungen für Hardware, Software und DB2® Universal Database (DB2 UDB) entspricht.

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.

Hardware- und Betriebssystemvoraussetzungen:

Empfehlung: Verwenden Sie für die Primärdatenbank und die Bereitschaftsdatenbank identische Hostcomputer. D. h., 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.

DB2 UDB-Voraussetzungen:

Die von der Primärdatenbank und der Bereitschaftsdatenbank verwendeten Datenbankversionen müssen identisch sein. Während einem schrittweisen Upgrade kann die Datenbankversion der Bereitschaftsdatenbank vorübergehend neuer als die der Primärdatenbank sein. Die DB2 UDB-Version der Primärdatenbank darf nie neuer sein als die der Bereitschaftsdatenbank. Die DB2 UDB-Software für die Primärdatenbank und die Bereitschaftsdatenbank muss dieselbe Bitgröße (32 oder 64 Bit) verwenden. Die Tabellenbereiche und ihre Behälter müssen für die Primärdatenbank und die Bereitschaftsdatenbank identisch sein. Andere Merkmale, die ebenfalls identisch sein müssen, umfassen den Tabellenbereichstyp (DMS oder SMS), die Tabellenbereichsgröße, den Behälterpfad und den Dateityp des Behälters (unformatierte Einheit oder Dateisystem). Die Menge des für Protokolldateien zugeordneten Speicherbereichs sollte für die Primärdatenbank und die Bereitschaftsdatenbank ebenfalls identisch sein.

Wenn Sie eine Tabellenbereichsanweisung wie CREATE TABLESPACE, ALTER TABLESPACE oder DROP TABLESPACE in der Primärdatenbank absetzen, wird diese Anweisung in der Bereitschaftsdatenbank wiederholt. Sie müssen sicherstellen, dass die einbezogenen Einheiten in beiden Datenbanken definiert sind, bevor Sie die Tabellenbereichsanweisung in der Primärdatenbank absetzen.

Wenn die Tabellenbereichskonfiguration für die Primärdatenbank und die Bereitschaftsdatenbank nicht identisch sind, können bei der Protokollwiederholung in der Bereitschaftsdatenbank Fehler auftreten, z. B. "nicht genügend Speicher" oder "Tabellenbereichsbehälter nicht gefunden". Wenn dies der Fall ist, wird der betroffene Tabellenbereich in den Status *Aktualisierende Wiederherstellung anstehend* versetzt und wird bei nachfolgender Protokollwiederholung ignoriert. Im Fall einer Übernahmeoperation steht dieser Tabellenbereich den Anwendungen nicht zur Verfügung. Wenn die erforderlichen Sicherungsimages und Protokolldateiarchive verfügbar sind, können Sie den Tabellenbereich wiederherstellen, indem Sie zuerst die Anweisung SET TABLESPACE CONTAINERS mit der Option IGNORE ROLLFORWARD CONTAINER OPERATIONS absetzen und anschließend den Befehl ROLLFORWARD absetzen. Die Primärdatenbank und die Bereitschaftsdatenbank benötigen nicht denselben Datenbankpfad. Wenn relative Behälterpfade verwendet werden, entspricht derselbe relative Pfad möglicherweise verschiedenen absoluten Behälterpfaden in der Primärdatenbank und der Bereitschaftsdatenbank.

Pufferpoolvoraussetzungen:

Da Pufferpooloperationen auch in der Bereitschaftsdatenbank wiederholt werden, ist es wichtig, dass die Primärdatenbank und die Bereitschaftsdatenbank dieselbe Speicherkapazität haben.

Zugehörige Konzepte:

- „Einschränkungen für HADR" auf Seite 204

Zugehörige Tasks:

- „Ausführen eines schrittweisen Upgrades in einer HADR-Umgebung" auf Seite 235

Einschränkungen für HADR

In der folgenden Liste sind die für HADR geltenden Einschränkungen zusammengefasst:

- HADR wird nur von DB2® UDB Enterprise Server Edition (ESE) unterstützt. HADR wird jedoch nicht unterstützt, wenn Sie für ESE mehrere Datenbankpartitionen verwenden.
- Die Primärdatenbank und die Bereitschaftsdatenbank müssen dieselbe Betriebssystemversion und dieselbe Version von DB2 UDB verwenden, außer während einem schrittweisen Upgrade.
- Das DB2 UDB-Release für die Primärdatenbank und die Bereitschaftsdatenbank muss dieselbe Bitgröße (32 oder 64 Bit) verwenden.
- Lesevorgänge in der Bereitschaftsdatenbank werden nicht unterstützt. Clients können keine Verbindung zur Bereitschaftsdatenbank herstellen.
- Die Protokollarchivierung kann nur von der aktuellen Primärdatenbank ausgeführt werden.
- Sicherungsoperationen in der Bereitschaftsdatenbank werden nicht unterstützt.
- Nicht protokollierte Operationen, wie Änderungen an Datenbankkonfigurationsparametern und der Datei des Wiederherstellungsprotokolls, werden in der Bereitschaftsdatenbank nicht repliziert.
- Ladeoperationen mit der Option COPY NO werden nicht unterstützt.
- Die Verwendung von Data Links wird nicht unterstützt.

Zugehörige Konzepte:

- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201
- „Systemvoraussetzungen für HADR" auf Seite 202
- „Replizierte Operationen für HADR" auf Seite 216
- „Nicht replizierte Operationen für HADR" auf Seite 217

Datenbankkonfiguration für HADR

Zum Erzielen einer optimalen Leistung mit HADR (High Availability Disaster Recovery) stellen Sie sicher, dass Ihre Datenbankkonfiguration den folgenden Anforderungen entspricht.

Empfehlung: Die Datenbankkonfigurationsparameter und Datenbankmanagerkonfigurationsparameter sollten auf allen Systemen, auf denen sich die Primärdatenbank und die Bereitschaftsdatenbank befinden, so genau wie möglich übereinstimmen. Wenn die Konfigurationsparameter für die Bereitschaftsdatenbank nicht richtig gesetzt wurden, können folgende Probleme auftreten:

- Die Bereitschaftsdatenbank gibt Fehlernachrichten zurück, während sie die von der Primärdatenbank übertragenen Protokolldateien wiederholt.
- Nach einer Übernahmeoperation kann die neue Primärdatenbank die Auslastung nicht bewältigen, wodurch es zu Leistungsproblemen oder zur Ausgabe von Fehlernachrichten kommen kann, die die Anwendungen früher während ihrer Verbindung zur ursprünglichen Primärdatenbank nicht erhielten.

Änderungen an den Konfigurationsparametern in der Primärdatenbank werden nicht automatisch an die Bereitschaftsdatenbank weitergegeben und 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.

Anmerkung: Der Datenbankkonfigurationsparameter LOGFILSIZ bildet eine Ausnahme. Auch wenn dieser Parameter nicht in der Bereitschaftsdatenbank repliziert wird, ignoriert die Bereitschaftsdatenbank die lokale Konfiguration für LOGFILSIZ und erstellt lokale Protokolldateien, die von der Größe her den Protokolldateien in der Primärdatenbank entsprechen, um identische Protokolldateien für beide Datenbanken zu garantieren.

Größe des Protokollempfangspuffers in der Bereitschaftsdatenbank:

Standardmäßig entspricht die Größe des Protokollempfangspuffers in der Bereitschaftsdatenbank dem doppelten Wert des in der Primärdatenbank angegebenen Konfigurationsparameters LOGBUFSZ. Diese Größe kann in manchen Fällen nicht ausreichen. Wenn beispielsweise der HADR-Modus ASYNC verwendet wird und sich die Primärdatenbank und die Bereitschaftsdatenbank im Peerstatus befinden, kann die Kapazität des Protokollempfangspuffers in der Bereitschaftsdatenbank beim Auftreten einer hohen Transaktionslast erschöpfen und die Protokollübertragungsoperation von der Primärdatenbank blockieren. Um diese temporären hohen Systemauslastungen zu bewältigen, können Sie die Größe des Protokollempfangspuffers in der Bereitschaftsdatenbank erhöhen, indem Sie die Registrierdatenbankvariable DB2_HADR_BUF_SIZE modifizieren.

Ladeoperationen und HADR:

Wenn eine Ladeoperation in der Primärdatenbank mit der Option COPY YES ausgeführt wird, wird der Befehl in der Primärdatenbank ausgeführt, und die Daten in der Bereitschaftsdatenbank werden repliziert, solange über den mit dem Befehl LOAD angegebenen Pfad oder der angegebenen Einheit auf die Kopie zugegriffen werden kann. Kann die Bereitschaftsdatenbank nicht auf diese Daten zugreifen, wird der Tabellenbereich, in dem die Tabelle gespeichert ist, in der Bereitschaftsdatenbank als fehlerhaft markiert. Die Bereitschaftsdatenbank wird zukünftige Protokollsätze, die zu diesem Tabellenbereich gehören, überspringen.

Wenn eine Ladeoperation in der Primärdatenbank mit der Option NONRECOVERABLE ausgeführt wird, wird der Befehl in der Primärdatenbank ausgeführt, und die Tabelle wird in der Bereitschaftsdatenbank als fehlerhaft markiert. Die Bereitschaftsdatenbank wird zukünftige Protokollsätze überspringen, die zu diesem Tabellenbereich gehören. Sie können den Befehl LOAD wahlweise mit den Optionen COPY YES und REPLACE angeben, um die Tabelle zurückzuholen, oder Sie können die Tabelle löschen, um den Bereich wiederherzustellen.

Da HADR das Ausführen einer Ladeoperation mit der Option COPY NO nicht unterstützt, wird der Befehl automatisch in eine Ladeoperation mit der Option NONRECOVERABLE konvertiert. Um die Konvertierung einer Ladeoperation mit der Option COPY NO in eine Ladeoperation mit der Option COPY YES zu aktivieren, setzen Sie in der Primärdatenbank die Registrierdatenbankvariable DB2_LOAD_COPY_NO_OVERRIDE. Diese Registrierdatenbankvariable wird von

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

Wenn Tabellendaten von einer Ladeoperation mit der Option COPY YES repliziert werden, werden die Indizes wie folgt repliziert:

- Wenn der Indexierungsmodus auf REBUILD gesetzt ist und das Tabellenattribut auf LOG INDEX BUILD, oder wenn das Tabellenattribut auf DEFAULT gesetzt ist und der Datenbankkonfigurationsparameter LOGINDEXBUILD auf ON, schließt die Primärdatenbank das erneut erstellte Indexobjekt in die Kopierdatei ein, um der Bereitschaftsdatenbank das Replizieren des Indexobjekts zu ermöglichen. Wenn das Indexobjekt in der Bereitschaftsdatenbank vor der Ladeoperation als fehlerhaft markiert wurde, wird es nach der Ladeoperation durch die erneute Indexerstellung wieder verwendbar.
- Wenn der Indexierungsmodus auf INCREMENTAL gesetzt ist und das Tabellenattribut auf LOG INDEX BUILD, oder wenn das Tabellenattribut auf NULL gesetzt ist und der Datenbankkonfigurationsparameter LOGINDEXBUILD in der Primärdatenbank auf ON, wird das Indexobjekt in der Bereitschaftsdatenbank nur dann aktualisiert, wenn es nicht vor der Ladeoperation als fehlerhaft markiert wurde. Andernfalls wird der Index in der Bereitschaftsdatenbank als fehlerhaft markiert.

HADR-Konfigurationsparameter:

Zur Unterstützung von HADR stehen mehrere neue Datenbankkonfigurationsparameter zur Verfügung. Das Setzen dieser Parameter hat keine Auswirkungen auf die Rolle einer Datenbank. Zum Ändern der Rolle einer Datenbank müssen Sie die Befehle START HADR oder STOP HADR absetzen.

Die an den HADR-Konfigurationsparametern vorgenommenen Änderungen treten erst dann in Kraft, wenn die Datenbank heruntergefahren und erneut gestartet wurde. In einer Umgebung mit partitionierten Datenbanken sind die HADR-Konfigurationsparameter sichtbar und können geändert werden, sie werden jedoch ignoriert.

Der lokale Hostname der Primärdatenbank muss mit dem fernen Hostnamen der Bereitschaftsdatenbank übereinstimmen, und der lokale Hostname der Bereitschaftsdatenbank muss mit dem fernen Hostnamen der Primärdatenbank übereinstimmen. Verwenden Sie die Konfigurationsparameter HADR_LOCAL_HOST und HADR_REMOTE_HOST, um die lokalen und fernen Hosts für die einzelnen Datenbanken festzulegen. Die Konsistenz der Konfiguration der lokalen und fernen Hostnamen wird beim Herstellen einer Verbindung überprüft, um sicherzustellen, dass der angegebene ferne Host der erwartete Knoten ist.

Der Synchronisationsmodus (HADR_SYNCMODE) und das Zeitlimitintervall (HADR_TIMEOUT) muss für die Primärdatenbank und die Bereitschaftsdatenbank identisch sein. Die Konsistenz dieser Konfigurationsparameter wird überprüft, wenn ein HADR-Paar eine Verbindung herstellt.

Für die Kommunikation zwischen der Primärdatenbank und der Bereitschaftsdatenbank werden TCP-Verbindungen verwendet. Eine Primärdatenbank, die nicht mit einer Bereitschaftsdatenbank verbunden ist, da sie entweder gerade gestartet wird oder die Verbindung unterbrochen wurde, ist an ihrem lokalen Port empfangsbereit für neue Verbindungen. Eine Bereitschaftsdatenbank, die nicht mit einer Primärdatenbank verbunden ist, sendet weiter Verbindungsanforderungen an ihren fernen Host.

Auch wenn der lokale Host und die lokalen Serviceparameter (HADR_LOCAL-_HOST, HADR_LOCAL_SVC) nur in der Primärdatenbank verwendet werden, sollten Sie sie trotzdem auch in der Bereitschaftsdatenbank setzen, um sicherzustellen, dass sie bereit sind, falls die Bereitschaftsdatenbank die Rolle der Primärdatenbank übernehmen muss.

Es folgt eine Beispielkonfiguration für die Primärdatenbank und die Bereitschaftsdatenbank:

Für die 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
```

Für die 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
```

Zugehörige Konzepte:

- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201

Zugehörige Referenzen:

- „hadr_db_role - Rolle der HADR-Datenbank (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „hadr_local_host - Name des lokalen Hosts für HADR (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „hadr_local_svc - Lokaler HADR-Servicename (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „hadr_remote_host - Name des fernen Hosts für HADR (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „hadr_remote_inst - HADR-Exemplarname des fernen Servers (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „hadr_remote_svc - Ferner HADR-Servicename (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*

- „hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „hadr_timeout - HADR-Zeitlimitwert (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „vendoropt - Lieferantenoptionen (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*

Status von Bereitschaftsdatenbanken bei HADR

Wenn Sie die HADR-Funktion verwenden und die Bereitschaftsdatenbank gestartet wird, erhält sie den Status *Lokales Catch-up* und versucht, die Protokolldateien in ihrem lokalen Protokollpfad zu lesen. Wenn die Datenbank keine Protokolldatei in ihrem lokalen Protokollpfad findet und eine Protokollarchivierungsmethode angegeben wurde, wird die Protokolldatei unter Verwendung der angegebenen Methode abgerufen. Nachdem die Protokolldateien gelesen wurden, werden sie in der Bereitschaftsdatenbank wiederholt. Während dieser Zeit ist keine Verbindung zur Primärdatenbank erforderlich. Wenn keine Verbindung vorhanden ist, versucht die Bereitschaftsdatenbank jedoch, eine Verbindung zur Primärdatenbank herzustellen. Wird das Ende der lokalen Protokolldateien erreicht, wechselt die Bereitschaftsdatenbank in den Status *Fernes Catch-up anstehend*.

Die Bereitschaftsdatenbank behält den Status *Fernes Catch-up anstehend*, bis eine Verbindung zur Primärdatenbank aufgebaut ist, danach wechselt sie in den Status *Fernes Catch-up*. Währenddessen liest die Primärdatenbank Protokolldaten aus ihrem Protokollpfad oder unter Verwendung einer Protokollarchivierungsmethode und sendet diese Protokolldaten an die Bereitschaftsdatenbank. Die Bereitschaftsdatenbank empfängt und wiederholt die Protokolldaten. Nachdem alle Protokolldateien auf Platte von der Bereitschaftsdatenbank wiederholt wurden, erhalten die Primär- und Bereitschaftssysteme den Peerstatus.

Im Peerstatus werden Protokollseiten jedes Mal an die Bereitschaftsdatenbank übertragen, wenn die Primärdatenbank eine Protokollseite auf Platte schreibt. Die Protokollseiten werden in die lokalen Protokolldateien in der Bereitschaftsdatenbank geschrieben, um sicherzustellen, dass die Protokolldateifolgen der Primärdatenbank und die Bereitschaftsdatenbank identisch sind. Die Protokollseiten können anschließend in der Bereitschaftsdatenbank wiederholt werden.

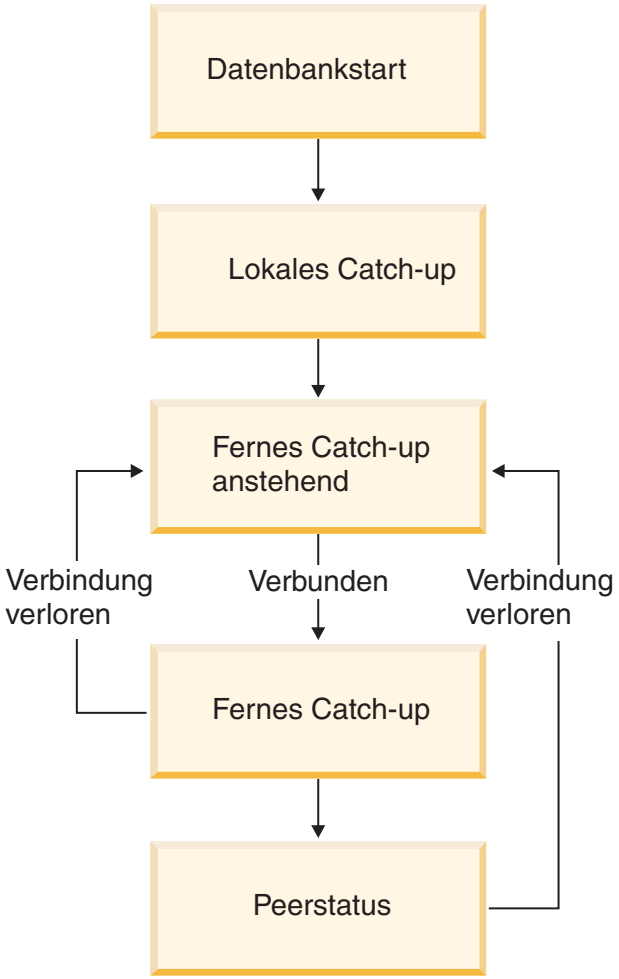


Abbildung 16. Status der Bereitschaftsdatenbank

Sie können den Status der Bereitschaftsdatenbank anzeigen, indem Sie den Befehl GET SNAPSHOT mit der Option FOR DATABASE ON absetzen. Den Status einer Bereitschaftsdatenbank MUSIC beispielsweise können Sie mit dem folgenden Befehl anzeigen:

```
get snapshot for database on music
```

Wird der Befehl für die Bereitschaftsdatenbank abgesetzt, wird im Feld `Status` einer der Bereitschaftsdatenbankstatus zurückgegeben. Wird die Abfrage für eine Primärdatenbank ausgeführt, die mit einer Bereitschaftsdatenbank verbunden ist, wird der Status der Bereitschaftsdatenbank zurückgegeben. Ist die Primärdatenbank nicht mit einer Bereitschaftsdatenbank verbunden, wird der Status `disconnected` zurückgegeben.

Es folgt eine Beispielausgabe für den Abschnitt `HADR-Status`, der vom Befehl GET SNAPSHOT zurückgegeben wird:

```
HADR-Status

  Rolle                       = Primär
  Status                      = Peer
  Synchronisationsmodus       = Synchron
  Verbindungsstatus           = Verbunden, 11/03/2002 12:23:09.35092
  Fehlende Überwachungssignale = 0
  Lokaler Host                = host1.ibm.com
```

```
Lokaler Service             = hadr_service
Ferner Host                 = host2.ibm.com
Ferner Service              = hadr_service
Fernes Exemplar             = dbinst2
Zeitlimit (Sekunden)        = 120
Position des Primärprotokolls (Datei, Seite, Protokollfolgenummer)
                            = S0001234.LOG, 12, 0000000000BB800C
Position des Bereitschaftsprotokolls (Datei, Seite, Protokollfolgenummer)
                            = S0001234.LOG, 12, 0000000000BB800C
Von benutzten Seiten belegter Protokollspeicherbereich (Byte)
                            = 8723
```

Anmerkungen:

1. Geht die Verbindung zwischen der Primärdatenbank und der Bereitschaftsdatenbank verloren, während sich die Bereitschaftsdatenbank im Status *Fernes Catch-up* befindet, erhält diese den Status *Fernes Catch-up anstehend*.
2. Da die Bereitschaftsdatenbank die empfangenen Protokolldateien in ihren lokalen Protokollpfad schreibt, dürfen Sie für die Primärdatenbank und die Bereitschaftsdatenbank kein gemeinsam genutztes lokales Dateisystem oder Netzdateisystem als Protokollpfad verwenden. Wenn DB2® einen gemeinsam benutzten Protokollpfad entdeckt, wird eine Fehlernachricht zurückgegeben.
3. Um den Catch-up-Prozess zu beschleunigen, können Sie eine gemeinsam genutzte Protokollarchiveinheit verwenden. Wenn die gemeinsam genutzte Einheit jedoch eine serielle Einheit ist (z. B. ein Bandlaufwerk), kann durch die gemischten Lese- und Schreibvorgänge die Leistung verringert werden.
4. Sie können die Protokolldateien der Primärdatenbank manuell in den Protokollpfad der Bereitschaftsdatenbank kopieren, um sie für lokales Catch-up zur Verfügung zu stellen. Der Kopiervorgang muss vor dem Starten der Bereitschaftsdatenbank ausgeführt werden, da die Bereitschaftsdatenbank beim Erreichen des Endes der lokalen Protokolldateien in den Status *Fernes Catch-up anstehend* wechselt und nicht mehr versuchen wird, auf die Protokolldateien zuzugreifen. Wenn die Bereitschaftsdatenbank in den Status *Fernes Catch-up* wechselt, überschneidet sich außerdem das Kopieren von Protokolldateien in den Protokollpfad mit dem Schreiben von lokalen Protokolldateien der Bereitschaftsdatenbank. Wenn nach dem Wechseln der Bereitschaftsdatenbank in den Status *Fernes Catch-up anstehend* weitere Protokolldateien verfügbar werden, können Sie die Bereitschaftsdatenbank herunterfahren und erneut starten, um sicherzustellen, dass sie wieder den Status *Lokales Catch-up* erhält.

Zugehörige Konzepte:

- „Verwalten von Protokolldateien" auf Seite 51
- „Verwalten von Protokolldateien durch Protokollarchivierung" auf Seite 54

Zugehörige Referenzen:

- Anhang G, „Benutzerexit zur Datenbankwiederherstellung", auf Seite 369
- „GET SNAPSHOT Command" im Handbuch *Command Reference*
- „Konfigurationsparameter für die Datenbankprotokollierung" auf Seite 42
- „logarchmeth1 - Primäre Protokollarchivierungsmethode (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*
- „logarchmeth2 - Sekundäre Protokollarchivierungsmethode (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*

Synchronisationsmodi für HADR

Für HADR (High Availability Disaster Recovery) können Sie einen von drei Synchronisationsmodi auswählen, um Ihre bevorzugte Schutzebene vor potenziellen Datenverlusten anzugeben. Der Synchronisationsmodus gibt an, wie lange Schreibvorgänge zwischen der Primärdatenbank und der Bereitschaftsdatenbank verwaltet werden. Diese Modi können nur dann angewendet werden, wenn sich die Datenbanken im Peerstatus befinden.

Verwenden Sie den Konfigurationsparameter HADR_SYNCMODE, um den Synchronisationsmodus festzulegen. Gültige Werte sind:

SYNC (synchron)
Dieser Modus bietet den größten Schutz vor Datenverlust, benötigt jedoch von allen drei 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 wiederholen kann, kann sie bei ihrem nächsten Start die Protokollsätze aus ihren lokalen Protokolldateien abrufen und wiederholen. Wenn die Primärdatenbank fehlschlägt, 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 in der ursprünglichen Primärdatenbank nie als festgeschrieben dokumentiert wurden. Dies kann auftreten, wenn die Primärdatenbank fehlschlägt, bevor sie eine Empfangsbestätigungsnachricht von der Bereitschaftsdatenbank verarbeiten konnte. Clientanwendungen sollten gegebenenfalls die Datenbank abfragen, um zu ermitteln, ob solche Transaktionen vorhanden sind.

Wenn die Primärdatenbank ihre Verbindung zur Bereitschaftsdatenbank verliert, verlieren die Datenbanken auch ihren Peerstatus, und Transaktionen werden nicht mehr zurückgehalten, um auf eine Bestätigung von der Bereitschaftsdatenbank zu warten. Wird die Funktionsübernahme beim Aufheben der Verbindung zu den Datenbanken ausgeführt, ist nicht gewährleistet, dass alle in der Primärdatenbank festgeschriebenen Transaktionen auch in der Bereitschaftsdatenbank angezeigt werden.

Schlägt die Primärdatenbank fehl, während sich die Datenbanken im Peerstatus 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 fun-

giert) muss lediglich die neuen Protokollsätze wiederholen, 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, wird eine Fehlernachricht ausgegeben, wenn Sie für die ursprüngliche Primärdatenbank den Befehl START HADR mit der Option AS STANDBY absetzen. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar erneut erfolgreich hinzugefügt wurde, können Sie die Datenbank zurücksetzen, indem Sie den Befehl TAKEOVER HADR ohne die Option BY FORCE absetzen. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar nicht wieder hinzugefügt werden kann, können Sie sie reinitialisieren, indem Sie ein Sicherungsimage 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.

Schlägt die Primärdatenbank fehl, 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 vollständigen Wiederherstellungsoperation 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 Wiederherstellungsoperation ausgeführt wird. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar erneut erfolgreich hinzugefügt wurde, können Sie die Datenbank zurücksetzen, indem Sie den Befehl TAKEOVER HADR ohne die Option BY FORCE absetzen. Wenn die ursprüngliche Primärdatenbank dem HADR-Paar nicht wieder hinzugefügt werden kann, können Sie sie reinitialisieren, indem Sie ein Sicherungsimage der neuen Primärdatenbank wiederherstellen.

ASYNC (asynchron)

In diesem Modus ist das Risiko eines Transaktionsverlusts bei einem Ausfall des primären Systems am größten. Dieser Modus hat von den drei Modi jedoch die kürzeste Transaktionsantwortzeit.

In diesem Modus wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokolle in Protokolldateien in der Primärdatenbank 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 das Bereitschaftssystem weitergeleitet werden.

Bei einem Ausfall der Hostmaschine der Primärdatenbank, des Netzes oder der Bereitschaftsdatenbank können Protokolldateien, die zu diesem Zeitpunkt gerade weitergeleitet werden, verloren gehen. Steht die Primärdatenbank zur Verfügung, können die fehlenden Protokolldateien erneut an die Bereitschaftsdatenbank übertragen werden, wenn die Verbindung zwischen den beiden Datenbanken wiederhergestellt wird. Wird jedoch eine Funktionsübernahme erforderlich, während Protokolldateien fehlen, erreichen weder die Protokolldateien noch die zugeordneten Transaktionen die Bereitschaftsdatenbank. Fehlende Protokolldateien und ein Ausfall der Primärdatenbank sind die Ursachen für einen permanenten Transaktionsverlust.

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.

Schlägt die Primärdatenbank fehl, 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 vollständigen Wiederherstellungsoperation 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 Wiederherstellungsoperation 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 die Option BY FORCE absetzen. Wenn die ursprüngliche Primär-

datenbank dem HADR-Paar nicht wieder hinzugefügt werden kann, können Sie sie reinitialisieren, indem Sie ein Sicherungsimage der neuen Primärdatenbank wiederherstellen.

Zugehörige Konzepte:

- „Status von Bereitschaftsdatenbanken bei HADR" auf Seite 208

Zugehörige Tasks:

- „Reintegrieren einer Datenbank nach einer Übernahmeoperation" auf Seite 234

Zugehörige Referenzen:

- „hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*

Automatische Clientweiterleitung und HADR

Die automatische Clientweiterleitung kann mit HADR verwendet werden, um es Clientanwendungen zu ermöglichen, die Serververbindung nach einer Unterbrechung wiederherzustellen und mit nur minimalen Unterbrechungen weiterzuarbeiten. Eine Weiterleitung ist nur möglich, wenn auf dem Server eine alternative Datenbankposition angegeben wurde. Die automatische Clientweiterleitung wird nur mit TCP/IP-Protokoll unterstützt.

Sie können die automatische Clientweiterleitung mit HADR verwenden, um es Clientanwendungen nach einer Übernahmeoperation zu ermöglichen, eine Verbindung zur neuen Primärdatenbank herzustellen. Wenn die automatische Clientweiterleitung nicht aktiviert ist, erhalten Clientanwendungen die Fehlernachricht SQL30081, und es werden keine weiteren Versuche unternommen, eine Verbindung zum Server herzustellen. Im folgenden Beispiel wird erläutert, wie Sie den Befehl UPDATE ALTERNATE SERVER FOR DATABASE zum Konfigurieren der automatischen Clientweiterleitung für HADR verwenden.

Beispiel

Ihr System ist wie folgt konfiguriert:

- 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 Portnummer 456.

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 alternativen Serverinformationen 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.

Anmerkungen:

1. Die alternative Hostposition ist in der Systemdatenbankverzeichnisdatei auf dem Server gespeichert.
2. Zum Aktivieren der automatischen Clientweiterleitung müssen Sie den Befehl UPDATE ALTERNATE SERVER FOR DATABASE verwenden. Die automatische Clientweiterleitung verwendet die Datenbankkonfigurationsparameter HADR_REMOTE_HOST und HADR_REMOTE_SVC nicht.

Zugehörige Konzepte:

- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201
- „Implementierung der automatischen Clientweiterleitung" im Handbuch *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „UPDATE ALTERNATE SERVER FOR DATABASE Command" im Handbuch *Command Reference*
- „db2UpdateAlternateServerForDB - Update Alternate Server for Database" im Handbuch *Administrative API Reference*

Indexprotokollierung und HADR

Berücksichtigen Sie beim Festlegen der Konfigurationsparameter für HADR-Datenbanken die folgenden Empfehlungen.

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. 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 Konfigurations-

parameter 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 eine Sicherung der Datenbank, und wenden Sie diese Sicherung 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.

Zugehörige Referenzen:

- „indexrec - Zeitpunkt für Indexneuerstellung" im Handbuch *Systemverwaltung: Optimierung*
- „ALTER TABLE statement" im Handbuch *SQL Reference, Volume 2*
- „logindexbuild - Erstellte Indexseiten protokollieren (Konfigurationsparameter)" im Handbuch *Systemverwaltung: Optimierung*

Replizierte Operationen für HADR

Wenn Sie HADR verwenden, werden die folgenden Operationen aus der Primärdatenbank in der 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 die Offlinereorganisation nicht in Gruppen zusammengefasst ist, 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.

Zugehörige Konzepte:

- „Tabellenreorganisation" im Handbuch *Systemverwaltung: Optimierung*
- „Indexreorganisation" im Handbuch *Systemverwaltung: Optimierung*
- „Datenbankkonfiguration für HADR" auf Seite 204
- „Nicht replizierte Operationen für HADR" auf Seite 217

Nicht replizierte Operationen für HADR

HADR verwendet Datenbankprotokolle, um Daten in der Bereitschaftsdatenbank zu replizieren. Nicht protokollierte Operationen sind in der Primärdatenbank zwar zulässig, sie werden aber nicht in der Bereitschaftsdatenbank repliziert. Bei nicht replizierten Operationen handelt es sich unter anderem um Folgende:

- Tabellen, die mit der Option NOT LOGGED INITIALLY erstellt wurden.
- BLOBs und CLOBs werden nicht repliziert, der Speicherbereich für die BLOBs und CLOBs wird jedoch in der Bereitschaftsdatenbank zugeordnet.
- Data Links werden in HADR-Datenbanken nicht unterstützt. Wenn der Befehl START HADR oder ACTIVATE DATABASE abgesetzt wird oder wenn eine erste Clientverbindung eine Datenbank aktiviert, die eine HADR-Rolle inne hat (Primär oder Bereitschaft), und der Datenbankmanagerkonfigurationsparameter

DATALINKS ist auf YES gesetzt, schlägt die Operation fehl. Um die HADR-Datenbank zu verwenden, setzen Sie den Konfigurationsparameter DATALINKS auf NO, fahren Sie das Datenbankexemplar herunter, und starten Sie es erneut.

Anmerkung: Vorhandene DATALINK-Spalten sind nicht betroffen, wenn eine Standarddatenbank in eine Primärdatenbank oder Bereitschaftsdatenbank konvertiert wird. Auch wenn der Konfigurationsparameter DATALINKS auf NO gesetzt ist, können Sie immer noch neue DATALINK-Spalten in einer HADR-Datenbank erstellen. Sie können DATALINK-Spalten jedoch nicht einfügen, aktualisieren oder auswählen.

- Aktualisierungen an der Datenbankkonfiguration, die mit den Befehlen UPDATE DATABASE CONFIGURATION und UPDATE DATABASE MANAGER CONFIGURATION ausgeführt werden, werden nicht repliziert.
- Die Datei des Wiederherstellungsprotokolls und daran vorgenommene Änderungen werden nicht automatisch von der Primärdatenbank an die Bereitschaftsdatenbank übertragen.

  Sie können eine Anfangskopie der Verlaufsdatei (die Sie aus dem Sicherungsimage der Primärdatenbank abrufen) in die Bereitschaftsdatenbank kopieren, indem Sie den Befehl RESTORE DATABASE mit der Option REPLACE HISTORY FILE absetzen:

  ```
  RESTORE DB KELLY REPLACE HISTORY FILE
  ```

  Nachdem HADR initialisiert wurde und in der Primärdatenbank Sicherungsaktivitäten ausgeführt werden können, wird die Verlaufsdatei in der Bereitschaftsdatenbank bald nicht mehr auf dem neuesten Stand sein. Wenn in der Primärdatenbank Sicherungsoperationen ausgeführt werden, können Sie den folgenden Befehl absetzen, um die Verlaufsdatei in der Bereitschaftsdatenbank zu aktualisieren:

  ```
  RESTORE DB KELLY HISTORY FILE
  ```

  Wird eine Übernahmeoperation ausgeführt und verfügt die Bereitschaftsdatenbank über eine aktuelle Verlaufsdatei, generieren Sicherungs- und Wiederherstellungsoperationen in der neuen Primärdatenbank neue Sätze in der Verlaufsdatei und fügen sich nahtlos an die in der ursprünglichen Primärdatenbank generierten Sätze an. Wenn die Verlaufsdatei nicht auf dem neuesten Stand ist oder Einträge fehlen, ist das Ausführen einer automatischen Teilwiederherstellung eventuell nicht möglich und es ist stattdessen eine manuelle Teilwiederherstellungsoperation erforderlich.

Zugehörige Konzepte:

- „Tabellenreorganisation" im Handbuch *Systemverwaltung: Optimierung*
- „Indexreorganisation" im Handbuch *Systemverwaltung: Optimierung*
- „Datei des Wiederherstellungsprotokolls" auf Seite 62
- „Datenbankkonfiguration für HADR" auf Seite 204
- „Replizierte Operationen für HADR" auf Seite 216

Cluster-Manager und HADR

Sie können HADR mit einem Cluster-Manager verwenden, um die Verfügbarkeit des DBMS (Database Management System - Datenbankverwaltungssystem) zu erhöhen. Sie können HADR dazu auf zwei Weisen konfigurieren:

- Konfigurieren Sie ein HADR-Paar, bei dem die Primärdatenbank und die Bereitschaftsdatenbank von demselben Cluster-Manager bedient 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 Wiederherstellung nach einem Systemabsturz oder andere Wiederherstellungsmethoden 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 Sie ein HADR-Paar, bei dem die Primärdatenbank und die Bereitschaftsdatenbank nicht von demselben Cluster-Manager bedient 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 Wiederherstellung 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.

Zugehörige Konzepte:

- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201
- „Automatische Clientweiterleitung und HADR" auf Seite 214

Zugehörige Tasks:

- „Tauschen von Datenbankrollen bei HADR" auf Seite 231

Initialisieren von HADR

Verwenden Sie die folgende Prozedur, um die Primärdatenbank und die Bereitschaftsdatenbank für HADR (High Availability Disaster Recovery) zu konfigurieren und zu initialisieren.

Prozedur:

HADR kann über den Befehlszeilenprozessor (CLP), den Assistenten zum Konfigurieren von HADR in der Steuerzentrale oder durch Aufrufen der entsprechenden Anwendungsprogrammierschnittstellen (APIs) initialisiert werden.

Gehen Sie wie folgt vor, um auf Ihrem System HADR mit dem 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. Der Hostname darf nur mit einer einzigen IP-Adresse übereinstimmen.

   Anmerkung: Die Exemplarnamen für die Primärdatenbank und Bereitschaftsdatenbanken müssen nicht identisch sein.
2. Erstellen Sie die Bereitschaftsdatenbank, indem Sie ein Sicherungsimage 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. In diesem Fall 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
   ```

   Anmerkungen:

   a. Die Datenbanknamen für die Primärdatenbank und die Bereitschaftsdatenbank müssen identisch sein.

   b. Es wird empfohlen, dass Sie nach der Wiederherstellungsoperation oder der Initialisierung der geteilten Spiegeldatenbank nicht den Befehl ROLLFORWARD DATABASE für die Bereitschaftsdatenbank absetzen. Die Ergebnisse einer Operation zur aktualisierenden Wiederherstellung können leicht von dem Ergebnis abweichen, das durch Wiederholung der Protokolle in der Bereitschaftsdatenbank unter Verwendung von HADR erzielt wird. Wenn die Datenbanken nicht identisch sind, schlägt das Absetzen des Befehls START HADR mit der Option AS STANDBY fehl.

c. Wenn Sie den Befehl RESTORE DATABASE verwenden, wird empfohlen, die Option REPLACE HISTORY FILE zu verwenden.
d. Beim Konfigurieren der Bereitschaftsdatenbank sollten Sie die folgenden Optionen des Befehls RESTORE DATABASE vermeiden: TABLESPACE, INTO, REDIRECT und WITHOUT ROLLING FORWARD.
e. 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: Exemplarname, Protokollpfad und Datenbankpfad. Sie dürfen jedoch nicht den Namen der Datenbank oder die Pfade der Tabellenbereichsbehälter ändern.

3. Setzen Sie die HADR-Konfigurationsparameter in der Primärdatenbank und der Bereitschaftsdatenbank.

   Anmerkung: Es ist sehr wichtig, dass Sie die folgenden Konfigurationsparameter im Anschluss an die Erstellung der Bereitschaftsdatenbank setzen:
   - HADR_LOCAL_HOST
   - HADR_LOCAL_SVC
   - HADR_REMOTE_HOST
   - HADR_REMOTE_SVC
   - HADR_REMOTE_INST

   Wenn diese Parameter vor dem Erstellen der Bereitschaftsdatenbank gesetzt werden, geben die Einstellungen in der Bereitschaftsdatenbank die Einstellungen der Primärdatenbank wieder.

4. Stellen Sie eine Verbindung zum Bereitschaftsexemplar 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 zuerst die Primärdatenbank starten, schlägt diese Startprozedur fehl, wenn die Bereitschaftsdatenbank nicht innerhalb des vom Datenbankkonfigurationsparameter HADR_TIMEOUT angegebenen Zeitlimits gestartet wird.

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

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

6. HADR wird nun für die Primärdatenbank und die Bereitschaftsdatenbank gestartet:

Gehen Sie wie folgt vor, um den Assistenten zum Konfigurieren von HADR-Datenbanken zu öffnen:

1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis Sie die Datenbank finden, für die HADR konfiguriert werden soll.
2. Klicken Sie die Datenbank mit Maustaste 2 an, und klicken Sie im Kontextmenü die Option **HADR (High Availability Disaster Recovery)→Einrichten** an. Der Assistent zum Konfigurieren von HADR-Datenbanken wird geöffnet.

Zusatzinformationen bietet die Kontexthilfefunktion der Steuerzentrale.

Zugehörige Konzepte:

- „Datenbankkonfiguration für HADR" auf Seite 204

Zugehörige Tasks:

- „Verwenden des Wiederherstellungsprogramms" auf Seite 96
- „Verwenden einer geteilten Spiegeldatenbank als Sicherungsimage" auf Seite 195

Zugehörige Referenzen:

- „START HADR" auf Seite 222
- „STOP HADR" auf Seite 227

START HADR

Startet HADR-Operationen für eine Datenbank.

Berechtigung:

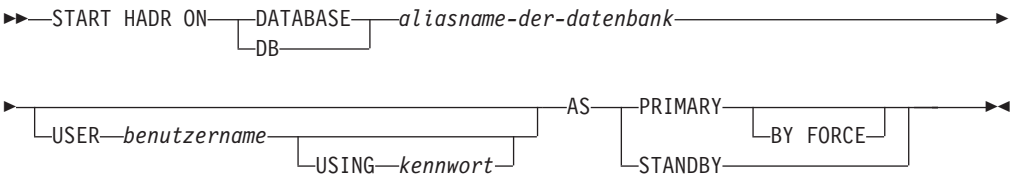
Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Erforderliche Verbindung:

Exemplar. Der Befehl stellt eine Datenbankverbindung her, sofern noch keine vorhanden ist, und schließt die Datenbankverbindung nach Abschluss des Befehls.

Befehlssyntax:



Befehlsparameter:

DATABASE *aliasname-der-datenbank*
Gibt die Datenbank an, in der die HADR-Operationen gestartet werden sollen.

USER *benutzername*
Gibt den Benutzernamen an, unter dem die HADR-Operationen gestartet werden sollen.

USING *kennwort*
Das Kennwort, das verwendet wird, um *benutzername* zu authentifizieren.

AS PRIMARY
Gibt an, dass HADR-Primärdatenbankoperationen in der Datenbank gestartet werden sollen.

BY FORCE
Gibt an, dass die HADR-Primärdatenbank nicht auf das Herstellen einer Verbindung durch die Bereitschaftsdatenbank warten wird. Nach einem Start mit BY FORCE akzeptiert die Primärdatenbank weiter gültige Verbindungen von der Bereitschaftsdatenbank, sobald die Bereitschaftsdatenbank zu einem späteren Zeitpunkt verfügbar wird.

Vorsicht: Verwenden Sie die Option AS PRIMARY BY FORCE des Befehls START HADR mit Vorsicht. Wenn die Bereitschaftsdatenbank in eine Primärdatenbank umgewandelt und mit dem Befehl START HADR mit der Option AS PRIMARY BY FORCE ein Neustart der ursprünglichen Primärdatenbank ausgeführt wurde, arbeiten beide Datenbankkopien unabhängig voneinander als Primärdatenbanken. (Diese Konstellation wird manchmal als *Split Brain* oder als *Dual Primary* bezeichnet.) In diesem Fall kann jede Primärdatenbank Verbindungen annehmen und Transaktionen ausführen, und es werden keine Aktualisierungen der jeweils anderen Datenbank empfangen und wiederholt. Dies führt dazu, dass die beiden Datenbankkopien inkonsistent werden.

AS STANDBY
Gibt an, dass HADR-Bereitschaftsdatenbankoperationen in der Datenbank gestartet werden sollen. Die Bereitschaftsdatenbank versucht so lange eine Verbindung zur HADR-Primärdatenbank herzustellen, bis die Verbindung erfolgreich aufgebaut wurde oder bis der Verbindungsversuch von der Primärdatenbank explizit zurückgewiesen wurde. (Die Verbindung kann von der Primärdatenbank zurückgewiesen werden, wenn ein HADR-Konfigurationsparameter falsch gesetzt wurde oder wenn die Datenbankkopien nicht konsistent sind. In beiden Fällen sollte der Verbindungsversuch nicht wiederholt werden.)

Hinweise:

Die folgende Tabelle zeigt das Verhalten der Datenbank bei verschiedenen Bedingungen:

Datenbankstatus	Verhalten bei Befehl START HADR mit Option AS PRIMARY	Verhalten bei Befehl START HADR mit Option AS STANDBY
Inaktive Standarddatenbank	Aktiviert als HADR-Primärdatenbank.	Die Datenbank startet als Bereitschaftsdatenbank, wenn sie sich im Modus *Aktualisierende Wiederherstellung anstehend* (als Ergebnis einer Wiederherstellung oder geteilten Spiegelung) oder im Modus *Aktualisierende Wiederherstellung wird ausgeführt* befindet. Andernfalls wird ein Fehler zurückgegeben.
Aktive Standarddatenbank	Datenbank übernimmt Rolle der HADR-Primärdatenbank.	Eine Fehlernachricht wird zurückgegeben.

Datenbankstatus	Verhalten bei Befehl START HADR mit Option AS PRIMARY	Verhalten bei Befehl START HADR mit Option AS STANDBY
Inaktive Primärdatenbank	Aktiviert als HADR-Primärdatenbank.	Integriert die fehlgeschlagene Primärdatenbank nach einer Funktionsübernahme wieder als neue Primärdatenbank in das HADR-Paar. Es gelten einige Einschränkungen.
Aktive Primärdatenbank	Eine Warnung wird ausgegeben.	Eine Fehlernachricht wird zurückgegeben.
Inaktive Bereitschaftsdatenbank	Eine Fehlernachricht wird zurückgegeben.	Startet die Datenbank als Bereitschaftsdatenbank.
Aktive Bereitschaftsdatenbank	Eine Fehlernachricht wird zurückgegeben.	Eine Warnung wird ausgegeben.

db2HADRStart - HADR starten

Startet HADR-Operationen in einer Datenbank.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Erforderliche Verbindung:

Exemplar. Die API stellt eine Datenbankverbindung her, sofern noch keine vorhanden ist, und schließt die Datenbankverbindung nach Beendigung der API.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2HADRStart */
/* ... */
SQL_API_RC SQL_API_FN
  db2HADRStart (
  db2Uint32 versionNumber,
  void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRStartStruct
{
  char                        *piDbAlias;
  char                        *piUserName;
  char                        *piPassword;
  db2Uint32                   iDbRole;
  db2Uint16                   iByForce;
} db2HADRStartStruct;
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2gHADRStart */
/* ... */
SQL_API_RC SQL_API_FN
  db2gHADRStart (
  db2Uint32 versionNumber,
  void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRStartStruct
{
  char                        *piDbAlias;
   db2Uint32 iAliasLen;
  char                        *piUserName;
   db2Uint32 iUserNameLen;
  char                        *piPassword;
   db2Uint32 iPasswordLen;
  db2Uint32                   iDbRole;
  db2Uint16                   iByForce;
} db2gHADRStartStruct;
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pParmStruct*, übergeben wird.

pParmStruct
Eingabe. Ein Zeiger auf die Struktur *db2HADRStartStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

piDbAlias
Eingabe. Ein Zeiger auf den Datenbankaliasnamen.

iAliasLen
Eingabe. Gibt die Länge des Datenbankaliasnamens in Byte an.

piUserName
Eingabe. Ein Zeiger auf den Benutzernamen, unter dem der Befehl ausgeführt wird.

iUserNameLen
Eingabe. Gibt die Länge des Benutzernamens in Byte an.

piPassword
Eingabe. Ein Zeiger auf eine Zeichenfolge, die das Kennwort enthält.

iPasswordLen
Eingabe. Gibt die Länge des Kennworts in Byte an.

iDbRole
Eingabe. Gibt an, ob die Datenbank als HADR-Primärdatenbank oder HADR-Bereitschaftsdatenbank gestartet werden soll. Gültige Werte:

DB2HADR_DB_ROLE_PRIMARY
DB2HADR_DB_ROLE_STANDBY

iByForce
Eingabe. Dieses Argument wird ignoriert, wenn *iDbRole* auf DB2HADR_DB_ROLE_STANDBY gesetzt ist. Gültige Werte:

DB2HADR_NO_FORCE
Gibt an, dass HADR nur dann für die Primärdatenbank gestartet wird, wenn eine Bereitschaftsdatenbank innerhalb eines vorgegebenen Zeitlimits eine Verbindung zu dieser Primärdatenbank herstellt.

DB2HADR_FORCE
Gibt an, dass der Start von HADR erzwungen wird, ohne dass auf das Herstellen einer Verbindung durch eine Bereitschaftsdatenbank gewartet wird.

Zugehörige Referenzen:

- „db2HADRStop - HADR stoppen" auf Seite 229
- „db2HADRTakeover - Als Primärdatenbank übernehmen" auf Seite 239
- „START HADR" auf Seite 222

Stoppen von HADR

Verwenden Sie den Befehl STOP HADR, um HADR-Operationen in der Primärdatenbank oder der Bereitschaftsdatenbank zu stoppen. Sie können HADR wahlweise für eine oder beide Datenbanken stoppen. Wenn Sie im Bereitschaftssystem Wartungsarbeiten ausführen, müssen Sie HADR nur für die Bereitschaftsdatenbank stoppen. Wenn Sie HADR vollständig stoppen wollen, können Sie HADR für beide Datenbanken stoppen.

Einschränkungen:

Sie können den Befehl STOP HADR nur für eine Primärdatenbank oder eine Bereitschaftsdatenbank ausführen. Wenn Sie diesen Befehl für eine Standarddatenbank absetzen, wird ein Fehler zurückgegeben.

Prozedur:

Sie können HADR über den Befehlszeilenprozessor (CLP), das Fenster **HADR verwalten** in der Steuerzentrale oder über die API **db2HADRStop** stoppen.

Wenn Sie HADR-Operationen in der Primärdatenbank oder der Bereitschaftsdatenbank über den CLP stoppen wollen, setzen Sie den Befehl STOP HADR für die Datenbank ab, in der HADR-Operationen gestoppt werden sollen.

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 Wiederherstellung 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 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.

Gehen Sie wie folgt vor, um das Fenster **HADR verwalten** zu öffnen:
1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis Sie die Datenbank finden, für die HADR verwaltet werden soll.
2. Klicken Sie die Datenbank mit Maustaste 2 an, und klicken Sie im Kontextmenü die Option **HADR (High Availability Disaster Recovery)→Verwalten** an. Das Fenster **HADR verwalten** wird geöffnet.

Zusatzinformationen bietet die Kontexthilfefunktion der Steuerzentrale.

Zugehörige Konzepte:

- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201

Zugehörige Referenzen:

- „START HADR" auf Seite 222
- „STOP HADR" auf Seite 227

STOP HADR

Stoppt HADR-Operationen für eine Datenbank.

Berechtigung:

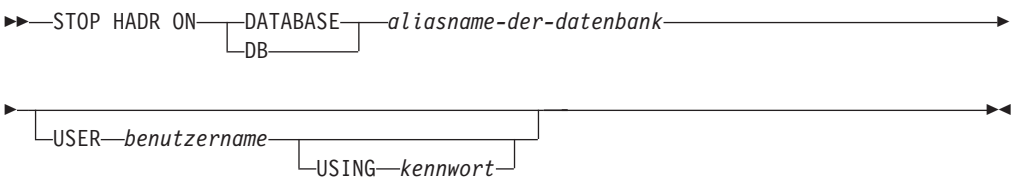
Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Erforderliche Verbindung:

Exemplar. Der Befehl stellt eine Datenbankverbindung her, sofern noch keine vorhanden ist, und schließt die Datenbankverbindung nach Abschluss des Befehls.

Befehlssyntax:



Befehlsparameter:

DATABASE *aliasname-der-datenbank*
Gibt die Datenbank an, in der die HADR-Operationen gestoppt werden sollen.

USER *benutzername*
Gibt den Benutzernamen an, unter dem die HADR-Operationen gestoppt werden sollen.

USING *kennwort*
Das Kennwort, das verwendet wird, um *benutzername* zu authentifizieren.

Hinweise:

Die folgende Tabelle zeigt das Verhalten der Datenbank bei verschiedenen Bedingungen:

Datenbankstatus	Verhalten bei Befehl STOP HADR
Inaktive Standarddatenbank	Eine Fehlernachricht wird zurückgegeben.
Aktive Standarddatenbank	Eine Fehlernachricht wird zurückgegeben.
Inaktive Primärdatenbank	Datenbankrolle wird in Standarddatenbank geändert. Datenbankkonfigurationsparameter *hadr_db_role* wird auf STANDARD aktualisiert. Datenbank bleibt offline. Übernimmt beim nächsten Neustart die Rolle einer Standarddatenbank.
Aktive Primärdatenbank	Stoppt das Übertragen von Protokollen an die HADR-Bereitschaftsdatenbank und fährt alle HADR-EDUs in der HADR-Primärdatenbank herunter. Datenbankrolle wird in Standarddatenbank geändert, und Datenbank bleibt online. Datenbank behält die Rolle einer Standarddatenbank, bis der Befehl START HADR explizit mit der Option AS PRIMARY abgesetzt wird. Der Befehl STOP HADR hat keine Auswirkungen auf offene Sitzungen. Sie können die Befehle STOP HADR und START HADR wiederholt absetzen, während die Datenbank online bleibt. Die Auswirkungen dieser Befehle werden dynamisch in Kraft gesetzt.
Inaktive Bereitschaftsdatenbank	Datenbankrolle wird in Standarddatenbank geändert. Datenbankkonfigurationsparameter *hadr_db_role* wird auf STANDARD aktualisiert. Datenbank bleibt offline. Datenbank wird in Modus *Aktualisierende Wiederherstellung anstehend* versetzt.
Aktive Bereitschaftsdatenbank	Eine Fehlernachricht wird zurückgegeben: Inaktivieren Sie die Bereitschaftsdatenbank vor der versuchten Konvertierung in eine Standarddatenbank.

db2HADRStop - HADR stoppen

Stoppt HADR-Operationen in einer Datenbank.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Erforderliche Verbindung:

Exemplar. Die API stellt eine Datenbankverbindung her, sofern noch keine vorhanden ist, und schließt die Datenbankverbindung nach Beendigung der API.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2HADRStop */
/* ... */
SQL_API_RC SQL_API_FN
  db2HADRStop (
  db2Uint32 versionNumber,
  void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRStopStruct
{
  char                        *piDbAlias;
  char                        *piUserName;
  char                        *piPassword;
} db2HADRStopStruct;
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2gHADRStop */
/* ... */
SQL_API_RC SQL_API_FN
  db2gHADRStop (
  db2Uint32 versionNumber,
  void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRStopStruct
{
  char                        *piDbAlias;
   db2Uint32 iAliasLen;
  char                        *piUserName;
   db2Uint32 iUserNameLen;
  char                        *piPassword;
   db2Uint32 iPasswordLen;
} db2gHADRStopStruct;
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pParmStruct*, übergeben wird.

pParmStruct
Eingabe. Ein Zeiger auf die *db2HADRStopStruct*-Struktur.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

piDbAlias
Eingabe. Ein Zeiger auf den Datenbankaliasnamen.

iAliasLen
Eingabe. Gibt die Länge des Datenbankaliasnamens in Byte an.

piUserName
Eingabe. Ein Zeiger auf den Benutzernamen, unter dem der Befehl ausgeführt wird.

iUserNameLen
Eingabe. Gibt die Länge des Benutzernamens in Byte an.

piPassword
Eingabe. Ein Zeiger auf eine Zeichenfolge, die das Kennwort enthält.

iPasswordLen
Eingabe. Gibt die Länge des Kennworts in Byte an.

Zugehörige Referenzen:

- „db2HADRStart - HADR starten" auf Seite 224
- „db2HADRTakeover - Als Primärdatenbank übernehmen" auf Seite 239
- „STOP HADR" auf Seite 227

Tauschen von Datenbankrollen bei HADR

Wenn Sie HADR verwenden, können Sie mit dem Befehl TAKEOVER HADR die Rollen der Primärdatenbank und der Bereitschaftsdatenbank tauschen.

Warnung: Bevor Sie diese Prozedur ausführen, stellen Sie sicher, dass die Primärdatenbank keine Datenbanktransaktionen mehr verarbeitet.

Einschränkungen:

- 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 dann für einen Rollentausch 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.

Prozedur:

Sie können den Rollentausch der HADR-Datenbanken über den Befehlszeilenprozessor (CLP), das Fenster **HADR verwalten** in der Steuerzentrale oder über die API **db2HADRTakeover** ausführen.

Wenn Sie über den CLP eine Übernahmeoperation für eine Bereitschaftsdatenbank einleiten wollen, setzen Sie den Befehl TAKEOVER HADR ohne die Option BY FORCE für die Bereitschaftsdatenbank ab.

Im folgenden Beispiel wird die Übernahmeoperation für die Bereitschaftsdatenbank LEAFS ausgeführt:

```
TAKEOVER HADR ON DB LEAFS
```

Gehen Sie wie folgt vor, um das Fenster **HADR verwalten** zu öffnen:

1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis Sie die Datenbank finden, für die HADR verwaltet werden soll.
2. Klicken Sie die Datenbank mit Maustaste 2 an, und klicken Sie im Kontextmenü die Option **HADR (High Availability Disaster Recovery)→Verwalten** an. Das Fenster **HADR verwalten** wird geöffnet.

Zusatzinformationen bietet die Kontexthilfefunktion der Steuerzentrale.

Zugehörige Konzepte:

- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201
- „Status von Bereitschaftsdatenbanken bei HADR" auf Seite 208

Zugehörige Referenzen:

- „TAKEOVER HADR" auf Seite 237

Verwenden des Befehls TAKEOVER HADR bei einer Funktionsübernahme

Wenn Sie die aktuelle Bereitschaftsdatenbank als neue Primärdatenbank übernehmen wollen, da die aktuelle Primärdatenbank nicht verfügbar ist, können Sie eine Funktionsübernahme ausführen.

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, aber nicht mehr mit der Bereitschaftsdatenbank kommunizieren kann, könnte eine erzwungene Übernahmeoperation (Befehl TAKEOVER HADR mit Option BY FORCE) in zwei Primärdatenbanken resultieren. 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 ihr Exemplar, sofern möglich. (Wenn das primäre System blockiert, ausgefallen oder aus anderen Gründen nicht verfügbar ist, ist dies eventuell nicht möglich.) Nach einer Übernahmeoperation wird die fehlgeschlagene Datenbank bei einem Neustart nicht automatisch die Rolle der Primärdatenbank übernehmen.
- Die Wahrscheinlichkeit und der Umfang eines Transaktionsverlusts hängt von Ihrer spezifischen Konfiguration und den Umständen ab:
  - Wenn die Primärdatenbank fehlschlägt, während sie sich im Peerstatus 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 fehlschlägt, während sie sich im Peerstatus 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 fehlschlägt, während sie sich im Peerstatus 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 auch verloren gehen, wenn die Primärdatenbank und die Bereitschaftsdatenbank gleichzeitig ausfallen.
  - Wenn die Primärdatenbank fehlschlägt, 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.

Einschränkungen:

- Der Befehl TAKEOVER HADR kann nur für die Bereitschaftsdatenbank abgesetzt werden.
- HADR tauscht keine Daten mit dem DB2®-Fehlermonitor (db2fm) aus, der dazu verwendet werden kann, eine fehlgeschlagene Datenbank automatisch erneut zu starten. Wenn der Fehlermonitor aktiviert ist, sollten Sie sich darüber im Klaren sein, dass der Fehlermonitor für eine vermutlich fehlgeschlagene Primärdatenbank möglicherweise 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 vollständige aktualisierende Wiederherstellung für die frühere Bereitschaftsdatenbank ausführen, 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, befolgen Sie die Prozedur zum Initialisieren von HADR.

Prozedur:

In einem Funktionsübernahmeszenario kann eine Übernahmeoperation über den Befehlszeilenprozessor (CLP), das Fenster **HADR verwalten** in der Steuerzentrale oder die API **db2HADRTakeover** ausgeführt werden.

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 ausgegangen wird, dass die Primärdatenbank offline ist.

Wenn die Primärdatenbank nicht vollständig inaktiviert ist, ist die Bereitschaftsdatenbank weiter mit ihr verbunden und wird an die Primärdatenbank eine Anforderung zum Herunterfahren senden. Die Bereitschaftsdatenbank wird unabhängig davon, ob sie eine Bestätigung erhält, dass die Primärdatenbank heruntergefahren wurde, die Rolle der Primärdatenbank übernehmen.

Gehen Sie wie folgt vor, um das Fenster **HADR verwalten** zu öffnen:

1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis Sie die Datenbank finden, für die HADR verwaltet werden soll.
2. Klicken Sie die Datenbank mit Maustaste 2 an, und klicken Sie im Kontextmenü die Option **HADR (High Availability Disaster Recovery)→Verwalten** an. Das Fenster **HADR verwalten** wird geöffnet.

Zusatzinformationen bietet die Onlinehilfefunktion der Steuerzentrale.

Zugehörige Konzepte:

- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201
- „Synchronisationsmodi für HADR" auf Seite 211
- „Status von Bereitschaftsdatenbanken bei HADR" auf Seite 208

Zugehörige Tasks:

- „Initialisieren von HADR" auf Seite 220
- „Tauschen von Datenbankrollen bei HADR" auf Seite 231

Zugehörige Referenzen:

- „ROLLFORWARD DATABASE" auf Seite 144
- „DEACTIVATE DATABASE Command" im Handbuch *Command Reference*
- „STOP HADR" auf Seite 227
- „TAKEOVER HADR" auf Seite 237

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.

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: Dieser Befehl wird fehlschlagen, wenn die Protokolldatenströme der beiden Datenbankkopien nicht kompatibel sind. Bei Verwendung von HADR darf die ursprüngliche Primärdatenbank vor allem keine protokollierten Operationen anwenden, die nicht in

der ursprünglichen Bereitschaftsdatenbank wiederholt wurden, bevor diese die Rolle als neue Primärdatenbank übernahm. War dies der Fall, können Sie die ursprüngliche Primärdatenbank als Bereitschaftsdatenbank neu starten, indem Sie ein Sicherungsimage der neuen Primärdatenbank wiederherstellen oder eine geteilte Spiegeldatenbank initialisieren.

Nachdem die ursprüngliche Primärdatenbank dem HADR-Paar als Bereitschaftsdatenbank hinzugefügt wurde, können Sie eine Zurücksetzungsoperation ausführen, um die Rollen der Datenbanken wieder zu tauschen, sodass die ursprüngliche Primärdatenbank erneut als aktuelle Primärdatenbank genutzt werden kann. Setzen Sie den folgenden Befehl für die Bereitschaftsdatenbank ab, um diese Zurücksetzungsoperation auszuführen:

```
TAKEOVER HADR ON DB LEAFS
```

Anmerkungen:

1. Wenn sich die HADR-Datenbanken nicht im Peerstatus befinden oder keine Verbindung zwischen ihnen besteht, wird dieser Befehl fehlschlagen.
2. Es wird das Schließen von offenen Primärdatenbanksitzungen erzwungen, und unvollständige Transaktionen werden rückgängig gemacht.
3. Wenn Sie die Rollen der Primärdatenbank und die Bereitschaftsdatenbank tauschen, kann die Option BY FORCE des Befehls TAKEOVER HADR nicht angegeben werden.

Zugehörige Konzepte:

- „Synchronisationsmodi für HADR" auf Seite 211
- „Status von Bereitschaftsdatenbanken bei HADR" auf Seite 208

Zugehörige Tasks:

- „Tauschen von Datenbankrollen bei HADR" auf Seite 231

Zugehörige Referenzen:

- „TAKEOVER HADR" auf Seite 237

Ausführen eines schrittweisen Upgrades in einer HADR-Umgebung

Verwenden Sie diese Prozedur in einer HADR-Umgebung, wenn Sie einen Upgrade der Software (Betriebssystem oder DB2® UDB) oder Hardware ausführen oder Änderungen an Datenbankkonfigurationsparametern vornehmen. Durch diese Prozedur bleibt der Datenbankservice während des Upgradeprozesses verfügbar, wobei nur während des Wechsels von einer Datenbank zur nächsten eine kurze Serviceunterbrechung auftritt. Da HADR eine optimale Leistung bringt, wenn sich die Primärdatenbank und die Bereitschaftsdatenbank auf identischen Systemen befinden, sollten Sie die Änderungen so schnell wie möglich auf beide Systeme anwenden.

Anmerkung: Alle DB2 UDB-FixPaks und Upgrades sollten zuerst in einer Testumgebung implementiert werden, bevor Sie sie auf Ihr Produktionssystem anwenden.

Vorbedingungen:

Das HADR-Paar sollte sich im Peerstatus befinden, bevor der schrittweise Upgrade gestartet wird.

Einschränkungen:

Diese Prozedur kann nicht für einen Upgrade von einer übergeordneten Version von DB2 UDB auf eine neuere Version verwendet werden.

Prozedur:

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. Fahren Sie die Bereitschaftsdatenbank mit dem Befehl DEACTIVATE DATABASE herunter.
   b. Falls erforderlich, fahren Sie das Exemplar in der Bereitschaftsdatenbank herunter.
   c. Nehmen Sie Änderungen an der Software und/oder der Hardware und/oder den DB2-Konfigurationsparametern vor.
   d. Falls erforderlich, starten Sie das Exemplar in der Bereitschaftsdatenbank erneut.
   e. Starten Sie die Bereitschaftsdatenbank mit dem Befehl ACTIVATE DATABASE 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. Setzen Sie den Befehl TAKEOVER HADR für die Bereitschaftsdatenbank ab.
   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 DB2 UDB-FixPak 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 aktualisierten Primärdatenbank. Da die neue Bereitschaftsdatenbank eine ältere Version von DB2 UDB verwendet, kann sie möglicherweise die neuen von der aktualisierten Primärdatenbank generierten Protokollsätze nicht verstehen und wird heruntergefahren. Damit eine Verbindung zur Primärdatenbank hergestellt werden kann, muss für die Bereitschaftsdatenbank ebenfalls ein Upgrade ausgeführt 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 umfassenden Datenbankservice und umfassenden 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.

Zugehörige Konzepte:

- „Automatische Clientweiterleitung und HADR" auf Seite 214
- „Status von Bereitschaftsdatenbanken bei HADR" auf Seite 208

Zugehörige Referenzen:

- „GET SNAPSHOT Command" im Handbuch *Command Reference*
- „ACTIVATE DATABASE Command" im Handbuch *Command Reference*
- „DEACTIVATE DATABASE Command" im Handbuch *Command Reference*
- „TAKEOVER HADR" auf Seite 237

TAKEOVER HADR

Weist eine HADR-Bereitschaftsdatenbank an, im HADR-Paar die Rolle der neuen HADR-Primärdatenbank zu übernehmen.

Berechtigung:

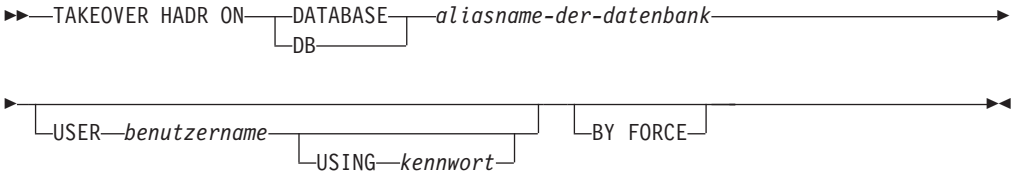
Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Erforderliche Verbindung:

Exemplar. Der Befehl stellt eine Datenbankverbindung her, sofern noch keine vorhanden ist, und schließt die Datenbankverbindung nach Abschluss des Befehls.

Befehlssyntax:



Befehlsparameter:

DATABASE *aliasname-der-datenbank*
Gibt die aktuelle HADR-Bereitschaftsdatenbank an, die die Rolle der neuen HADR-Primärdatenbank übernehmen soll.

USER *benutzername*
Gibt den Benutzernamen an, unter dem die Übernahmeoperation gestartet werden soll.

USING *kennwort*
Das Kennwort, das verwendet wird, um *benutzername* zu authentifizieren.

BY FORCE
Gibt an, dass die Datenbank nicht auf eine Bestätigung wartet, dass die ursprüngliche HADR-Primärdatenbank heruntergefahren wurde. Diese Option ist erforderlich, wenn sich das HADR-Paar nicht im Peerstatus befindet.

Hinweise:

Die folgende Tabelle zeigt das Verhalten des Befehls TAKEOVER HADR bei Verwendung für eine aktive Bereitschaftsdatenbank bei jeder möglichen Kombination von Status und Option. Wird dieser Befehl für eine inaktive Bereitschaftsdatenbank abgesetzt, wird eine Fehlernachricht zurückgegeben.

Status der Bereitschaftsdatenbank	Verwendung der Option BY FORCE	Übernahmeverhalten
Lokales Catch-up oder fernes Catch-up	Nein	Eine Fehlernachricht wird zurückgegeben.
Lokales Catch-up oder fernes Catch-up	Ja	Eine Fehlernachricht wird zurückgegeben.
Peer	Nein	Primärdatenbank und Bereitschaftsdatenbank tauschen die Rollen. Wenn bei der Übernahme kein Fehler auftritt, gehen dabei keine Daten verloren. Treten bei der Übernahme jedoch Fehler auf, kann es zu Datenverlusten kommen, und ist nicht gewährleistet, dass die Rollen der Datenbanken getauscht wurden. Beachten Sie die folgenden Richtlinien zur Behandlung von Fehlern bei einer Übernahme, bei der die Primärdatenbank und die Bereitschaftsschalter ihre Rollen tauschen: 1. Tritt bei einer Übernahmeoperation ein Fehler auf, ist nicht gewährleistet, dass die Rollen der HADR-Systeme getauscht wurden. Stellen Sie wenn möglich sicher, dass beide Datenbanken online sind. Überprüfen Sie die HADR-Rolle der verfügbaren Datenbank(en) mit Snapshot Monitor oder indem Sie den Wert des Datenbankkonfigurationsparameters *hadr_db_role* überprüfen. 2. Wenn die gewünschte neue Primärdatenbank weiter die Rolle der Bereitschaftsdatenbank hat und Sie nach wie vor eine Übernahme ausführen wollen, setzen Sie den Befehl TAKEOVER HADR erneut ab (beachten Sie die folgende Richtlinie in Bezug auf die Option BY FORCE). 3. Es kann der Fall eintreten, dass schließlich beide Datenbanken die Rolle einer Bereitschaftsdatenbank haben. In diesem Fall können Sie die Option BY FORCE jeweils für den Knoten absetzen, der als neue Primärdatenbank verwendet werden soll. Die Option BY FORCE ist in diesem Fall erforderlich, da die beiden Bereitschaftsdatenbanken nicht die übliche HADR-Verbindung zwischen Bereitschaftsdatenbank und Primärdatenbank aufbauen können.
Peer	Ja	Die Bereitschaftsdatenbank benachrichtigt die Primärdatenbank, sich selbst (die Primärdatenbank) herunterzufahren. Die Bereitschaftsdatenbank empfängt keine weiteren Protokolle von der Primärdatenbank, beendet das Wiederholen der bereits empfangenen Protokolle und übernimmt anschließend die Rolle der Primärdatenbank. Die Bereitschaftsdatenbank wartet *nicht* auf eine Bestätigung, dass die Primärdatenbank die Übernahmenachricht empfangen hat oder heruntergefahren wurde.

Status der Bereitschaftsdatenbank	Verwendung der Option BY FORCE	Übernahmeverhalten
Fernes Catch-up anstehend	Nein	Eine Fehlernachricht wird zurückgegeben.
Fernes Catch-up anstehend	Ja	Die Bereitschaftsdatenbank wird zu einer Primärdatenbank.

db2HADRTakeover - Als Primärdatenbank übernehmen

Weist eine Bereitschaftsdatenbank an, die Rolle der Primärdatenbank zu übernehmen.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Erforderliche Verbindung:

Exemplar. Die API stellt eine Datenbankverbindung her, sofern noch keine vorhanden ist, und schließt die Datenbankverbindung nach Beendigung der API.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2HADRTakeover */
/* ... */
SQL_API_RC SQL_API_FN
  db2HADRTakeover (
  db2Uint32 versionNumber,
  void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRTakeoverStruct
{
  char                        *piDbAlias;
  char                        *piUserName;
  char                        *piPassword;
  db2Uint16                   iByForce;
} db2HADRTakeoverStruct;
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2gHADRTakeover */
/* ... */
SQL_API_RC SQL_API_FN
  db2gHADRTakeover (
```

```
  db2Uint32 versionNumber,
  void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRTakeoverStruct
{
  char                        *piDbAlias;
   db2Uint32 iAliasLen;
  char                        *piUserName;
   db2Uint32 iUserNameLen;
  char                        *piPassword;
   db2Uint32 iPasswordLen;
  db2Uint16                   iByForce;
} db2gHADRTakeoverStruct;
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pParmStruct*, übergeben wird.

pParmStruct
Eingabe. Ein Zeiger auf die Struktur *db2HADRStartStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

piDbAlias
Eingabe. Ein Zeiger auf den Datenbankaliasnamen.

iAliasLen
Eingabe. Gibt die Länge des Datenbankaliasnamens in Byte an.

piUserName
Eingabe. Ein Zeiger auf den Benutzernamen, unter dem der Befehl ausgeführt wird.

iUserNameLen
Eingabe. Gibt die Länge des Benutzernamens in Byte an.

piPassword
Eingabe. Ein Zeiger auf eine Zeichenfolge, die das Kennwort enthält.

iPasswordLen
Eingabe. Gibt die Länge des Kennworts in Byte an.

iByForce
Eingabe. Gültige Werte:

DB2HADR_NO_FORCE
Gibt an, dass nur dann eine Übernahme ausgeführt wird, wenn sich beide Systeme im Peerstatus befinden und miteinander verbunden sind; die Übernahme bewirkt einen Rollentausch zwischen der HADR-Primärdatenbank und der HADR-Bereitschaftsdatenbank.

DB2HADR_FORCE
Gibt an, dass die Bereitschaftsdatenbank die Rolle der Primärdatenbank übernimmt, ohne auf eine Bestätigung zu warten, dass die ursprüngliche Primärdatenbank heruntergefahren wurde.

Zugehörige Referenzen:

- „db2HADRStart - HADR starten" auf Seite 224
- „db2HADRStop - HADR stoppen" auf Seite 229
- „TAKEOVER HADR" auf Seite 237

Kapitel 8. Clusterunterstützung unter AIX

Unterstützung für High Availability Cluster Multi-Processing (HACMP)

Enhanced Scalability (ES - Erweiterte Skalierbarkeit) ist eine Einrichtung von High Availability Cluster Multi-Processing (HACMP) für AIX. Diese Einrichtung stellt dieselbe Fehlerbehebung durch Funktionsübernahme bereit und besitzt dieselbe Ereignisstruktur wie HACMP. Enhanced Scalability bietet außerdem Folgendes:

- Größere HACMP-Cluster
- Zusätzliche Fehlerbehandlung durch *benutzerdefinierte Ereignisse*. Überwachte Bereiche können benutzerdefinierte Ereignisse auslösen, die so unterschiedlich sein können wie das Abbrechen eines Prozesses oder die Tatsache, dass der Pagingbereich bald an die Kapazitätsgrenze stößt. Solche Ereignisse umfassen vorher und nachher ausgeführte Ereignisse, die dem Funktionsübernahmeprozess bei Bedarf hinzugefügt werden können. Sonderfunktionen, die für verschiedene Implementierungen spezifisch sind, können in die Abläufe der vorher und nachher ausgeführten HACMP-Ereignisströme integriert werden.

  Eine *rules-Datei* (`/usr/sbin/cluster/events/rules.hacmprd`) enthält die Definitionen der HACMP-Ereignisse. Benutzerdefinierte Ereignisse werden dieser Datei hinzugefügt. Die beim Eintritt von Ereignissen auszuführenden Prozedurdateien sind Teil dieser Definition.
- HACMP-Clientdienstprogramme zur Überwachung und Erkennung von Statusänderungen (in einem oder mehreren Clustern) von physischen, außerhalb des HACMP-Clusters befindlichen AIX-Knoten aus.

Die Knoten in HACMP ES-Clustern tauschen Nachrichten aus, die als *Überwachungssignale* oder *Keepalive-Pakete* bezeichnet werden, und durch die jeder Knoten die anderen Knoten über seine Verfügbarkeit informiert. Ein Knoten, der nicht mehr reagiert, veranlasst die übrigen Knoten im Cluster, die Wiederherstellung zu starten. Der Wiederherstellungsprozess wird als *node_down-Ereignis* oder auch als *Funktionsübernahme* bezeichnet. Nach einem Wiederherstellungsprozess folgt die erneute Integration des Knotens in den Cluster. Dieses Ereignis wird als *node_up-Ereignis* bezeichnet.

Es gibt zwei Ereignistypen: Standardereignisse, die innerhalb der Funktionen von HACMP ES 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. Bei der Planung, welche Maßnahmen als Teil des Wiederherstellungsprozesses durchzuführen sind, ermöglicht HACMP zwei Funktionsübernahmeoptionen: den reinen Bereitschaftsmodus (Hot bzw. Idle Standby) und den Modus der gegenseitigen Übernahme (Mutual Takeover).

Anmerkung: Wenn Sie HACMP verwenden, stellen Sie sicher, dass DB2®-Exemplare nicht zur Bootzeit gestartet werden. Verwenden Sie dazu das Dienstprogramm **db2iauto** wie folgt:

```
db2iauto -off exemplarname
```

Dabei gilt Folgendes:

`exemplarname` ist der Anmeldename des Exemplars.

Clusterkonfiguration

In einer Konfiguration im *Bereitschaftsmodus* (Hot Standby) hat der AIX-Prozessorknoten, der als Übernahmeknoten fungiert, *keine andere* Arbeitsbelastung. Bei einer Konfiguration für die *gegenseitige Funktionsübernahme* hat der AIX-Prozessorknoten, der als Übernahmeknoten fungiert, *andere* Arbeitsbelastungen.

Allgemein wird DB2 Universal Database in Umgebungen mit partitionierten Datenbanken im Modus der gegenseitigen Übernahme (Mutual Takeover) ausgeführt, wobei es auf jedem Knoten Partitionen gibt. Eine Ausnahme bildet ein Szenario, in dem der Katalogknoten Teil einer Bereitschaftskonfiguration ist.

Bei der Planung einer umfangreichen DB2-Installation auf einem RS/6000® SP™ mit HACMP ES müssen Sie besonders auf die Verteilung der Knoten des Clusters innerhalb der oder zwischen den RISC System/6000 SP-Rahmen (Frames) achten. Wenn ein Knoten und der zugehörige Ausweichknoten in verschiedenen SP-Rahmen angelegt sind, ist eine Funktionsübernahme möglich, wenn ein ganzer Rahmen ausfällt (d. h., die Stromversorgung/Switch Boards des Rahmens ausfallen). Dies ist jedoch eine sehr seltene Ausnahme, weil es *N*+1 Stromversorgungen in jedem SP-Rahmen gibt und jeder SP-Switch redundante Pfade mit *N*+1 Ventilatoren und Stromzuführungen besitzt. Im Fall eines Rahmenausfalls ist eventuell ein manueller Eingriff erforderlich, um die übrigen Rahmen wieder in Funktion zu setzen. Diese Wiederherstellungsprozedur ist im Handbuch *SP Administration Guide* dokumentiert. HACMP ES ermöglicht die Wiederherstellung nach SP-Knotenausfällen. Die Wiederherstellung nach Rahmenausfällen ist von einer geeigneten Anordnung von Clustern innerhalb eines oder mehrerer SP-Rahmen(s) abhängig.

Ein weiterer Planungsaspekt ist die Verwaltung großer Cluster. Die Verwaltung eines kleinen Clusters ist einfacher als die eines großen. Aber die Verwaltung eines einzelnen großen Clusters ist 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 Netzwerken, 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, starten und sich in einen HACMP-Cluster integrieren. Eine Konfiguration mit mehreren Clustern lässt sich schneller starten als ein einziger großer Cluster. HACMP ES unterstützt sowohl einzelne als auch mehrere Cluster, vorausgesetzt, ein Knoten und der zugehörige Ausweichknoten befinden sich im selben Cluster.

Die HACMP ES-Funktion der Fehlerbehebung durch Funktionsübernahme ermöglicht eine vordefinierte Zuordnung einer Ressourcengruppe zu einem physischen Knoten (auch als *hintereinandergeschaltete Zuordnung* bezeichnet). Die 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 HACMP ES 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 Tabellenbereichsbehälter 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. Gerade diese Art von Operation wird durch HACMP automatisiert. HACMP ES kann außerdem dazu verwendet werden, NFS-Dateisysteme wiederherzustellen, die von Hauptbenutzerverzeichnissen des DB2-Exemplars verwendet werden.

Lesen Sie sich im Rahmen Ihrer Planung für die Wiederherstellung mit DB2 UDB in einer Umgebung mit partitionierten Datenbanken die Dokumentation zu HACMP ES gut durch. Sie sollten die Handbücher zu Konzepten, Planung, Installation und Verwaltung lesen und anschließend die Wiederherstellungsarchitektur für Ihre Umgebung entwerfen. Ermitteln Sie für die einzelnen Subsysteme, die Sie für die Wiederherstellung anhand bekannter möglicher Fehlerpunkte bestimmt haben, die benötigten HACMP-Cluster und die Wiederherstellungsknoten (entweder im Bereitschaftsmodus oder im Modus der gegenseitigen Übernahme).

Es ist ausdrücklich zu empfehlen, sowohl Platten als auch Adapter in der externen Plattenkonfiguration zu spiegeln. Bei physischen DB2-Knoten, die für HACMP 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, so dass 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 Behälternamen über alle Datenbanken hinweg eindeutig sein müssen.

Eine Methode zur Sicherstellung der Eindeutigkeit besteht darin, die Partitionsnummer als Teil des Namens zu verwenden. Sie können einen Knotenausdruck für die Syntax von Behälterzeichenfolgen angeben, wenn Sie SMS- oder DMS-Behälter erstellen. Wenn Sie den Ausdruck angeben, kann die Knotennummer Teil des Behälternamens sein, oder, wenn Sie zusätzliche Argumente angeben, können die Ergebnisse dieser Argumente Teil des Behälternamens sein. Verwenden Sie das Argument ″ $N″ ([Leerzeichen]$N) zur Angabe des Knotenausdrucks. Das Argument muss an das Ende der Behälterzeichenfolge gesetzt werden und kann nur in einem der folgenden Formate verwendet werden:

Tabelle 1. Argumente zur Behältererstellung. Als Knotennummer wird fünf (5) angenommen.

Syntax	Beispiel	Wert
[Leerzeichen]$N	″ $N″	5
[Leerzeichen]$N+[zahl]	″ $N+1011″	1016
[Leerzeichen]$N%[zahl]	″ $N%3″	2
[Leerzeichen]$N+[zahl]%[zahl]	″ $N+12%13″	4
[Leerzeichen]$N%[zahl]+[zahl]	″ $N%3+20″	22

Anmerkungen:

1. % bedeutet Modulus.
2. In allen Fällen werden die Operatoren von links nach rechts ausgewertet.

Es folgen einige Beispiele zur Erstellung von Behältern mit Hilfe dieses Spezialarguments:

- Erstellen von Behältern zur Verwendung auf einem Zweiknotensystem

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
   (device '/dev/rcont $N' 20000)
```

  Die folgenden Behälter würden verwendet:

```
/dev/rcont0   - auf Knoten 0
/dev/rcont1   - auf Knoten 1
```

- Erstellen von Behältern zur Verwendung auf einem Vierknotensystem

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
   (file '/DB2/containers/TS2/container $N+100' 10000)
```

  Die folgenden Behälter 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 Behältern zur Verwendung auf einem Zweiknotensystem

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
   ('/TS3/cont $N%2, '/TS3/cont $N%2+2')
```

  Die folgenden Behälter 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 HACMP ES

Nach der Konfiguration werden die Datenbankpartitionen in einem Exemplar durch HACMP ES 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 HACMP-Cluster mit zwei Knoten als vier Cluster mit 16 Knoten parallel zu starten.

Eine Prozedurdatei, `rc.db2pe`, gehört zum Lieferumfang von DB2 UDB Enterprise Server Edition (und wird auf jedem Knoten in `/usr/bin` installiert), um Hilfestellung bei der Konfiguration für HACMP ES-Funktionsübernahme bzw. -Wiederherstellung auf Bereitschaftsknoten bzw. Knoten mit gegenseitiger Übernahme zu geben. 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.)

HACMP ES-Ereignisüberwachung und benutzerdefinierte Ereignisse

Ein Beispiel für ein benutzerdefiniertes Ereignis ist das Einleiten einer Funktionsübernahmeoperation, wenn auf einem bestimmten Knoten ein Prozess abgebrochen wird. Beispiele, die benutzerdefinierte Ereignisse illustrieren, wie zum Beispiel das Herunterfahren einer Datenbankpartition und das Erzwingen eines Transaktionsabbruchs, um Pagingbereich freizugeben, befinden sich im Unterverzeichnis `samples/hacmp/es`.

Eine rules-Datei, /user/sbin/cluster/events/rules.hacmprd, enthält Definitionen von HACMP-Ereignissen. Jede Ereignisbeschreibung in dieser Datei besitzt die folgenden neun Komponenten:

- Ereignisname, der eindeutig sein muss.
- Status oder Qualifikationsmerkmal für das Ereignis. Der Ereignisname und der Status sind die Regelauslöser. HACMP ES Cluster Manager leitet die Wiederherstellung nur dann ein, wenn eine Regel mit einem Auslöser vorhanden ist, die dem Ereignisnamen und dem Status entspricht.
- Ressourcenprogrammpfad, eine vollständige Pfadangabe der Datei *xxx*.rp, die das Wiederherstellungsprogramm enthält.
- Wiederherstellungstyp. Diese Angabe ist zur zukünftigen Verwendung reserviert.
- Wiederherstellungsebene. Diese Angabe ist zur zukünftigen Verwendung reserviert.
- Ressourcenvariablenname, der für Ereignisse des Ereignismanagers verwendet wird.
- Exemplarvektor, der für Ereignisse des Ereignismanagers verwendet wird. Dies ist eine Gruppe von Elementen der Form "name=wert". Die Werte identifizieren die Kopie der Ressource im System eindeutig und somit auch die Kopie der Ressourcenvariablen.
- Vergleichselement, das für Ereignisse des Ereignismanagers verwendet wird. Dies ist ein Vergleichsausdruck zwischen einer Ressourcenvariablen und anderen Elementen. Wenn der Ausdruck wahr ist, generiert das Subsystem der Ereignisverwaltung ein Ereignis, um den Clustermanager und die entsprechende Anwendung zu benachrichtigen.
- Rearm-Vergleichselement, das für Ereignisse des Ereignismanagers verwendet wird. Dieses Vergleichselement dient zur Generierung eines Ereignisses, das den Status des primären Vergleichselements ändert. In der Regel stellt dieses Vergleichselement die Inversion des primären Vergleichselements dar. Es kann zusammen mit dem Ereignisvergleichselement auch dazu verwendet werden, einen oberen und einen unteren Grenzwert für eine bestimmte Bedingung festzulegen.

Jedes Objekt benötigt eine Zeile in der Ereignisdefinition, auch wenn die Zeile nicht verwendet wird. Wenn solche Zeilen gelöscht werden, kann HACMP ES Cluster Manager die Ereignisdefinition syntaktisch nicht korrekt analysieren, was zu einer Blockierung des Systems führen kann. Jede Zeile, die mit einem Zeichen "#" beginnt, wird als Kommentarzeile behandelt.

Anmerkung: Die rules-Datei verlangt exakt neun Zeilen für jede Ereignisdefinition, wobei die Kommentarzeilen nicht mitgezählt werden. Beim Hinzufügen eines benutzerdefinierten Ereignisses am Ende der rules-Datei muss unbedingt darauf geachtet werden, nicht benötigte Leerzeilen am Ende der Datei zu entfernen. Ansonsten wird der Knoten blockiert.

HACMP ES verwendet die PSSP-Ereigniserkennung zur Behandlung benutzerdefinierter Ereignisse. Das PSSP-Subsystem zur Ereignisverwaltung (Event Management) stellt eine umfassende Ereigniserkennung durch Überwachung verschiedener Hardware- und Softwareressourcen zur Verfügung.

Der Prozess kann folgendermaßen zusammengefasst werden:

1. Entweder Group Services/ES (für vordefinierte Ereignisse) oder die Ereignisverwaltung (für benutzerdefinierte Ereignisse) informiert HACMP ES Cluster Manager über das Ereignis.
2. Cluster Manager liest die Datei `rules.hacmprd` und bestimmt das Wiederherstellungsprogramm, das dem Ereignis zugeordnet ist.
3. Cluster Manager führt das Wiederherstellungsprogramm aus, das aus einer Folge von Wiederherstellungsbefehlen besteht.
4. Das Wiederherstellungsprogramm führt die Wiederherstellungsbefehle aus, die Shellprozeduren oder Binärbefehle sein können. (In HACMP für AIX stimmen die Wiederherstellungsbefehle mit der HACMP-Ereignisprozeduren überein.)
5. Cluster Manager empfängt den Rückkehrstatus von den Wiederherstellungsbefehlen. Ein unerwarteter Status führt zum Blockieren des Clusters, bis ein manueller Eingriff (mit Hilfe von `smit cm_rec_aids` oder mit Hilfe des Befehls `/usr/sbin/cluster/utilities/clruncmd`) durchgeführt wird.

Ausführliche Informationen zur Implementierung und zum Design von IBM® DB2 Universal Database™-Umgebungen mit hoher Verfügbarkeit unter AIX finden Sie in den folgenden White Papers, die auf der Website "DB2 UDB and DB2 Connect™ Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar sind:

- "IBM DB2 Universal Database Enterprise Edition for AIX and HACMP/ES"
- "IBM DB2 Universal Database Enterprise - Extended Edition for AIX and HACMP/ES"

Zugehörige Referenzen:

- „db2start - Start DB2 Command" im Handbuch *Command Reference*

Kapitel 9. Clusterunterstützung unter dem Windows-Betriebssystem

Unterstützung für Microsoft Cluster Server

Einführung

Microsoft Cluster Server (MSCS) ist eine Funktion der Betriebssysteme Windows® NT Server, Windows 2000 Server und Windows Server 2003. Diese Software unterstützt die Verbindung zweier Server (bis zu vier Servern in DataCenter Server) in einem Cluster für hohe Verfügbarkeit sowie die einfachere Verwaltung von Daten und Anwendungen. MSCS stellt automatisch Server- und Anwendungsfehler bzw. -ausfälle fest und führt eine Wiederherstellung durch. Mit MSCS können Serverauslastungen verschoben werden, um eine gleichmäßige Maschinennutzung zu gewährleisten und geplante Wartungsarbeiten ohne Ausfallzeiten durchführen zu können.

Die folgenden DB2®-Produkte bieten Unterstützung für MSCS:

- DB2 Universal Database™ Workgroup Server Edition
- DB2 Universal Database Enterprise Server Edition (DB2 ESE)
- DB2 Universal Database Connect Enterprise Edition (DB2 CEE)

DB2-MSCS-Komponenten

Ein Cluster ist eine Konfiguration mit mindestens zwei Knoten, die jeweils unabhängige Computersysteme sind. Der Cluster wird von Netzwerkclients als ein einzelner Server betrachtet.

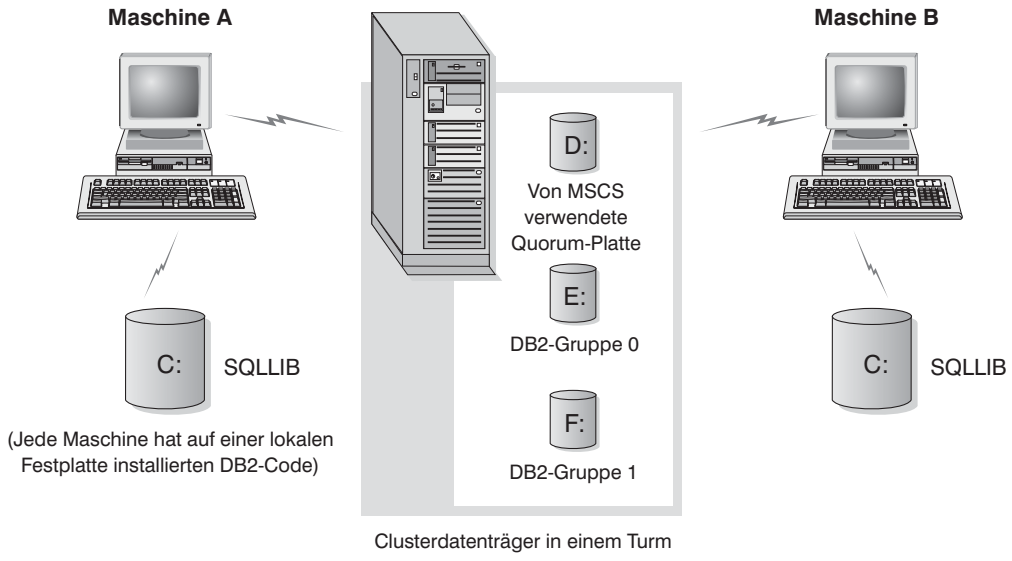


Abbildung 17. Beispiel für eine MSCS-Konfiguration

Die Knoten in einem MSCS-Cluster werden durch mindestens einen gemeinsam benutzten 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 benutzte Speicherbus ist mit mindestens einer Platte verbunden. Für jede Platte im gemeinsam benutzten Bus liegt das Eigentumsrecht jeweils nur bei einem Clusterknoten. Die DB2-Software befindet sich auf der lokalen Platte. Die DB2-Datenbankdateien (Tabellen, Indizes, Protokolldateien usw.) befinden sich auf den gemeinsam benutzten Platten. Da MSCS die Verwendung von unformatierten Partitionen in einem Cluster nicht unterstützt, kann DB2 in MSCS-Umgebungen nicht zur Verwendung von unformatierten Einheiten konfiguriert werden.

Die DB2-Ressource

In einer MSCS-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 mit MSCS erstellt DB2 seinen eigenen Ressourcentyp, ″DB2″. Jede DB2-Ressource verwaltet ein DB2-Exemplar; bei Ausführung in einer Umgebung mit partitionierten Datenbanken verwaltet jede DB2-Ressource eine Datenbankpartition. Der Name der DB2-Ressource ist der Exemplarname; in Umgebungen mit partitionierten Datenbanken setzt sich der Name der DB2-Ressource aus dem Exemplarnamen und der Nummer der Partition (oder des Knotens) zusammen.

Prä-Online- und Post-Onlineprozeduren

Sie können Prozeduren ausführen, bevor eine DB2-Ressource in den Onlinestatus versetzt wird und nachdem sie in den Onlinestatus versetzt wurde. Diese Prozeduren werden als Prä-Onlineprozeduren bzw. Post-Onlineprozeduren bezeichnet. Diese Prozeduren sind BAT-Dateien, die DB2- und Systembefehle ausführen können.

In Situationen, in denen mehrere DB2-Exemplare auf derselben Maschine ausgeführt werden, können Sie mit Hilfe der Prä- und Post-Onlineprozeduren die Konfiguration anpassen, so dass beide Exemplare erfolgreich gestartet werden können. Wenn eine Funktionsübernahme stattfindet, können Sie mit der Post-Onlineprozedur eine manuelle Datenbankwiederherstellung ausführen. Eine Post-Onlineprozedur 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 das DB2-Exemplar (oder den Knoten).
2. Ressource ’IP-Adresse’. Die Ressource ’IP-Adresse’ ermöglicht es Clientanwendungen, eine Verbindung zum DB2-Server herzustellen.
3. Ressource ’Netzwerkname’. Die Ressource ’Netzwerkname’ ermöglicht es Clientanwendungen, unter Verwendung eines Namens anstatt einer IP-Adresse eine Verbindung zum DB2-Server herzustellen. Für die Ressource ’Netzwerkname’ besteht eine Abhängigkeit zur Ressource ’IP-Adresse’.

Die Ressource 'Netzwerkname' ist optional. (Das Konfigurieren einer Ressource 'Netzwerkname' kann sich auf die Leistung der Funktionsübernahme auswirken.)

4. Mindestens eine Ressource 'Physische Platte'. Jede Ressource 'Physische Platte' verwaltet eine gemeinsam benutzte Platte im Cluster.

Anmerkung: Die Ressource 'DB2' ist so konfiguriert, dass sie von allen anderen Ressourcen in derselben Gruppe abhängt, so dass der DB2-Server nur dann gestartet werden kann, wenn alle anderen Ressourcen online sind.

Konfigurationen für Funktionsübernahme

Es sind zwei Konfigurationstypen verfügbar:
- Bereitschaftsmodus
- Gegenseitige Übernahme

In einer Umgebung mit partitionierten Datenbanken müssen nicht alle Cluster den gleichen Konfigurationstyp haben. Sie können Cluster haben, die im Bereitschaftsmodus konfiguriert sind, und andere, die für gegenseitige Übernahme konfiguriert sind. Wenn Ihr DB2-Exemplar zum Beispiel aus fünf Workstations besteht, können Sie zwei Maschinen zur gegenseitigen Übernahme und zwei im Bereitschaftsmodus konfigurieren und eine Maschine von der Funktionsübernahmekonfiguration ausnehmen.

Konfiguration für Bereitschaftsmodus

In einer Konfiguration für den Bereitschaftsmodus stellt eine Maschine im MSCS-Cluster eine dedizierte Übernahmeunterstützung bereit, und die andere Maschine ist Teil des Datenbanksystems. Wenn die zum Datenbanksystem gehörige Maschine ausfällt, wird ihr Datenbankserver auf der Übernahmemaschine gestartet. Wenn Sie in einem partitionierten Datenbanksystem mehrere logische Knoten auf einer Maschine betreiben und die Maschine ausfällt, werden die logischen Knoten auf der Übernahmemaschine gestartet. Abb. 18 zeigt ein Beispiel einer Konfiguration für den Bereitschaftsmodus.

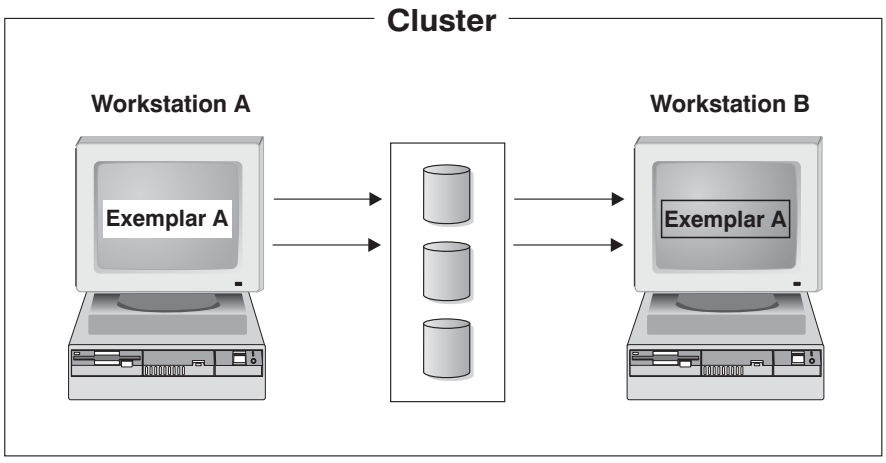


Abbildung 18. Konfiguration für Bereitschaftsmodus

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 MSCS-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. 19 zeigt ein Beispiel für die Konfiguration zur gegenseitigen Übernahme.

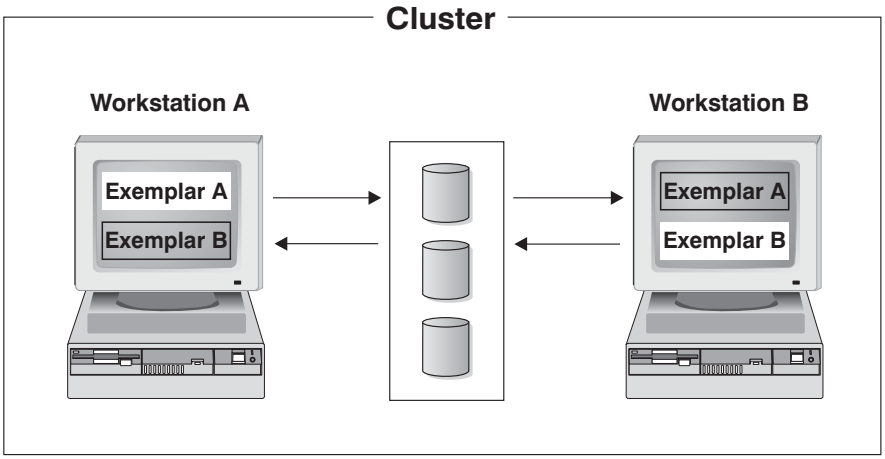


Abbildung 19. Konfiguration zur gegenseitigen Übernahme

Ausführliche Informationen zur Implementierung und zum Design von IBM® DB2 Universal Database-Umgebungen mit hoher Verfügbarkeit unter einem Windows-Betriebssystem finden Sie in den folgenden White Papers, die auf der Website "DB2 UDB and DB2 Connect™ Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar sind:

- "Implementing IBM DB2 Universal Database Enterprise - Extended Edition with Microsoft® Cluster Server"
- "Implementing IBM DB2 Universal Database Enterprise Edition with Microsoft Cluster Server"
- "DB2 Universal Database for Windows: High Availability Support Using Microsoft Cluster Server - Overview"

Kapitel 10. Clusterunterstützung für die Solaris-Betriebsumgebung

Clusterunterstützung für die Solaris-Betriebsumgebung

Eine hohe Verfügbarkeit in der Solaris™-Betriebsumgebung können Sie mit DB2® unter Einsatz von Sun™ Cluster oder Veritas Cluster Server (VCS) erreichen. Informationen zu Sun Cluster finden Sie im White Paper „DB2 Universal Database and High Availability on Sun Cluster 3.X", das auf der Website „DB2 UDB and DB2 Connect Online Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar ist. Informationen zu VERITAS Cluster Server finden Sie im White Paper „DB2 and High Availability on VERITAS Cluster Server", das auf der Website „IBM Support and downloads" unter http://www.ibm.com/support/docview.wss?uid=swg21045033 verfügbar ist.

Anmerkung: Wenn Sie Sun Cluster 3.0 oder Veritas Cluster Server verwenden, stellen Sie sicher, dass DB2-Exemplare nicht zur Bootzeit gestartet werden. Verwenden Sie dazu das Dienstprogramm **db2iauto** wie folgt:

```
db2iauto -off exemplarname
```

Dabei gilt Folgendes:

`exemplarname` ist der Anmeldename des Exemplars.

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 Netzwerkschnittstellen, mindestens einer öffentlichen Netzwerkschnittstelle sowie einigen gemeinsam benutzten Platten. Diese spezielle Konfiguration ermöglicht das Versetzen eines Datenbankservices von einer Maschine auf eine andere. Durch das Versetzen des Datenbankservices auf eine andere Maschine im Cluster wird es möglich, den Zugriff auf die zugehörigen Daten weiterhin aufrecht zu erhalten. Das Versetzen eines Datenbankservices von einer Maschine auf eine andere wird als *Funktionsübernahme* bezeichnet (siehe Abb. 20 auf Seite 252 zur Illustration).

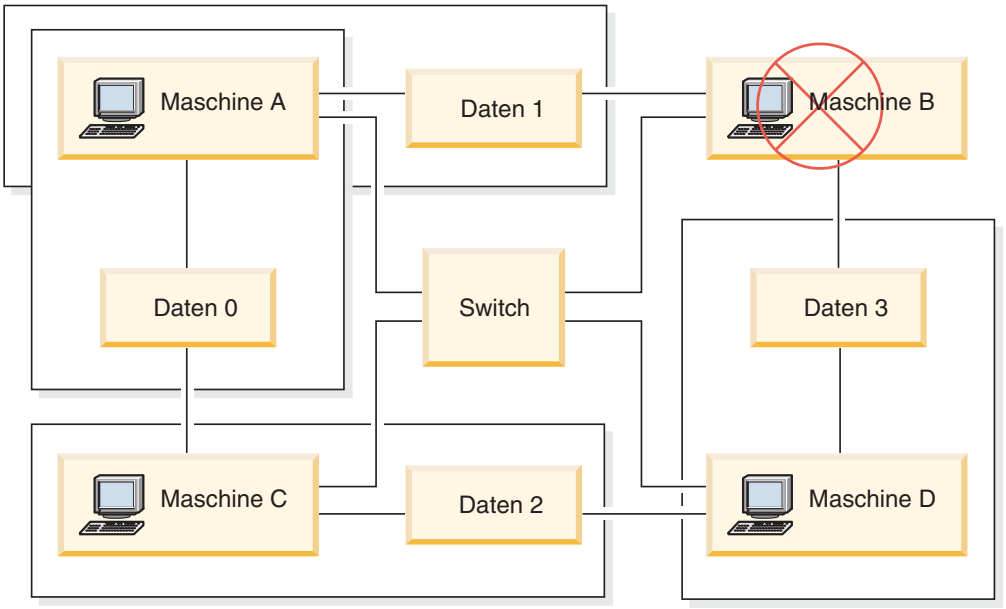


Abbildung 20. Funktionsübernahme. Wenn Maschine B ausfällt, wird ihr Datenbankservice auf eine andere Maschine im Cluster versetzt, so dass weiter auf die Daten zugegriffen werden kann.

Die privaten Netzwerkschnittstellen werden zum Senden von *Überwachungssignal*-nachrichten und von Steuernachrichten zwischen den Maschinen im Cluster verwendet. Die öffentlichen Netzwerkschnittstellen dienen zur direkten Kommunikation mit Clients des HA-Clusters. Die Platten in einem HA-Cluster sind mit zwei oder mehr Maschinen im Cluster verbunden, so dass bei Ausfall einer Maschine eine andere Maschine auf sie zugreifen kann.

Ein Datenbankservice, der auf einem HA-Cluster aktiv ist, besitzt mindestens eine öffentliche Netzwerkschnittstelle und eine Gruppe von Platten, die ihm zugeordnet sind. Die Clients eines HA-Datenbankservices stellen eine Verbindung über TCP/IP nur zu den logischen Netzwerkschnittstellen des Datenbankservices her. Wenn es zu einer Funktionsübernahme kommt, werden der Datenbankservice zusammen mit den zugehörigen Netzwerkschnittstellen 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 überbrücken zu können.

Auch wenn sich hochverfügbare Datenbankservices in ihrer Art sehr unterscheiden können, besitzen sie einige allgemeine Anforderungen. Clients eines hochverfügbaren Datenbankservices 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 tätigen.

Betrachten Sie einen Webbrowser, der auf einen hochverfügbaren Webserver 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

Webserver enthält. Der Browser erwartet, dass sowohl der Hostname als auch der Pfad nach einer Übernahme der Webserverfunktion gleich bleiben. Wenn der Browser gerade eine Datei vom Webserver herunterlädt, wenn die Serverfunktion von einer anderen Maschine übernommen wird, muss der Browser die Anforderung erneut absetzen.

Die Verfügbarkeit eines Datenbankservices 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 als Anzahl von „Neunen" angegeben:

```
99,99 % => Service ist (höchstens) 52,6 Minuten / Jahr außer Betrieb
99,999 % => Service ist (höchstens) 5,26 Minuten / Jahr außer Betrieb
99, 999 % => Service ist (höchstens) 31,5 Sekunden / Jahr außer Betrieb
```

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, so dass sie ersetzt werden können, bevor Probleme auftreten.

Fehlertoleranz

Eine andere Methode zur Erhöhung der Verfügbarkeit eines Datenbankservices ist Fehlertoleranz. Bei einer *fehlertoleranten* Maschine ist sämtliche Redundanz integriert, so dass 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.

Zugehörige Konzepte:

- „Unterstützung für Sun Cluster 3.0" auf Seite 254
- „Unterstützung für VERITAS Cluster Server" auf Seite 257

Unterstützung für Sun Cluster 3.0

Dieses Kapitel bietet einen Überblick darüber, wie DB2 mit Sun™ Cluster 3.0 arbeitet, um hohe Verfügbarkeit zu erreichen, und enthält eine Beschreibung des Agenten für hohe Verfügbarkeit (HA-Agent), der als Mittler zwischen zwei Softwareprodukten fungiert (siehe Abb. 21).

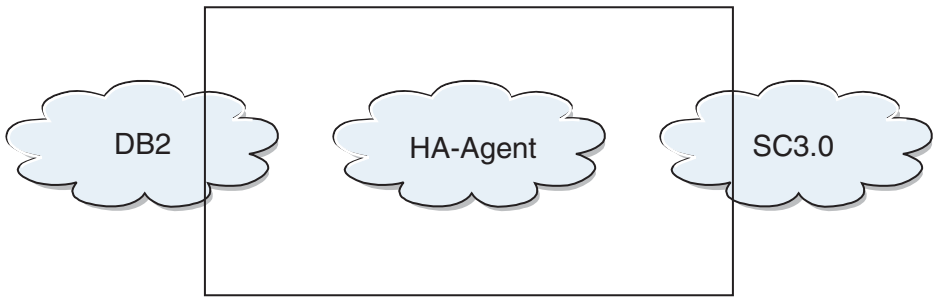


Abbildung 21. DB2, Sun Cluster 3.0 und hohe Verfügbarkeit. Die Abhängigkeiten zwischen DB2, Sun Cluster 3.0 und dem Agenten für hohe Verfügbarkeit (HA-Agent).

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 Services 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 Arbeitsbelastung 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/Globale Dateisysteme

Ein Cluster bzw. ein globales Dateisystem ist ein Proxy zwischen dem Kernel (auf einem Knoten) und dem Datenträgermanager des zu Grunde 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 zu Grunde 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.

Gerätegruppe

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 Wiederherstellung 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 Netzwerkressourcen (wie logische Hostnamen und gemeinsam benutze Adressen) und der Anwendungsexemplare.

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 Merkmalen, die Teil der statischen Konfiguration eines Clusters sind.

Der RGM verwendet die Ressourcentypmerkmale zur Verwaltung der Ressourcen eines bestimmten Typs.

Eine Ressource erbt die Merkmale und Werte ihres Ressourcentyps. Eine Ressource ist ein Exemplar der zu Grunde liegenden Anwendung, die auf dem Cluster ausgeführt wird. Der Name des Exemplars 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 ein zugeordnetes Merkmal, eine "Knotenliste", die vom Clusteradministrator definiert wird und anhand derer alle potenziellen Primärknoten oder Master der Ressourcengruppe angegeben werden.

Ausführliche Informationen zur Implementierung und zum Design von IBM® DB2 Universal Database-Umgebungen mit hoher Verfügbarkeit auf der Plattform Sun Cluster 3.0 finden Sie im White Paper "DB2 and High Availability on Sun Cluster 3.0", das auf der Website "DB2 UDB and DB2 Connect™ Support" unter http://www.ibm.com/software/data/pubs/papers/ bereitgestellt ist.

Zugehörige Konzepte:

- „Clusterunterstützung für die Solaris-Betriebsumgebung" auf Seite 251
- „Unterstützung für VERITAS Cluster Server" auf Seite 257

Unterstützung für VERITAS Cluster Server

Mit VERITAS Cluster Server (VCS) können geplante und ungeplante Ausfallzeiten vermieden werden. VERITAS Cluster Server vereinfacht die Serverkonsolidierung und ermöglicht eine effektive Verwaltung eines breiten Anwendungsspektrums in heterogenen Umgebungen. VERITAS Cluster Server unterstützt Cluster mit bis zu 32 Knoten in Storage Area Networks (SAN) und in konventionellen Client/Server-Umgebungen. Mit VERITAS Cluster Server können unterschiedlichste Systeme geschützt werden: von einzelnen geschäftskritischen Datenbankexemplaren bis hin zu sehr großen Clustern mit vielen Anwendungen in großen vernetzten Speicherumgebungen. In diesem Abschnitt werden die Funktionen von VERITAS Cluster Server kurz vorgestellt.

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.
  - Gängige Umgebungen erfordern (in jedem Clusterknoten) gespiegelte, persönliche Platten für die DB2® UDB-Binaries sowie von Knoten gemeinsam benutzte Platten für die DB2 UDB-Daten.
- Für Netzverbindungen:
  - Für die öffentlichen Netzverbindungen jede Netzverbindung, die Adressierung auf IP-Basis unterstützt.
  - Für die Überwachungssignalverbindungen (intern, zum Cluster) sind redundante Überwachungssignalverbindungen erforderlich; diese Voraussetzung kann durch den Einsatz von zwei zusätzlichen Ethernet-Controllern pro Server erfüllt werden, bzw. durch die Verwendung von einem zusätzlichen Ethernet-Controller pro Server und einer gemeinsam benutzten GAB-Platte pro Cluster.

Softwarevoraussetzungen

Die folgenden VERITAS-Softwarekomponenten sind 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 zur Verwaltung der Verfügbarkeit von Anwendungsservices wie DB2 UDB die Funktionsübernahme durch Anwendungen aktiviert. Der Status jedes einzelnen Clusterknotens und seiner zugehörigen Softwareservices wird regelmäßig überwacht. Wird ein Anwendungsservice (in diesem Fall der DB2 UDB-Service) durch einen auftretenden Fehler unterbrochen, stellt VERITAS Cluster Server und/oder der VCS HA-DB2-Agent den Fehler fest und führt automatisch Schritte zur Wiederherstellung des Services aus. Diese Schritte können den Neustart von DB2 UDB auf demselben Knoten oder das Versetzen von DB2 UDB auf einen anderen Knoten im Cluster und anschließenden Neustart auf diesem Knoten umfassen. Wenn eine Anwendung auf einen neuen Knoten migriert werden muss, versetzt VERITAS Cluster Server alles zur Anwendung Gehörende (z. B. Netz-IP-Adressen, Eigentumsrecht an zu Grunde liegendem Speicher) auf den neuen Knoten, so dass 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 zeigen.

Wenn in VERITAS Cluster Server ein Fehler auftritt, ist es möglich, dass Benutzer keine Unterbrechung des Services feststellen. 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 DB2), 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 verkürzt auch die geplanten Ausfallzeiten (z. B. für Wartung und Upgrades). Funktionsübernahmen können auch manuell eingeleitet werden. Wenn auf einem bestimmten Knoten ein Hardware- oder Betriebssystemupgrade ausgeführt werden muss, kann DB2 UDB auf einen anderen Knoten im Cluster migriert werden, der Upgrade ausgeführt und DB2 UDB anschließend wieder auf den ursprünglichen Knoten zurück migriert 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. DB2 UDB ist eine absturztolerante Anwendung.

Informationen dazu, wie Sie die für das Ausführen einer Funktionsübernahme benötigte Zeit unter Verwendung einer VERITAS CFS-, CVM- und VCS-Solution reduzieren können, finden Sie im White Paper „DB2 UDB Version 8 and VERITAS Database Edition: Accelerating Failover Times in DB2 UDB Database Environments", das auf der Website „DB2 UDB and DB2 Connect Online Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar ist.

Gemeinsam benutzter Speicher

Beim Einsatz mit dem VCS HA-DB2-Agenten erfordert VERITAS Cluster Server gemeinsam benutzten Speicher. Gemeinsam benutzter Speicher ist ein Speicher, der eine physische Verbindung zu mehreren Knoten im Cluster hat. Im gemeinsam benutzten 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 benutzten 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 fehlschlägt, kann Plattengruppe A aus dem fehlgeschlagenen 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 benutztem 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 benutztem 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 oder 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 Wiederherstellungsprozess 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 benutzen). Diese Agenten vereinfachen die Installation und Konfiguration von Clustern erheblich. Das mit VERITAS Cluster Server gelieferte Bündel umfasst über 20 Agenten.

Unternehmensagenten fokussieren in der Regel bestimmte Anwendungen, z. B. DB2 UDB. 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 Merkmalen, die dieser Ressource zugeordnet sind und die unter Gesichtspunkten der Hochverfügbarkeit wichtig sind. Eine Ressource erbt die Merkmale 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 unterste, überwachte Objekt ist eine Ressource, und es gibt verschiedene Ressourcentypen (z. B. gemeinsam benutzt, 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 UDB-Hochverfügbarkeitslösung (partitioniert oder nicht partitioniert) 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 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, und diese Hierarchie der Ressourcengruppenabhängigkeiten bildet für VERITAS Cluster Server die Grundlage, auf der es die Auswirkungen verschiedener Ressourcenausfälle einschätzt und die Wiederherstellung verwaltet. Beispiel: Falls die Ressourcengruppe ClientAnw1 nicht in den Onlinestatus versetzt werden kann, bevor nicht die Ressourcengruppe DB2 erfolgreich gestartet wurde, wird Ressourcengruppe ClientAnw1 als abhängig von Ressourcengruppe DB2 betrachtet.

Ausführliche Informationen zur Implementierung und zum Design von IBM DB2 Universal Database-Umgebungen mit hoher Verfügbarkeit finden Sie im White Paper mit dem Titel "DB2 UDB and High Availability with VERITAS Cluster Server", das Sie auf der Website http://www.ibm.com/support anzeigen können, indem Sie nach dem Stichwort "1045033" suchen.

Zugehörige Konzepte:

- „Clusterunterstützung für die Solaris-Betriebsumgebung" auf Seite 251
- „Unterstützung für Sun Cluster 3.0" auf Seite 254

Teil 3. Anhänge und Schlussteil

Anhang A. Lesen von Syntaxdiagrammen

Ein Syntaxdiagramm zeigt, wie ein Befehl angegeben werden muss, damit das Betriebssystem die Eingabe richtig auswerten kann.

Lesen Sie ein Syntaxdiagramm von links nach rechts und von oben nach unten. Folgen Sie dabei der horizontalen Linie (dem Hauptpfad). Wenn eine Linie mit einer Pfeilspitze endet, wird die Befehlssyntax fortgesetzt, und die nächste Linie beginnt mit einer Pfeilspitze. Ein vertikaler Balken kennzeichnet das Ende der Befehlssyntax.

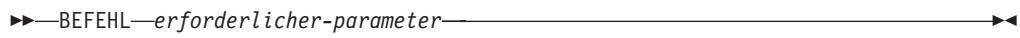
Bei der Eingabe von Daten eines Syntaxdiagramms müssen Sie auch Zeichen wie Anführungszeichen und Gleichheitszeichen eingeben.

Parameter sind als Schlüsselwörter oder Variablen klassifiziert:

- Schlüsselwörter sind Konstanten und werden in Großbuchstaben dargestellt. An der Eingabeaufforderung können sie jedoch in beliebiger Schreibweise eingegeben werden. Ein Befehlsname ist ein Beispiel für ein Schlüsselwort.
- Variablen sind Namen oder Werte, die vom Benutzer angegeben werden, und werden in Kleinbuchstaben dargestellt. An der Eingabeaufforderung können sie jedoch in beliebiger Schreibweise eingegeben werden, sofern dies nicht ausdrücklich anders angegeben ist. Ein Dateiname ist ein Beispiel für eine Variable.

Ein Parameter kann eine Kombination aus einem Schlüsselwort und einer Variablen sein.

Erforderliche Parameter werden im Hauptpfad angegeben:



Optionale Parameter werden unterhalb des Hauptpfads angegeben:

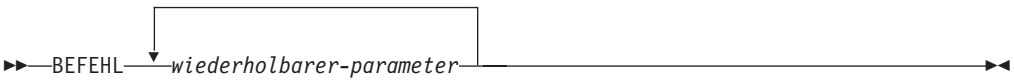
Der Standardwert eines Parameters wird oberhalb des Pfads angegeben:

Ein Stapel von Parametern, bei dem der erste Parameter im Hauptpfad angegeben ist, gibt an, dass einer der Parameter gewählt werden muss:

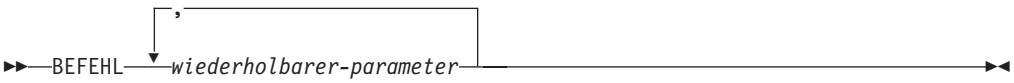
Ein Stapel von Parametern, bei dem der erste Parameter unterhalb des Hauptpfads angegeben ist, gibt an, dass einer der Parameter gewählt werden kann:

Ein nach links zurückweisender Pfeil oberhalb des Pfads gibt an, dass Elemente gemäß den folgenden Konventionen wiederholt werden können:

- Wenn der Pfeil durchgängig ist, kann das Element in einer Liste wiederholt werden, wobei die Elemente durch Leerstellen zu trennen sind:

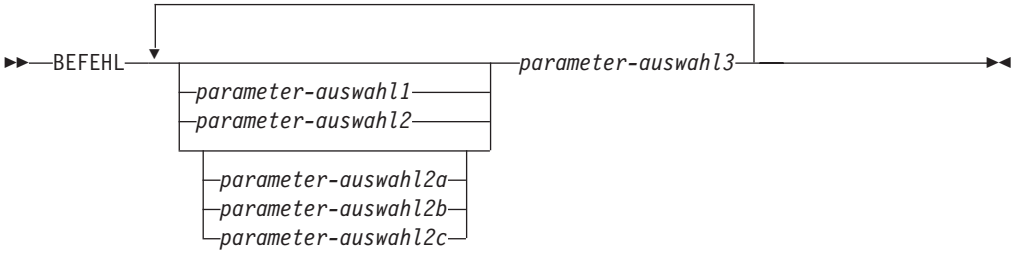


- Wenn der Pfeil ein Komma enthält, kann das Element in einer Liste wiederholt werden, wobei die Elemente durch Kommas zu trennen sind:



Elemente aus Parameterstapeln können gemäß den Stapelkonventionen für die zuvor erläuterten erforderlichen und optionalen Parameter wiederholt werden.

Manche Syntaxdiagramme enthalten Parameterstapel innerhalb von anderen Parameterstapeln. Elemente aus Stapeln können nur gemäß den zuvor erläuterten Konventionen wiederholt werden. Das heißt, wenn über einem inneren Stapel kein nach links zurückweisender Pfeil ist, über einem äußeren Stapel jedoch schon, kann nur ein einziger Parameter aus dem inneren Stapel gewählt und mit einem beliebigen Parameter des äußeren Stapels kombiniert werden; diese Kombination kann dann wiederholt werden. Das folgende Diagramm zeigt beispielsweise, dass der Parameter *parameter-auswahl2a* mit dem Parameter *parameter-auswahl2* kombiniert und diese Kombination dann wiederholt werden kann (*parameter-auswahl2* plus *parameter-auswahl2a*):



Einigen Befehlen ist ein optionaler Pfadparameter vorangestellt:

Wenn dieser Parameter nicht angegeben ist, sucht das System den Befehl im aktuellen Verzeichnis. Wenn der Befehl nicht gefunden wird, sucht das System weiterhin in allen Verzeichnissen in den Pfaden, die in der Datei `.profile` bzw. in der Anweisung PATH angegeben sind.

Einige Befehle weisen syntaktische Varianten auf, die funktionell identisch sind:

Anhang B. Warnungen, Fehler- und Beendigungsnachrichten

Die von den verschiedenen Dienstprogrammen generierten Nachrichten sind in den SQL-Nachrichten enthalten. Diese Nachrichten werden vom Datenbankmanager generiert, wenn eine Warnungs- oder Fehlerbedingung erkannt wurde. Jede Nachricht besitzt eine Nachrichten-ID, die aus einem Präfix (SQL) und einer vier- oder fünfstelligen Nachrichtennummer zusammengesetzt ist. Es gibt Informationsnachrichten, Warnungen und Nachrichten über kritische Systemfehler. Nachrichten-IDs, die mit `N` enden, sind Fehlernachrichten. IDs, die mit `W` enden, sind Warnungen oder Informationsnachrichten. Nachrichten-IDs, die mit `C` enden, weisen auf kritische Systemfehler hin.

Die Nachrichtennummer wird auch als *SQLCODE-Wert* bezeichnet. Der SQLCODE-Wert wird als positive oder negative Zahl an die Anwendung weitergegeben, je nach Nachrichtentyp (N, W oder C). N und C ergeben negative Werte, W ergibt positive Werte. DB2 gibt den SQLCODE-Wert an die Anwendung zurück, und die Anwendung kann die dem SQLCODE-Wert zugeordnete Nachricht abrufen. DB2 gibt auch einen *SQLSTATE*-Wert bei Bedingungen zurück, die das Ergebnis einer SQL-Anweisung sein könnten. Einigen SQLCODE-Werten sind SQLSTATE-Werte zugeordnet.

Sie können die Informationen in diesen Buch verwenden, um einen Fehler oder ein Problem einzugrenzen und das Problem zu beheben, indem Sie die entsprechende Wiederherstellungsmaßnahme ausführen. Diese Informationen können auch verwendet werden, um sich darüber zu informieren, wo Nachrichten generiert und protokolliert werden.

SQL-Nachrichten und der Nachrichtentext, die den SQLSTATE-Werten zugeordnet sind, können auch über die Befehlszeile des Betriebssystems aufgerufen werden. Geben Sie Folgendes an der Eingabeaufforderung des Betriebssystems ein, um die Hilfefunktion für diese Fehlernachrichten aufzurufen:

```
db2 ? SQLnnnnn
```

Dabei steht *nnnnn* für die Nachrichtennummer. Auf Systemen mit UNIX-Basis empfiehlt es sich, Begrenzer in Form von doppelten Anführungszeichen zu verwenden. Hierdurch werden Probleme verhindert, die auftreten können, wenn das Verzeichnis Dateinamen enthält, die aus einem Einzelbuchstaben bestehen:

```
db2 "? SQLnnnnn"
```

Die Nachrichten-ID, die als Parameter des Befehls **db2** akzeptiert wird, kann in beliebiger Schreibweise angegeben werden, und das letzte Zeichen ist nicht erforderlich. Demzufolge führen die folgenden Befehle alle zum gleichen Ergebnis:

```
db2 ? SQL0000N
db2 ? sql0000
db2 ? SQL0000n
```

Wenn der Nachrichtentext zu lang für den Bildschirm ist, können Sie den folgenden Befehl verwenden (auf UNIX-basierten Basisbetriebssystemen und anderen, die den Pipe-Befehl "more" unterstützen):

```
db2 ? SQLnnnnn | more
```

Sie können die Ausgabe auch in eine Datei umleiten, die dann angezeigt werden kann.

Die Hilfe kann auch im interaktiven Eingabemodus aufgerufen werden. Geben Sie Folgendes an der Eingabeaufforderung des Betriebssystems ein, um diesen Modus zu aktivieren:

```
db2
```

Geben Sie Folgendes an der Eingabeaufforderung `db2 =>` ein, um die Hilfe zu DB2-Nachrichten im interaktiven Eingabemodus aufzurufen:

```
? SQLnnnnn
```

Der Nachrichtentext, der SQLSTATE-Werten zugeordnet ist, kann durch Absetzen folgender Befehle abgerufen werden:

```
db2 ? nnnnn
oder
db2 ? nn
```

Dabei ist *nnnnn* ein fünfstelliger SQLSTATE-Wert (alphanumerisch) und *nn* ein zweistelliger SQLSTATE-Klassencode (die ersten beiden Stellen des SQLSTATE-Werts).

Anhang C. Weitere DB2-Befehle

In diesem Anhang werden wiederherstellungsbezogene System- und CLP-Befehle beschrieben, die in diesem Handbuch nicht ausführlich behandelt werden.

Systembefehle

db2adutl - DB2-Objekte in TSM verwalten

Ermöglicht Benutzern das Abfragen, Extrahieren, Prüfen und Löschen von Sicherungsimages und Ladekopieimages, die mit Hilfe von Tivoli Storage Manager gespeichert wurden. Ermöglicht es Benutzern außerdem, Zugriff auf Objekte auf einem TSM-Server zu erteilen und zu widerrufen.

Auf UNIX-Systemen befindet sich dieses Dienstprogramm im Verzeichnis `sqllib/adsm`. Unter Windows befindet es sich im Verzeichnis `sqllib\bin`.

Berechtigung:

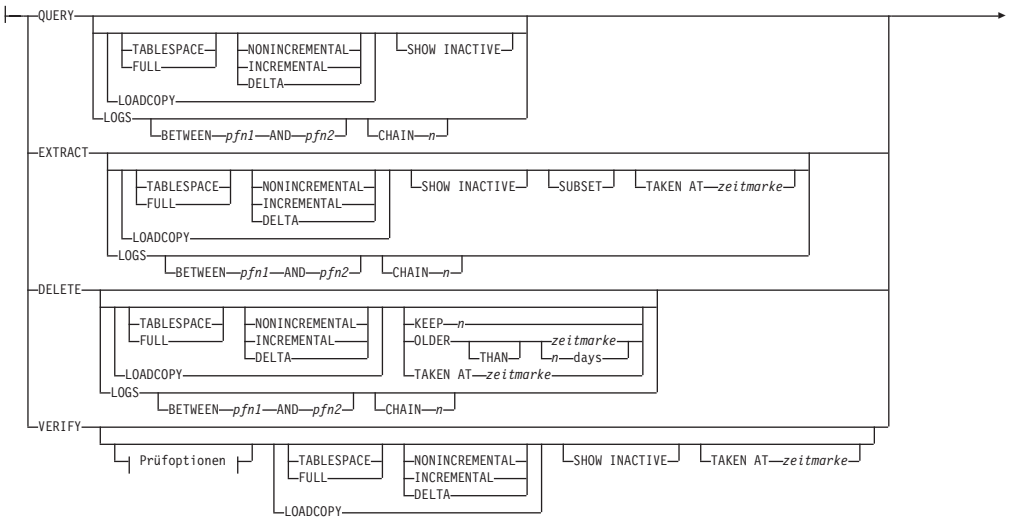
Keine

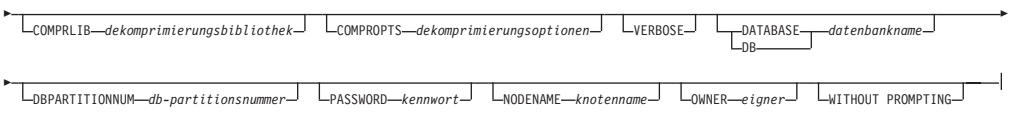
Erforderliche Verbindung:

Keine

Befehlssyntax:

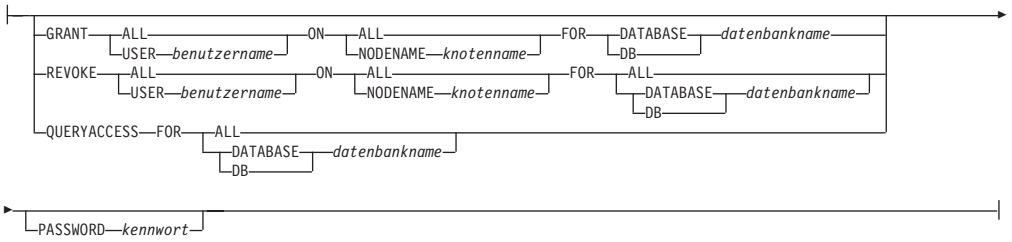
db2-objektoptionen:





Prüfoptionen:

Zugriffssteuerungsoptionen:



Befehlsparameter:

QUERY
Fragt den TSM-Server nach DB2-Objekten ab.

EXTRACT
Kopiert DB2-Objekte vom TSM-Server in das aktuelle Verzeichnis auf der lokalen Maschine.

DELETE
Inaktiviert Sicherungsobjekte oder löscht Protokollarchive auf dem TSM-Server.

VERIFY
Führt Konsistenzprüfung für die Sicherungskopie auf dem Server aus.

Anmerkung: Dieser Parameter kann dazu führen, dass das gesamte Sicherungsimage über das Netzwerk übertragen wird.

ALL Zeigt alle verfügbaren Informationen an.

CHECK
Zeigt die Ergebnisse der Prüfbits und der Kontrollsummen an.

DMS Zeigt Informationen aus den Headern von Datenseiten von DMS-Tabellenbereichen an.

HEADER
Zeigt die Datenträgerheaderinformationen an.

HEADERONLY
Zeigt dieselben Informationen an wie HEADER, liest jedoch nur

die 4 KB Datenträgerheaderdaten vom Anfang des Images. Das Image wird nicht auf Gültigkeit geprüft.

LFH Zeigt die Protokolldateiheaderdaten (LFH - Log File Header) an.

OBJECT
Zeigt Details aus den Objektheadern an.

PAGECOUNT
Zeigt die Anzahl Seiten der einzelnen Objekttypen an, die im Image vorhanden sind.

TABLESPACES
Zeigt Details des Tabellenbereichs, einschließlich Behälterinformationen, für die Tabellenbereiche im Image an.

TABLESPACESONLY
Zeigt die gleichen Informationen wie TABLESPACES an, das Image wird jedoch nicht auf Gültigkeit überprüft.

TABLESPACE
Berücksichtigt nur Sicherungsimages von Tabellenbereichen.

FULL Berücksichtigt nur vollständige Sicherungsimages von Datenbanken.

NONINCREMENTAL
Berücksichtigt nur Nicht-Teilsicherungsimages.

INCREMENTAL
Berücksichtigt nur Teilsicherungsimages.

DELTA
Berücksichtigt nur Deltateilsicherungsimages.

LOADCOPY
Berücksichtigt nur Ladekopieimages.

LOGS Berücksichtigt nur Protokollarchivimages.

BETWEEN *pfn1* **AND** *pfn2*
Gibt an, dass die Protokolle zwischen der Protokollfolgenummer 1 und der Protokollfolgenummer 2 verwendet werden sollen.

CHAIN *n*
Gibt die Ketten-ID der zu verwendenden Protokolle an.

SHOW INACTIVE
Berücksichtigt Sicherungsobjekte, die inaktiviert wurden.

SUBSET
Extrahiert Seiten aus einem Image in eine Datei. Zum Extrahieren von Seiten benötigen Sie eine Eingabe- und eine Ausgabedatei. Die Standardeingabedatei hat den Namen `extractPage.in`. Sie können den Namen der Standardeingabedatei außer Kraft setzen, indem Sie die Umgebungsvariable DB2LISTFILE auf einen vollständigen Pfad setzen. Die Eingabedatei hat folgendes Format:

Für SMS-Tabellenbereiche:

```
S <tabellenbereichsid> <objektid> <objekttyp> <startseite> <anzseiten>
```

Für DMS-Tabellenbereiche:

```
D <tabellenbereichsid> <objekttyp> <startseite> <anzseiten>
```

Anmerkung: Der Objekttyp ist nur beim Prüfen von DMS-Ladekopie-images erforderlich.

Für Protokolldateien:

```
L <protokollnummer> <anfangsposition> <anzseiten>
```

Für andere Daten (z. B. Anfangsdaten):

```
O <objekttyp> <anfangsposition> <anzbyte>
```

Die Standardausgabedatei hat den Namen `extractPage.out`. Sie können den Namen der Standardausgabedatei außer Kraft setzen, indem Sie die Umgebungsvariable DB2EXTRACTFILE auf einen vollständigen Pfad setzen.

TAKEN AT *zeitmarke*
Gibt ein Sicherungsimage über seine Zeitmarke an.

KEEP *n*
Inaktiviert alle Objekte des angegebenen Typs außer den *n* letzten Objekten nach Zeitmarke.

OLDER THAN *zeitmarke* **oder** *n* **days**
Gibt an, dass Objekte mit einer Zeitmarke älter als *zeitmarke* oder *n* Tage inaktiviert werden.

COMPRLIB *dekomprimierungsbibliothek*
Gibt den Namen der Bibliothek an, die zum Ausführen der Dekomprimierung verwendet werden soll. Der Name muss ein vollständig qualifizierter Pfad für eine Datei auf dem Server sein. Wird dieser Parameter nicht angegeben, versucht DB2, die im Image gespeicherte Bibliothek zu verwenden. Wurde die Sicherung nicht komprimiert, wird der Wert dieses Parameters ignoriert. Wenn die angegebene Bibliothek nicht geladen werden kann, schlägt die Operation fehl.

COMPROPTS *dekomprimierungsoptionen*
Beschreibt einen Block binärer Daten, der an die Initialisierungsroutine in der Dekomprimierungsbibliothek übergeben wird. DB2 übergibt diese Zeichenfolge direkt vom Client an den Server, daher muss die Handhabung der Bytefolgeumkehrung oder Codepagekonvertierung von der Dekomprimierungsbibliothek übernommen werden. Ist '@' das erste Zeichen des Datenblocks, werden die restlichen Daten von DB2 als Name einer Datei interpretiert, die sich auf dem Server befindet. DB2 ersetzt in diesem Fall den Inhalt dieses Datenblocks durch den Inhalt dieser Datei und übergibt diesen neuen Wert stattdessen an die Initialisierungsroutine. Die maximal zulässige Länge für diese Zeichenfolge ist 1024 Byte.

DATABASE *datenbankname*
Betrifft nur Objekte, die dem angegebenen Datenbanknamen zugeordnet sind.

DBPARTITIONNUM *db-partitionsnummer*
Betrifft nur Objekte, die auf der Datenbankpartition mit der angegebenen Nummer erstellt worden sind.

PASSWORD *kennwort*
Gibt das TSM-Clientkennwort für diesen Knoten an, falls dies erforderlich ist. Wird eine Datenbank ohne das zugehörige Kennwort angegeben, wird der für den Datenbankkonfigurationsparameter *tsm_password* angegebene Wert an TSM übergeben. Andernfalls wird kein Kennwort verwendet.

NODENAME *knotenname*
Betrifft nur Images, die einem spezifischen TSM-Knotennamen zugeordnet sind.

OWNER *eigner*
Betrifft nur Objekte, die der angegebene Eigner erstellt hat.

WITHOUT PROMPTING
Der Benutzer wird vor dem Löschen von Objekten nicht zur Bestätigung aufgefordert.

VERBOSE
Zeigt zusätzliche Informationen an.

GRANT ALL / USER *benutzername*
Fügt allen Benutzern oder den angegebenen Benutzern Zugriffsrechte auf TSM-Dateien am aktuellen TSM-Knoten hinzu. Das Erteilen von Zugriff für Benutzer ermöglicht diesen den Zugriff für alle aktuellen und zukünftigen Dateien, die zur angegebenen Datenbank gehören.

REVOKE ALL / USER *benutzername*
Entnimmt allen Benutzern oder den angegebenen Benutzern Zugriffsrechte auf TSM-Dateien am aktuellen TSM-Knoten.

QUERYACCESS
Ruft die aktuelle Zugriffsliste ab. Eine Liste von Benutzern und TSM-Knoten wird angezeigt.

ON ALL / NODENAME *knotenname*
Gibt den TSM-Knoten an, für den die Zugriffsrechte geändert werden.

FOR ALL / DATABASE *datenbankname*
Gibt die Datenbank an, die berücksichtigt werden muss.

Beispiele:

1. Es folgt eine Beispielausgabe des Befehls `db2 backup database rawsampl use tsm`:

```
Sicherung erfolgreich. Die Zeitmarke für diese Sicherung ist: 20031209184503
```

   Es folgt eine Beispielausgabe des Befehls `db2adutl query`, der im Anschluss an die Sicherungsoperation abgesetzt wurde:

```
Query for database RAWSAMPL

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20031209184403, Oldest log: S0000050.LOG, Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for RAWSAMPL

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for RAWSAMPL

Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for RAWSAMPL

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for RAWSAMPL

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for RAWSAMPL

Retrieving LOCAL COPY information.
  No LOCAL COPY images found for RAWSAMPL
```

```
Retrieving log archive information.
   Log file: S0000050.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.46.13
   Log file: S0000051.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.46.43
   Log file: S0000052.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.47.12
   Log file: S0000053.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.50.14
   Log file: S0000054.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.50.56
   Log file: S0000055.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.52.39
```

2. Es folgt eine Beispielausgabe des Befehls `db2adutl delete full taken at 20031209184503 db rawsampl`:

```
Query for database RAWSAMPL

Retrieving FULL DATABASE BACKUP information.
  Taken at: 20031209184503  DB Partition Number: 0    Sessions: 1

  Do you want to delete this file (Y/N)? y

    Are you sure (Y/N)? y


Retrieving INCREMENTAL DATABASE  BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for RAWSAMPL

Retrieving DELTA DATABASE  BACKUP information.
  No DELTA DATABASE BACKUP images found for RAWSAMPL
```

Es folgt eine Beispielausgabe des Befehls `db2adutl query`, der im Anschluss an die Operation abgesetzt wurde, mit der das Gesamtsicherungsimage gelöscht wurde. Beachten Sie die Zeitmarke für das Sicherungsimage.

```
Query for database RAWSAMPL

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20031209184403, Oldest log: S0000050.LOG, Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for RAWSAMPL

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for RAWSAMPL

Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for RAWSAMPL

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for RAWSAMPL

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for RAWSAMPL

Retrieving LOCAL COPY information.
  No LOCAL COPY images found for RAWSAMPL

Retrieving log archive information.
   Log file: S0000050.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.46.13
   Log file: S0000051.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.46.43
   Log file: S0000052.LOG, Chain Num: 0, DB Partition Number: 0,
```

```
 Taken at 2003-12-09-18.47.12
Log file: S0000053.LOG, Chain Num: 0, DB Partition Number: 0,
 Taken at 2003-12-09-18.50.14
Log file: S0000054.LOG, Chain Num: 0, DB Partition Number: 0,
 Taken at 2003-12-09-18.50.56
Log file: S0000055.LOG, Chain Num: 0, DB Partition Number: 0,
 Taken at 2003-12-09-18.52.39
```

3. Es folgt eine Beispielausgabe des Befehls `db2adutl queryaccess for all`:

```
Node                User                Database Name      type
------------------------------------------------------------------
bar2                jchisan             sample             B
<all>               <all>               test               B
------------------------------------------------------------------
Access Types: B – Backup images  L – Logs  A - both
```

Hinweise:

Aus den nachstehend aufgeführten Gruppen kann jeweils ein Parameter verwendet werden, um einzuschränken, welche Imagetypen in der Operation erfasst werden:

Granularität:

- FULL - nur Sicherungsimages von Datenbanken erfassen.
- TABLESPACE - nur Sicherungsimages von Tabellenbereichen erfassen.

Kumulativität:

- NONINCREMENTAL - nur Nicht-Teilsicherungsimages erfassen.
- INCREMENTAL - nur Teilsicherungsimages erfassen.
- DELTA - nur Deltateilsicherungsimages erfassen.

Kompatibilität:

Zur Kompatibilität mit älteren Versionen als Version 8 ist Folgendes möglich:

- Das Schlüsselwort NODE kann DBPARTITIONNUM ersetzen.

db2ckbkp - Sicherung überprüfen

Mit diesem Dienstprogramm können Sie die Integrität des Sicherungsimages testen und bestimmen, ob Sie das Image wiederherstellen können. Außerdem können Sie damit die Metadaten anzeigen, die im Sicherungsheader gespeichert sind.

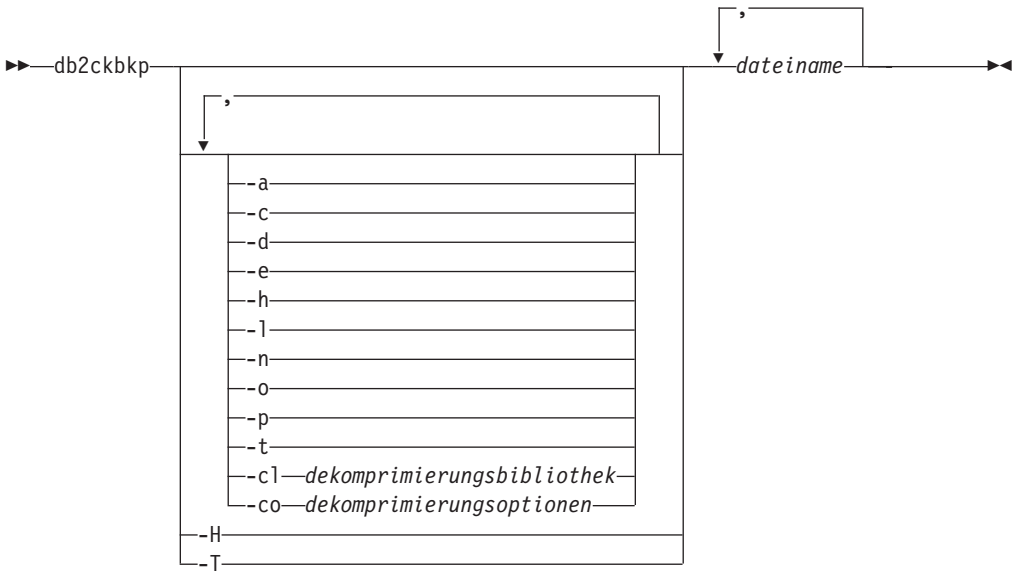
Berechtigung:

Beliebige Personen können auf das Dienstprogramm zugreifen. Benutzer müssen jedoch über Leseberechtigungen für die jeweiligen Imagesicherungen verfügen, um dieses Dienstprogramm für diese Sicherungen ausführen zu können.

Erforderliche Verbindung:

keine

Befehlssyntax:



Befehlsparameter:

-a Zeigt alle verfügbaren Informationen an.

-c Zeigt die Ergebnisse der Prüfbits und der Kontrollsummen an.

-cl *dekomprimierungsbibliothek*
Gibt den Namen der Bibliothek an, die zum Ausführen der Dekomprimierung verwendet werden soll. Der Name muss ein vollständig qualifizierter Pfad für eine Datei auf dem Server sein. Wird dieser Parameter nicht angegeben, versucht DB2, die im Image gespeicherte Bibliothek zu verwenden. Wurde die Sicherung nicht komprimiert, wird der Wert dieses Parameters ignoriert. Wenn die angegebene Bibliothek nicht geladen werden kann, schlägt die Operation fehl.

-co *dekomprimierungsoptionen*
Beschreibt einen Block binärer Daten, der an die Initialisierungsroutine in der Dekomprimierungsbibliothek übergeben wird. DB2 übergibt diese Zeichenfolge direkt vom Client an den Server, daher muss die Handhabung der Bytefolgeumkehrung oder Codepagekonvertierung von der Dekomprimierungsbibliothek übernommen werden. Ist '@' das erste Zeichen des Datenblocks, werden die restlichen Daten von DB2 als Name einer Datei interpretiert, die sich auf dem Server befindet. DB2 ersetzt in diesem Fall den Inhalt des Datenblocks durch den Inhalt dieser Datei und übergibt diesen neuen Wert stattdessen an die Initialisierungsroutine. Die maximal zulässige Länge für die Zeichenfolge ist 1024 Byte.

-d Zeigt Informationen aus den Headern von Datenseiten von DMS-Tabellenbereichen an.

-e Extrahiert Seiten aus einem Image in eine Datei. Zum Extrahieren von Seiten benötigen Sie eine Eingabe- und eine Ausgabedatei. Die Standardeingabedatei hat den Namen `extractPage.in`. Sie können den Namen der Standardeingabedatei außer Kraft setzen, indem Sie die Umgebungsvariable DB2LISTFILE auf einen vollständigen Pfad setzen.

Die Eingabedatei hat folgendes Format:

Für SMS-Tabellenbereiche:

```
S <tabellenbereichsid> <objektid> <objekttyp> <startseite> <anzseiten>
```

Für DMS-Tabellenbereiche:

```
D <tabellenbereichsid> <objekttyp> <startseite> <anzseiten>
```

Anmerkung: Der Objekttyp ist nur beim Prüfen von DMS-Ladekopie-images erforderlich.

Für Protokolldateien:

```
L <protokollnummer> <anfangsposition> <anzseiten>
```

Für andere Daten (z. B. Anfangsdaten):

```
O <objekttyp> <anfangsposition> <anzbyte>
```

Die Standardausgabedatei hat den Namen `extractPage.out`. Sie können den Namen der Standardausgabedatei außer Kraft setzen, indem Sie die Umgebungsvariable DB2EXTRACTFILE auf einen vollständigen Pfad setzen.

-h Zeigt Datenträgerheaderdaten einschließlich des Imagenamens und -pfades an, die vom Wiederherstellungsdienstprogramm erwartet werden.

-H Zeigt dieselben Informationen an wie -h, liest jedoch nur die 4 KB Datenträgerheaderdaten vom Anfang des Images. Das Image wird nicht auf Gültigkeit geprüft.

Anmerkung: Diese Option kann nicht in Kombination mit anderen Optionen verwendet werden.

-l Zeigt die Protokolldateiheaderdaten (LFH - Log File Header) und Spiegelprotokolldateiheaderdaten (MFH - Mirror Log File Header) an.

-n Fordert zum Anhängen eines Bandes auf. Nimmt ein Band pro Einheit an.

-o Zeigt Details aus den Objektheadern an.

-p Zeigt die Anzahl Seiten der einzelnen Objekttypen an.

-t Zeigt Details des Tabellenbereichs, einschließlich Behälterinformationen, für die Tabellenbereiche im Image an.

-T Zeigt die gleichen Informationen wie -t an, das Image wird jedoch nicht auf Gültigkeit überprüft.

Anmerkung: Diese Option kann nicht in Kombination mit anderen Optionen verwendet werden.

dateiname

Der Name der Sicherungsimagedatei. Mehrere Dateien können zur gleichen Zeit überprüft werden.

Anmerkungen:

1. Wenn die vollständige Sicherung aus mehreren Objekten besteht, ist die Gültigkeitsprüfung nur erfolgreich, wenn mit **db2ckbkp** alle Objekte zur gleichen Zeit geprüft werden.
2. Bei der Überprüfung mehrerer Teile eines Images müssen Sie das erste Sicherungsimageobjekt (.001) als erstes angeben.

Beispiele:

Beispiel 1 (auf UNIX-Plattformen)

```
db2ckbkp SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.001
SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.002
SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.003
[1] Buffers processed:  ##
[2] Buffers processed:  ##
[3] Buffers processed:  ##
Image Verification Complete - successful.
```

Beispiel 2 (auf Windows-Plattformen)

```
db2ckbkp SAMPLE.0\krodger\NODE0000\CATN0000\19990817\150714.001
SAMPLE.0\krodger\NODE0000\CATN0000\19990817\150714.002
SAMPLE.0\krodger\NODE0000\CATN0000\19990817\150714.003
[1] Buffers processed:  ##
[2] Buffers processed:  ##
[3] Buffers processed:  ##
Image Verification Complete - successful.
```

Beispiel 3

```
   db2ckbkp -h SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001

====================
MEDIA HEADER REACHED:
====================
        Server Database Name           -- SAMPLE2
        Server Database Alias          -- SAMPLE2
        Client Database Alias          -- SAMPLE2
        Timestamp                      -- 19990818122909
        Database Partition Number      -- 0
        Instance                       -- krodger
        Sequence Number                -- 1
        Release ID                     -- 900
        Database Seed                  -- 65E0B395
        DB Comment's Codepage (Volume) -- 0
        DB Comment (Volume)            --
        DB Comment's Codepage (System) -- 0
        DB Comment (System)            --
        Authentication Value           -- 255
        Backup Mode                    -- 0
        Include Logs                   -- 0
        Compression                    -- 0
        Backup Type                    -- 0
        Backup Gran.                   -- 0
        Status Flags                   -- 11
        System Cats inc                -- 1
        Catalog Database Partition No. -- 0
        DB Codeset                     -- ISO8859-1
        DB Territory                   --
        LogID                          -- 1074717952
        LogPath                        -- /home/krodger/krodger/NODE0000/
                                          SQL00001/SQLOGDIR
        Backup Buffer Size             -- 4194304
        Number of Sessions             -- 1
        Platform                       -- 0

The proper image file name would be:
SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001
[1] Buffers processed:  ####
Image Verification Complete - successful.
```

Hinweise:

1. Wenn ein Sicherungsimage in mehreren Sitzungen erstellt wurde, können Sie mit **db2ckbkp** alle Dateien zur gleichen Zeit untersuchen. Die Benutzer müssen dafür sorgen, dass die Sitzung mit der Folgenummer 001 als erste Datei angegeben ist.
2. Außerdem können Sie mit diesem Dienstprogramm Sicherungsimages prüfen, die auf Band gespeichert sind (mit Ausnahme von Images, die mit variabler Blockgröße erstellt wurden). Dazu bereiten Sie das Band so vor wie für eine Wiederherstellungsoperation und rufen anschließend das Dienstprogramm unter Angabe des Bandeinheitennamens auf. Auf UNIX-System geben Sie z. B. Folgendes an:

   ```
   db2ckbkp -h /dev/rmt0
   ```

   Unter Windows:

   ```
   db2ckbkp -d \\.\tape1
   ```
3. Wenn sich das Image auf einer Bandeinheit befindet, geben Sie den Pfad der Bandeinheit an. Sie werden aufgefordert, sicherzustellen, dass das Band angehängt ist, außer Sie haben die Option '-n' angegeben. Falls mehrere Bänder vorhanden sind, muss das erste Band im ersten gegebenen Einheitenpfad angehängt werden. (Dies ist das Band mit der Folge 001 im Header.) Wenn ein Band entdeckt wird, wird der Benutzer standardmäßig zum Anhängen des Bandes aufgefordert. Dem Benutzer stehen an der Eingabeaufforderung mehrere Optionen zur Auswahl. Die Eingabeaufforderung und die Optionen lauten wie folgt (wobei die Einheit I im Einheitenpfad /dev/rmt0 angegeben ist):

   ```
   Please mount the source media on device /dev/rmt0.
   Continue(c), terminate only this device(d), or abort this tool(t)?
   (c/d/t)
   ```

   Die Eingabeaufforderung wird für jede angegebene Einheit angezeigt, sowie jedes Mal, wenn die Einheit das Ende des Bandes erreicht.

Zugehörige Referenzen:

- „db2adutl - DB2-Objekte in TSM verwalten" auf Seite 271

db2ckrst - Reihenfolge der Images der Teilwiederherstellung überprüfen

Fragt das Datenbankprotokoll ab und generiert eine Liste von Zeitmarken für die Sicherungsimages, die für eine Teilwiederherstellung erforderlich sind. Außerdem wird eine vereinfachte Wiederherstellungssyntax für eine manuelle Teilwiederherstellung generiert.

Berechtigung:

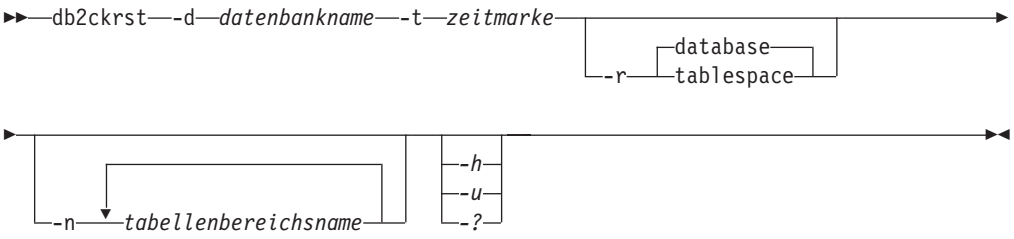
Keine

Erforderliche Verbindung:

Keine

db2ckrst - Reihenfolge der Images für Teilwiederherstellung überprüfen

Befehlssyntax:



Befehlsparameter:

-d *datenbankname*
Gibt den Aliasnamen für die Datenbank zurück, die wiederhergestellt wird.

-t *zeitmarke*
Gibt die Zeitmarke für ein Sicherungsimage zurück, das teilweise wiederhergestellt wird.

-r Gibt den Typ der Wiederherstellung an, die ausgeführt wird. Die Standardeinstellung ist `database`.

Anmerkung: Sollten Sie `tablespace` ausgewählt haben und keine Tabellenbereichsnamen angegeben sein, sucht das Dienstprogramm im Protokolleintrag des angegebenen Images und verwendet für die Wiederherstellung die aufgelisteten Tabellenbereichsnamen.

-n *tabellenbereichsname*
Gibt den Namen mindestens eines Tabellenbereichs an, der wiederhergestellt wird.

Anmerkung: Wenn Sie den Wiederherstellungstyp `database` auswählen und eine Liste von Tabellenbereichsnamen angeben, fährt das Dienstprogramm als Wiederherstellungsoperation für Tabellenbereiche fort und verwendet dabei die angegebenen Tabellenbereichsnamen.

-h/-u/-?
Zeigt Hilfetext an. Wenn Sie diese Option angeben, werden alle anderen Optionen ignoriert, und es wird lediglich ein Hilfetext angezeigt.

Beispiele:

```
db2ckrst -d mr -t 20001015193455 -r database
db2ckrst -d mr -t 20001015193455 -r tablespace
db2ckrst -d mr -t 20001015193455 -r tablespace -n tbsp1 tbsp2

   > db2 backup db mr

Sicherung erfolgreich. Die Zeitmarke für diese Sicherungsimagedatei ist:
20001016001426

   > db2 backup db mr incremental

Sicherung erfolgreich. Die Zeitmarke für diese Sicherungsimagedatei ist:
20001016001445

   > db2ckrst -d mr -t 20001016001445

Suggested restore order of images using timestamp 20001016001445 for database mr.
  ==========================================================
  db2 restore db mr incremental taken at 20001016001445
```

```
  db2 restore db mr incremental taken at 20001016001426
  db2 restore db mr incremental taken at 20001016001445
  ============================================================

   > db2ckrst -d mr -t 20001016001445 -r tablespace -n userspace1
Suggested restore order of images using timestamp 20001016001445 for database mr.
  ============================================================
     db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at
     20001016001445
     db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at
     20001016001426
     db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at
     20001016001445
  ============================================================
```

Hinweise:

Das Datenbankprotokoll muss bereits vorhanden sein, damit Sie dieses Dienstprogramm verwenden können. Wenn kein Datenbankprotokoll vorhanden ist, geben Sie für den Befehl RESTORE die Option HISTORY FILE an, bevor Sie dieses Dienstprogramm verwenden.

Wenn Sie die Option FORCE des Befehls PRUNE HISTORY verwenden, können Sie Einträge löschen, die für die automatische inkrementelle Wiederherstellung von Datenbanken erforderlich sind. Manuelle Wiederherstellungen arbeiten weiterhin korrekt. Die Verwendung dieses Befehls kann außerdem verhindern, dass das Dienstprogramm dbckrst die vollständige Kette der erforderlichen Sicherungsimages korrekt analysieren kann. Die Standardoperation des Befehls PRUNE HISTORY verhindert, dass erforderliche Einträge gelöscht werden. Es wird dringend davon abgeraten, die Option FORCE mit dem Befehl PRUNE HISTORY zu verwenden.

Dieses Dienstprogramm ist kein Ersatz für die Protokollierung Ihrer Sicherungen.

Zugehörige Tasks:

- „Wiederherstellen von Teilsicherungsimages” auf Seite 32

Zugehörige Referenzen:

- „RESTORE DATABASE” auf Seite 102
- „PRUNE HISTORY/LOGFILE” auf Seite 296

db2flsn - Protokollfolgenummer suchen

Gibt den Namen der Datei zurück, die den Protokollsatz enthält, welcher mit einer Protokollfolgenummer (LSN - Log Sequence Number) angegeben ist.

Berechtigung:

Keine

Befehlssyntax:

Befehlsparameter:

-q Gibt an, dass nur der Protokolldateiname gedruckt werden soll. Weder Fehlernachrichten noch Warnungen werden gedruckt, und den Status können Sie nur anhand des Rückkehrcodes bestimmen. Gültige Fehlercodes:
- -100 Ungültige Eingabe
- -101 Die LFH-Datei kann nicht geöffnet werden
- -102 Lesen der LFH-Datei fehlgeschlagen
- -103 Ungültiger LFH
- -104 Datenbank nicht wiederherstellbar
- -105 Protokollfolgenummer zu lang
- -500 Logischer Fehler

Weitere gültige Rückkehrcodes:
- 0 Erfolgreiche Ausführung
- 99 Warnung: Das Ergebnis basiert auf der zuletzt bekannten Protokolldateigröße.

eingabe-pfn
Eine Zeichenfolge aus zwölf Zeichen, mit der der interne, sechs Byte lange, hexadezimale Wert mit führenden Nullen dargestellt wird.

Beispiele:

```
db2flsn 000000BF0030
   Given LSN is contained in log file S0000002.LOG

db2flsn -q 000000BF0030
   S0000002.LOG

db2flsn 000000BE0030
   Warning: the result is based on the last known log file size.
   The last known log file size is 23 4K pages starting from log extent 2.

   Given LSN is contained in log file S0000001.LOG

db2flsn -q 000000BE0030
   S0000001.LOG
```

Hinweise:

Die Protokollkopfsteuerdatei `SQLOGCTL.LFH` muss sich im aktuellen Verzeichnis befinden. Da sich diese Datei im Datenbankverzeichnis befindet, können Sie das Tool vom Datenbankverzeichnis aus ausführen, oder Sie können die Steuerdatei in das Verzeichnis kopieren, von dem aus das Tool ausgeführt wird.

Das Tool verwendet den Datenbankkonfigurationsparameter *logfilsiz*. DB2 zeichnet die drei neuesten Werte für diesen Parameter und die erste Protokolldatei, die mit den jeweiligen Werten von *logfilsiz* erstellt wird, auf; dadurch kann das Tool bei Änderungen von *logfilsiz* ordnungsgemäß arbeiten. Wenn die angegebene Protokollfolgenummer vor dem ersten aufgezeichneten Wert von *logfilsiz* liegt, verwendet das Tool diesen Wert und gibt eine Warnung zurück. Sie können das Tool mit Datenbankmanagern niedrigerer Versionen als UDB Version 5.2 verwenden; in diesem Fall wird die Warnung sogar mit einem richtigen Ergebnis zurückgegeben (sofern der Wert für *logfilsiz* unverändert bleibt).

Dieses Tool können Sie nur mit wiederherstellbaren Datenbanken verwenden. Eine Datenbank ist wiederherstellbar, wenn sie so konfiguriert ist, dass der Konfigurationsparameter *logarchmeth1* oder *logarchmeth2* auf einen anderen Wert als `OFF` gesetzt ist.

db2inidb - Spiegeldatenbank initialisieren

Initialisiert eine Spiegeldatenbank in einer Umgebung mit geteilten Spiegeldatenbanken. Die Spiegeldatenbank kann als Klon der Primärdatenbank initialisiert, in den Status *Aktualisierende Wiederherstellung anstehend* versetzt oder als Sicherungsimage zur Wiederherstellung der Primärdatenbank verwendet werden. Dieser Befehl kann nur für eine geteilte Spiegeldatenbank ausgeführt werden, und die geteilte Spiegeldatenbank kann erst nach Ausführung dieses Befehls verwendet werden.

Berechtigung:

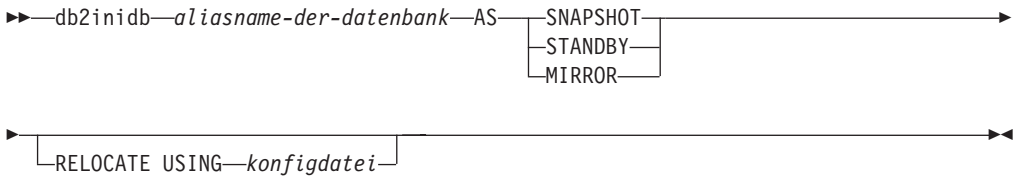
Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*

Erforderliche Verbindung:

Keine

Befehlssyntax:



Befehlsparameter:

aliasname-der-datenbank
Gibt den Aliasnamen der zu initialisierenden Datenbank an.

SNAPSHOT
Gibt an, dass die gespiegelte Datenbank als Klon der primären Datenbank initialisiert wird.

STANDBY
Gibt an, dass die Datenbank in den Status *Aktualisierende Wiederherstellung anstehend* versetzt wird.

Anmerkung: Neue Protokolle aus der primären Datenbank können abgerufen und auf die Bereitschaftsdatenbank angewendet werden. Die Bereitschaftsdatenbank können Sie anschließend bei einem Ausfall anstelle der primären Datenbank verwenden.

MIRROR
Gibt an, dass die gespiegelte Datenbank als Sicherungsimage verwendet werden soll, das zur Wiederherstellung der Primärdatenbank verwendet werden kann.

RELOCATE USING *konfigdatei*
Gibt an, dass die Datenbankdateien anhand der in der *konfigdatei* aufgelisteten Informationen verlagert werden sollen, bevor die Datenbank als

Momentaufnahme-, Bereitschafts- oder Spiegeldatenbank initialisiert wird. Das Format der *konfigdatei* ist beschrieben in "db2relocatedb - Relocate Database Command".

Hinweise:

In einer Umgebung mit partitionierten Datenbanken muss der Befehl **db2inidb** jeweils in allen Partitionen ausgeführt werden, bevor die geteilten Spiegeldatenbanken dieser Partitionen verwendet werden können. Der Befehl **db2inidb** kann mit Hilfe des Befehls **db2_all** in allen Partitionen gleichzeitig ausgeführt werden.

Wenn Sie jedoch die Option RELOCATE USING verwenden, können Sie den Befehl **db2inidb** nicht mit dem Befehl **db2_all** auf allen Partitionen gleichzeitig ausführen. Für jede Partition muss eine eigene Konfigurationsdatei angegeben werden, die den Wert NODENUM der zu ändernden Partition enthält. Wenn z. B. der Name der Datenbank geändert wird, sind alle Partitionen betroffen, und der Befehl `db2relocatedb` muss auf jeder Partition mit einer eigenen Konfigurationsdatei ausgeführt werden. Wenn Behälter versetzt werden, die zu einer einzelnen Datenbankpartition gehören, muss der Befehl **db2relocatedb** nur einmal auf dieser Partition ausgeführt werden.

Wenn der Parameter RELOCATE USING *konfigdatei* angegeben und die Datenbank erfolgreich verlagert wurde, wird die angegebene *konfigdatei* in das Datenbankverzeichnis kopiert und in `db2path.cfg` umbenannt. Während einer nachfolgenden Wiederherstellung nach einem Systemabsturz oder einer aktualisierenden Wiederherstellung wird diese Konfigurationsdatei bei der Verarbeitung von Protokolldateien zum Umbenennen von Behälterpfaden verwendet.

Wird eine Klondatenbank initialisiert, wird die angegebene *konfigdatei* nach Beendigung einer Wiederherstellung nach einem Systemabsturz automatisch aus dem Datenbankverzeichnis entfernt.

Wird eine Bereitschaftsdatenbank oder eine Spiegeldatenbank initialisiert, wird die angegebene *konfigdatei* nach Beendigung oder Abbruch einer aktualisierenden Wiederherstellung automatisch aus dem Datenbankverzeichnis entfernt. Neue Behälterpfade können nach Ausführung des Befehls `db2inidb` der Datei `db2path.cfg` hinzugefügt werden. Dies wäre erforderlich, wenn für die Originaldatenbank CREATE- oder ALTER TABLESPACE-Operationen ausgeführt werden und für die Bereitschaftsdatenbank andere Pfade verwendet werden müssen.

Zugehörige Tasks:

- „Verwenden einer geteilten Spiegeldatenbank zum Klonen einer Datenbank" auf Seite 192
- „Verwenden einer geteilten Spiegeldatenbank als Bereitschaftsdatenbank" auf Seite 194
- „Verwenden einer geteilten Spiegeldatenbank als Sicherungsimage" auf Seite 195

Zugehörige Referenzen:

- „db2relocatedb - Relocate Database Command" im Handbuch *Command Reference*
- „Beschreibungen der Befehle rah und db2_all" im Handbuch *Systemverwaltung: Implementierung*

db2mscs - Windows-Dienstprogramm zur Funktionsübernahme (Failover Utility) einrichten

Erstellt die Infrastruktur für die Unterstützung der DB2-Funktionsübernahme unter Windows unter Verwendung von Microsoft Cluster Server (MSCS). Mit diesem Dienstprogramm können Sie die Funktionsübernahme sowohl in Umgebungen mit einzelnen Partitionen als auch in Umgebungen mit partitionierten Datenbanken aktivieren.

Berechtigung:

Der Benutzer muss sich an einem Domänenbenutzerkonto angemeldet haben, das zur Gruppe der Administratoren der einzelnen Maschinen im MSCS-Cluster gehört.

Befehlssyntax:

Befehlsparameter:

-f:eingabedatei
Gibt die Eingabedatei DB2MSCS.CFG für das MSCS-Dienstprogramm an. Wenn dieser Parameter nicht angegeben wird, liest das Dienstprogramm DB2MSCS die Datei DB2MSCS.CFG, die sich im aktuellen Verzeichnis befindet.

-u:exemplarname
Diese Option ermöglich das Widerrufen der db2mscs-Operation und das Zurückversetzen des Exemplars in das durch exemplarname angegebene Nicht-MSCS-Exemplar.

Hinweise:

Das Dienstprogramm DB2MSCS ist ein Standalone-Befehlszeilendienstprogramm, mit dem ein Nicht-MSCS-Exemplar in ein MSCS-Exemplar umgewandelt werden kann. Das Dienstprogramm erstellt alle MSCS-Gruppen, Ressourcen und Ressourcenabhängigkeiten. Es kopiert ebenfalls alle in der Windows-Registrierdatenbank enthaltenen DB2-Informationen in den Clusterabschnitt der Registrierungsdatenbank und versetzt das Exemplarverzeichnis auf eine gemeinsam benutzte Clusterplatte. Das Dienstprogramm DB2MSCS verwendet als Eingabe eine vom Benutzer bereitgestellte Konfigurationsdatei, in der angegeben ist, wie der Cluster aufgebaut werden soll. Die Datei DB2MSCS.CFG ist eine ASCII-Textdatei, die Parameter enthält, die vom Dienstprogramm DB2MSCS gelesen werden können. Geben Sie jeden Eingabeparameter in einer eigenen Zeile an, und verwenden Sie dabei das folgende Format: PARAMETERSCHLÜSSELWORT=parameterwert. Beispiel:

```
CLUSTER_NAME=FINANCE
GROUP_NAME=DB2 Group
IP_ADDRESS=9.21.22.89
```

Zwei Beispielkonfigurationsdateien finden Sie im Unterverzeichnis CFG des DB2-Installationsverzeichnisses. Die erste (DB2MSCS.EE) ist ein Beispiel für Umgebungen mit einer einzelnen Datenbankpartition. Die zweite Konfigurationsdatei (DB2MSCS.EEE) ist ein Beispiel für Umgebungen mit partitionierten Datenbanken.

Die Parameter für die Datei DB2MSCS.CFG lauten wie folgt:

DB2_INSTANCE
Der Name des DB2-Exemplars. Dieser Parameter besitzt einen globalen Gültigkeitsbereich und wird in der Datei DB2MSCS.CFG nur einmal angegeben.

DAS_INSTANCE
Der Name des DB2-Verwaltungsserverexemplars. Geben Sie diesen Parameter an, um den DB2-Verwaltungsserver für die Ausführung in der MSCS-Umgebung zu migrieren. Dieser Parameter besitzt einen globalen Gültigkeitsbereich und wird in der Datei DB2MSCS.CFG nur einmal angegeben.

CLUSTER_NAME
Der Name des MSCS-Clusters. Alle Ressourcen, die nach dieser Zeile angegeben werden, werden in diesem Cluster erstellt, bis eine weitere Zeile mit dem Parameter CLUSTER_NAME angegeben wird.

DB2_LOGON_USERNAME
Der Benutzername des Domänenkontos für den DB2-Service (z. B. *domain\user*). Dieser Parameter besitzt einen globalen Gültigkeitsbereich und wird in der Datei DB2MSCS.CFG nur einmal angegeben.

DB2_LOGON_PASSWORD
Das Kennwort des Domänenkontos für den DB2-Service. Dieser Parameter besitzt einen globalen Gültigkeitsbereich und wird in der Datei DB2MSCS.CFG nur einmal angegeben.

GROUP_NAME
Der Name der MSCS-Gruppe. Wenn dieser Parameter angegeben wird und die MSCS-Gruppe nicht vorhanden ist, wird eine neue Gruppe erstellt. Wenn die Gruppe bereits vorhanden ist, wird sie als Zielgruppe verwendet. Alle MSCS-Ressourcen, die nach diesem Parameter angegeben werden, werden in dieser Gruppe erstellt oder in diese Gruppe versetzt, bis eine weitere Zeile mit dem Parameter GROUP_NAME angegeben wird. Geben Sie diesen Parameter einmal für jede Gruppe an.

DB2_NODE
Die Partitionsnummer des Datenbankpartitionsservers (oder der Datenbankpartition), die in die aktuelle MSCS-Gruppe aufgenommen werden soll. Wenn mehrere logische Datenbankpartitionen auf derselben Maschine vorhanden sind, benötigt jede Datenbankpartition einen eigenen Parameter DB2_NODE. Geben Sie diesen Parameter nach dem Parameter GROUP_NAME an, so dass die DB2-Ressourcen in der richtigen MSCS-Gruppe erstellt werden. Dieser Parameter ist für Datenbanksysteme mit mehreren Partitionen erforderlich.

IP_NAME
Der Name der Ressource 'IP-Adresse'. Der Wert für IP_NAME kann willkürlich gewählt sein, er muss jedoch im Cluster eindeutig sein. Wenn dieser Parameter angegeben wird, wird eine MSCS-Ressource des Typs 'IP-Adresse' erstellt. Dieser Parameter wird für ferne TCP/IP-Verbindungen benötigt. Dieser Parameter ist in Umgebungen mit einer einzelnen Datenbankpartition optional. Es wird empfohlen, als Namen den Hostnamen zu verwenden, der zur IP-Adresse gehört.

IP_ADDRESS
Die TCP/IP-Adresse für die IP-Ressource, die mit dem vorstehenden Parameter IP_NAME angegeben ist. Dieser Parameter ist erforderlich, wenn der

Parameter IP_NAME angegeben wird. Dies ist eine neue IP-Adresse, die von keiner Maschine im Netzwerk verwendet wird.

IP_SUBNET
Die TCP/IP-Teilnetzmaske für die IP-Ressource, die mit dem vorstehenden Parameter IP_NAME angegeben ist. Dieser Parameter ist erforderlich, wenn der Parameter IP_NAME angegeben wird.

IP_NETWORK
Der Name des MSCS-Netzwerks, zu dem die vorstehend angegebene IP-Adresse gehört. Dieser Parameter ist optional. Wenn dieser Parameter nicht angegeben wird, wird das erste vom System erkannte MSCS-Netzwerk verwendet. Der Name des MSCS-Netzwerks muss in exakt derselben Schreibweise eingegeben werden, wie er in in der Netzwerkverzweigung in Cluster Administrator angegeben ist.

Anmerkung: Die vorstehenden vier IP-Schlüsselwörter werden zum Erstellen einer Ressource 'IP-Adresse' verwendet.

NETNAME_NAME
Der Name der Ressource 'Netzwerkname'. Geben Sie diesen Parameter an, um die Ressource 'Netzwerkname' zu erstellen. Dieser Parameter ist für Umgebungen mit einer einzelnen Datenbankpartition optional. In Umgebungen mit partitionierten Datenbanken muss dieser Parameter für die Exemplareignermaschine angegeben werden.

NETNAME_VALUE
Der Wert für die Ressource 'Netzwerkname'. Dieser Parameter muss angegeben werden, wenn der Parameter NETNAME_NAME angegeben ist.

NETNAME_DEPENDENCY
Der Name der IP-Ressource, von der die Ressource 'Netzwerkname' abhängig ist. Für jede Ressource 'Netzwerkname' muss eine Abhängigkeit zu einer Ressource 'IP-Adresse' bestehen. Dieser Parameter ist optional. Wenn dieser Parameter nicht angegeben ist, besteht für die Ressource 'Netzwerkname' eine Abhängigkeit zur ersten IP-Ressource in der Gruppe.

SERVICE_DISPLAY_NAME
Der Anzeigename der Ressource 'Generischer Service'. Geben Sie diesen Parameter an, wenn Sie eine Ressource 'Generischer Service' erstellen wollen.

SERVICE_NAME
Der Servicename der Ressource 'Generischer Service'. Dieser Parameter muss angegeben werden, wenn der Parameter SERVICE_DISPLAY_NAME angegeben ist.

SERVICE_STARTUP
Optionaler Startparameter für die Ressource 'Generischer Service'.

DISK_NAME
Der Name der Ressource 'Physische Platte', die in die aktuelle Gruppe versetzt werden soll. Geben Sie so viele Plattenressourcen wie nötig an. Die Plattenressourcen müssen bereits vorhanden sein. Wenn das Dienstprogramm DB2MSCS das DB2-Exemplar für Funktionsübernahmeunterstützung konfiguriert, wird das Exemplarverzeichnis auf die erste MSCS-Platte in der Gruppe kopiert. Soll eine andere MSCS-Platte für das Exemplarverzeichnis angegeben werden, verwenden Sie den Parameter INSTPROF_DISK. Der Plattenname sollte in exakt derselben Schreibweise wie in Cluster Administrator eingegeben werden.

INSTPROF_DISK
Ein optionaler Parameter zur Angabe einer MSCS-Platte, die das DB2-Exemplarverzeichnis aufnehmen soll. Wird dieser Parameter nicht angegeben, verwendet das Dienstprogramm DB2MSCS die erste Platte, die zur selben Gruppe gehört.

INSTPROF_PATH
Ein optionaler Parameter, mit dem der exakte Pfad angegeben werden kann, in den das Exemplarverzeichnis kopiert wird. Dieser Parameter *muss* angegeben werden, wenn die ServerRAID-Netfinity-Plattenressource IPSHAdisks verwendet wird (z. B. INSTPROF_PATH=p:\db2profs). INSTPROF_PATH hat Vorrang vor INSTPROF_DISK, falls beide Parameter angegeben werden.

TARGET_DRVMAP_DISK
Ein optionaler Parameter zur Angabe der MSCS-Zielplatte für die Zuordnung von Datenbanklaufwerken in Datenbanksystemen mit mehreren Partitionen. Dieser Parameter gibt das Laufwerk an, auf dem die Datenbank erstellt wird, indem er es von dem Laufwerk zuordnet, das im Befehl zur Erstellung der Datenbank angegeben ist. Wenn dieser Parameter nicht angegeben wird, muss die Zuordnung von Datenbanklaufwerken manuell mit Hilfe des Dienstprogramms DB2DRVMP registriert werden.

DB2_FALLBACK
Ein optionaler Parameter, mit dem gesteuert werden kann, ob die Anwendungen zwangsweise abgemeldet werden sollen, wenn die DB2-Ressource in den Offlinestatus versetzt wird. Falls dieser Parameter nicht angegeben wird, wird für DB2_FALLBACK die Einstellung YES verwendet. Wenn Sie nicht wollen, dass die Anwendungen zwangsweise abgemeldet werden, setzen Sie DB2_FALLBACK auf NO.

db2rfpen - Auf Status 'Aktualisierende Wiederherstellung anstehend' zurücksetzen

Versetzt eine Datenbank in den Status *Aktualisierende Wiederherstellung anstehend*. Wenn Sie HADR (High Availability Disaster Recovery) verwenden, wird die Datenbank auf den Status einer Standarddatenbank zurückgesetzt.

Berechtigung:

Keine

Erforderliche Verbindung:

Keine

Befehlssyntax:

Befehlsparameter:

aliasname-der-datenbank
Gibt den Namen der Datenbank an, die in den Status *Aktualisierende Wiederherstellung anstehend* versetzt werden soll. Wenn Sie HADR (High Availability Disaster Recovery) verwenden, wird die Datenbank auf den Status einer Standarddatenbank zurückgesetzt.

-log protokolldateipfad
Gibt den Protokolldateipfad an.

Zugehörige Konzepte:

- „HADR (High Availability Disaster Recovery) - Übersicht" auf Seite 201

CLP-Befehle

ARCHIVE LOG

Schließt für eine wiederherstellbare Datenbank die aktive Protokolldatei und schneidet sie ab.

Berechtigung:

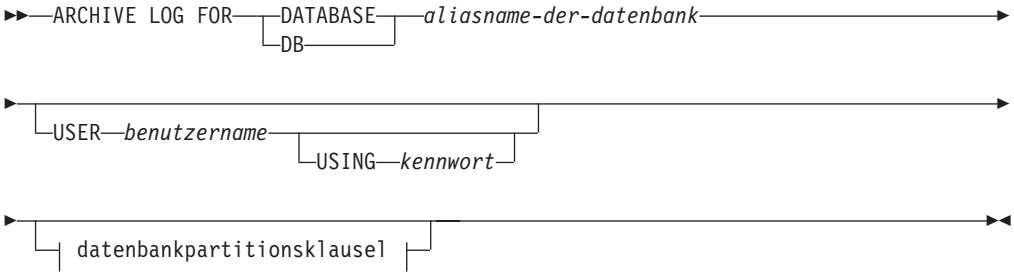
Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

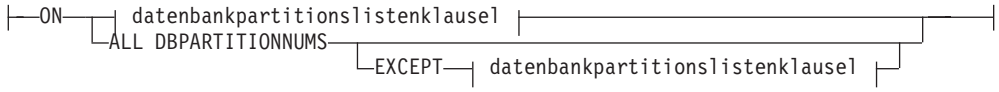
Erforderliche Verbindung:

Keine. Dieser Befehl stellt für die Dauer der Befehlsausführung eine Datenbankverbindung her.

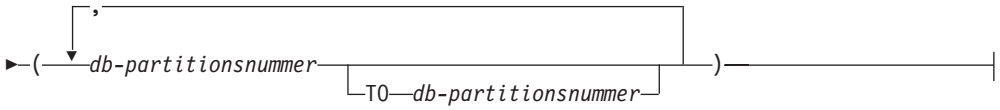
Befehlssyntax:



Datenbankpartitionsklausel:



Datenbankpartitionslistenklausel:



Befehlsparameter:

DATABASE aliasname-der-datenbank
Gibt den Aliasnamen der Datenbank an, deren aktives Protokoll archiviert werden soll.

USER benutzername
Gibt den Benutzernamen an, unter dem die Herstellung einer Verbindung versucht wird.

USING kennwort
Gibt das Kennwort an, mit dem Sie den Benutzernamen authentifizieren.

ON ALL DBPARTITIONNUMS
Gibt an, dass der Befehl für alle in der Datei `db2nodes.cfg` angegebenen Datenbankpartitionen ausgeführt werden soll. Dies ist die Standardeinstellung, falls keine Datenbankpartitionsnummernklausel angegeben ist.

EXCEPT
Gibt an, dass der Befehl für alle in der Datei `db2nodes.cfg` angegebenen Datenbankpartitionen ausgeführt werden soll, mit Ausnahme der Partitionen, die in der Datenbankpartitionsnummerliste aufgeführt sind.

ON DBPARTITIONNUM/ON DBPARTITIONNUMS
Gibt an, dass die Protokolle für die angegebene Datenbank in einer Gruppe von Datenbankpartitionen archiviert werden sollen.

db-partitionsnummer
Gibt eine Datenbankpartitionsnummer in der Datenbankpartitionsnummernliste an.

TO db-partitionsnummer
Sie können dies verwenden, wenn Sie einen Bereich von Datenbankpartitionen angeben, für die die Protokolle archiviert werden müssen. Alle Datenbankpartitionen ab der ersten angegebenen Datenbankpartitionsnummer bis einschließlich der zweiten angegebenen Datenbankpartitionsnummer werden in die Datenbankpartitionsnummerliste aufgenommen.

Hinweise:

Diesen Befehl können Sie verwenden, um einen vollständigen Satz von Protokolldateien bis zu einem bekannten Punkt aufzuzeichnen. Die Protokolldateien können Sie anschließend zur Aktualisierung einer Bereitschaftsdatenbank verwenden.

Dieser Befehl kann nur dann ausgeführt werden, wenn die aufrufende Anwendung oder Shell keine Datenbankverbindung zur angegebenen Datenbank hat. Dies verhindert, dass der Benutzer den Befehl mit nicht festgeschriebenen Transaktionen ausführt. Der Befehl ARCHIVE LOG schreibt die unvollständigen Transaktionen des Benutzers nicht zwangsläufig fest. Wenn die aufrufende Anwendung oder Shell eine Datenbankverbindung zur angegebenen Datenbank hat, wird der Befehl beendet, und es wird ein Fehler zurückgegeben. Falls während der Ausführung dieses Befehls eine andere Anwendung Transaktionen mit der angegebenen Datenbank verarbeitet, tritt eine leichte Verschlechterung der Leistung auf, da der Befehl den Protokollpuffer auf Platte schreibt. Alle Transaktionen, die Protokollsätze in den Puffer schreiben wollen, müssen warten, bis dieser Schreibvorgang beendet ist.

Wenn dieser Befehl in einer Umgebung mit partitionierten Datenbanken verwendet wird, kann mit Hilfe einer Datenbankpartitionsnummernklausel eine Untergruppe von Datenbankpartitionen angegeben werden. Wird keine Datenbankpartitionsnummernklausel angegeben, wird standardmäßig der Befehl beendet und das aktive Protokoll auf allen Datenbankpartitionen archiviert.

Bei der Verwendung dieses Befehls wird ein Teil des Speicherbereichs für die aktiven Protokolldateien belegt, da die aktive Protokolldatei abgeschnitten wird. Der Speicherbereich für die aktiven Protokolldateien erhält seine ursprüngliche Größe zurück, sobald das abgeschnittene Protokoll inaktiv wird. Eine häufige Verwendung dieses Befehls kann zu einer drastischen Reduzierung des für Transaktionen verfügbaren Speicherbereichs für aktive Protokolle führen.

Kompatibilität:

Zur Kompatibilität mit älteren Versionen als Version 8 ist Folgendes möglich:

- Das Schlüsselwort NODE kann DBPARTITIONNUM ersetzen.
- Das Schlüsselwort NODES kann DBPARTITIONNUMS ersetzen.

INITIALIZE TAPE

Bei der Ausführung auf Betriebssystemen, die auf Windows NT basieren, unterstützt DB2 Sicherungs- und Wiederherstellungsoperationen auf Datenstrombandeinheiten. Verwenden Sie diesen Befehl, um ein Band zu initialisieren.

Berechtigung:

Keine

Erforderliche Verbindung:

Keine

Befehlssyntax:

Befehlsparameter:

ON einheit
Gibt einen gültigen Bandeinheitennamen an. Der Standardwert ist `\\.\TAPE0`.

USING blockgröße
Gibt die Blockgröße für die Einheit in Byte an. Die Einheit wird mit der angegebenen Blockgröße initialisiert, sofern der Wert sich innerhalb des unterstützten Bereichs für Blockgrößen der jeweiligen Einheit befindet.

Anmerkung: Die Puffergröße, die für den Befehl BACKUP DATABASE und für RESTORE DATABASE angegeben wird, muss durch die hier angegebene Blockgröße teilbar sein.

Wenn für diesen Parameter kein Wert angegeben ist, wird die Einheit für die Verwendung dieser Standardblockgröße initialisiert. Wenn der Wert null angegeben ist, wird die Einheit für die Verwendung einer variablen

Blockgröße initialisiert; sollte die Einheit den Modus für variable Blockgrößen nicht unterstützen, wird ein Fehler zurückgegeben.

Zugehörige Referenzen:

- „BACKUP DATABASE" auf Seite 79
- „RESTORE DATABASE" auf Seite 102
- „REWIND TAPE" auf Seite 298
- „SET TAPE POSITION" auf Seite 298

LIST HISTORY

Listet Einträge in der Protokolldatei auf. Die Protokolldatei enthält Aufzeichnungen zu Wiederherstellungs- und Verwaltungsereignissen. Zu Wiederherstellungsereignissen gehören die Gesamtsicherung auf Datenbank- und Tabellenbereichsebene, die Teilsicherung, die Wiederherstellung und die aktualisierende Wiederherstellung. Weitere protokollierte Ereignisse sind das Erstellen, Ändern, Löschen oder Umbenennen eines Tabellenbereichs, die Reorganisation und das Löschen der Tabelle sowie das Laden.

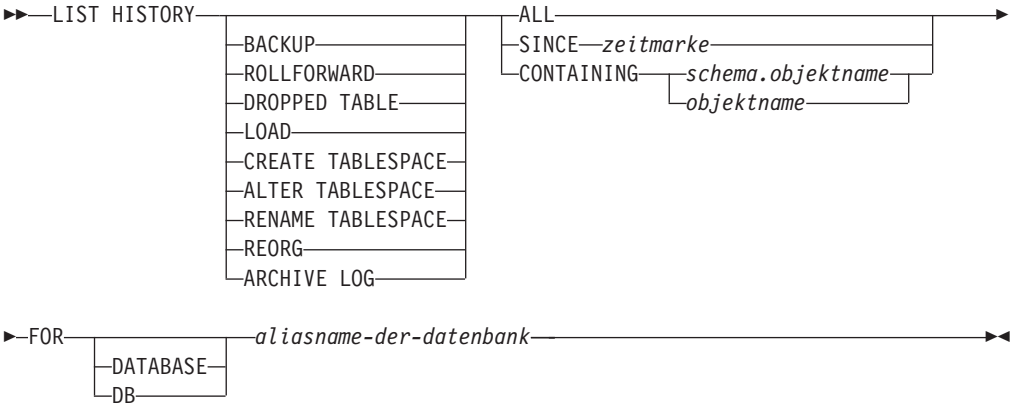
Berechtigung:

Keine

Erforderliche Verbindung:

Exemplar. Sie müssen eine Verbindung zu jeder fernen Datenbank herstellen, um diesen Befehl für die betreffende Datenbank ausführen zu können. Für lokale Datenbanken ist eine explizite Verbindung nicht erforderlich.

Befehlssyntax:



Befehlsparameter:

HISTORY
Listet alle Ereignisse auf, die momentan in der Protokolldatei aufgezeichnet sind.

BACKUP
Listet Sicherungs- und Wiederherstellungsoperationen auf.

ROLLFORWARD
Listet aktualisierende Wiederherstellungsoperationen auf.

DROPPED TABLE
Listet gelöschte Tabellensätze auf. Ein gelöschter Tabellensatz wird nur dann erstellt, wenn die Tabelle gelöscht wird und im Tabellenbereich, der die betreffende Tabelle enthält, die Option DROPPED TABLE RECOVERY zum Wiederherstellen gelöschter Tabellen aktiviert ist.

LOAD
Listet Ladeoperationen auf.

CREATE TABLESPACE
Listet auf den Tabellenbereich bezogene Erstellungs- und Löschoperationen auf.

RENAME TABLESPACE
Listet auf den Tabellenbereich bezogene Umbenennungsoperationen auf.

REORG
Listet Reorganisierungsoperationen auf.

ALTER TABLESPACE
Listet Änderungsoperationen des Tabellenbereichs auf.

ARCHIVE LOG
Listet Archivprotokolldateioperationen und die archivierten Protokolle auf.

ALL Listet alle Einträge des angegebenen Typs in der Protokolldatei auf.

SINCE zeitmarke
Sie können eine vollständige Zeitmarke (Format *jjjjmmtthhmmss*) oder ein Präfix (mindestens *jjjj*) angeben. Alle Einträge mit Zeitmarken größergleich der angegebenen Zeitmarke werden aufgelistet.

CONTAINING schema.objektname
Dieser qualifizierte Name kennzeichnet eine Tabelle eindeutig.

CONTAINING objektname
Dieser nicht qualifizierte Name kennzeichnet einen Tabellenbereich eindeutig.

FOR DATABASE aliasname-der-datenbank
Wird verwendet, um die Datenbank anzugeben, deren Protokolldatei aufgelistet werden soll.

Beispiele:

```
db2 list history since 19980201 for sample
db2 list history backup containing userspace1 for sample
db2 list history dropped table all for db sample
```

Hinweise:

Der von diesem Befehl generierte Bericht enthält die folgenden Symbole:

```
Operation

   A - Tabellenbereich erstellen
   B - Sicherung
   C - Kopie laden
   D - Gelöschte Tabelle
   F - Aktualisierend wiederherstellen
   G - Tabelle reorganisieren
   L - Laden
   N - Tabellenbereich umbenennen
   O - Tabellenbereich löschen
   Q - In den Wartemodus versetzen
```

```
   R - Wiederherstellen
   T - Tabellenbereich ändern
   U - Aus Tabelle laden
   X - Archivprotokolldatei

Typ

Archivprotokolldateitypen:
   P - Primärer Protokollpfad
   M - Sekundärer Protokollpfad (Spiegelprotokollpfad)
   F - Archivpfad für Funktionsübernahme
   1 - Primäre Protokollarchivierungsmethode
   2 - Sekundäre Protokollarchivierungsmethode

Sicherungstypen:

   F - Offline
   N - Online
   I - Inkrementell offline
   O - Inkrementell online
   D - Delta offline
   E - Delta online

Typen der aktualisierenden Wiederherstellung:

   E - Ende der Protokolle
   P - Zeitpunkt

Ladetypen:

   I - Einfügen
   R - Ersetzen

Tabellenbereichstypen:

   C - Behälter hinzufügen
   R - Neu ausgleichen

Typen des Versetzens in den Wartemodus:

   S - Wartemodus: SHARE
   U - Wartemodus: UPDATE
   X - Wartemodus: EXCLUSIVE
   Z - Wartemodus: RESET
```

PRUNE HISTORY/LOGFILE

Wird zum Löschen von Einträgen aus der Datei des Wiederherstellungsprotokolls verwendet oder zum Löschen von Protokolldateien aus dem Pfad für aktive Protokolldateien. Das Löschen von Einträgen aus der Datei des Wiederherstellungsprotokolls ist möglicherweise erforderlich, wenn die Datei sehr groß wird und der Aufbewahrungzeitraum lang ist.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Erforderliche Verbindung:

Datenbank

Befehlssyntax:

Befehlsparameter:

HISTORY zeitmarke
Gibt einen Bereich von Einträgen in der Datei des Wiederherstellungsprotokolls an, die gelöscht werden. Sie können eine vollständige Zeitmarke (Format *jjjjmmtthhmmss*) oder ein Präfix (mindestens *jjjj*) angeben. Alle Einträge mit Zeitmarken kleiner-gleich der angegebenen Zeitmarke werden aus der Datei des Wiederherstellungsprotokolls gelöscht.

WITH FORCE OPTION
Gibt an, dass die Einträge gemäß der angegebenen Zeitmarke entfernt werden, selbst wenn einige Einträge aus der neuesten Wiederherstellungsgruppe aus der Datei gelöscht werden. Eine Wiederherstellungsgruppe ist die neueste vollständige Datenbanksicherung einschließlich aller Wiederherstellungen dieses Sicherungsimages. Wenn Sie diesen Parameter nicht angeben, bleiben alle Einträge aus der Weiterleitung des Sicherungsimages im Protokoll.

AND DELETE
Gibt an, dass die zugeordneten Protokollarchive physisch gelöscht werden (auf Basis der Positionsinformationen), wenn der Protokolldateieintrag entfernt wird. Mit dieser Option kann vor allem sichergestellt werden, dass der Archivierungsspeicherbereich wieder verfügbar ist, wenn Protokollarchive nicht mehr benötigt werden.

Anmerkung: Wenn Sie Protokolle über ein Benutzerexitprogramm archivieren, können die Protokolle nicht mit dieser Option gelöscht werden.

LOGFILE PRIOR TO protokolldateiname
Gibt eine Zeichenfolge für einen Protokolldateinamen an, z. B. `S0000100.LOG`. Löscht alle Protokolldateien vor der angegebenen Protokolldatei (diese ausgeschlossen). Den Datenbankkonfigurationsparameter LOGRETAIN müssen Sie auf `RECOVERY` oder auf `CAPTURE` setzen.

Beispiele:

Zum Entfernen der Einträge für alle Wiederherstellungen, Ladevorgänge, Tabellenbereichssicherungen und vollständigen Datenbanksicherungen, die Sie vor dem 1. Dezember 1994 (einschließlich) ausgeführt haben, aus der Datei des Wiederherstellungsprotokolls, geben Sie Folgendes ein:

```
db2 prune history 199412
```

Anmerkung: 199412 wird als 19941201000000 interpretiert.

Hinweise:

Wenn Sie die Option FORCE des Befehls PRUNE HISTORY verwenden, können Sie Einträge löschen, die für die automatische Teilwiederherstellung von Datenbanken erforderlich sind. Manuelle Wiederherstellungen arbeiten weiterhin korrekt. Die Verwendung dieses Befehls kann außerdem verhindern, dass das Dienstprogramm `dbckrst` die vollständige Kette der erforderlichen Sicherungsimages korrekt analysieren kann. Wenn Sie den Befehl PRUNE HISTORY ohne die Option FORCE verwenden, wird verhindert, dass erforderliche Einträge gelöscht werden.

Das Löschen von Sicherungseinträgen aus der Protokolldatei bewirkt, dass zugehörige Dateisicherungen auf DB2 Data Links Manager-Servern gelöscht werden.

REWIND TAPE

Bei der Ausführung auf Betriebssystemen, die auf Windows NT basieren, unterstützt DB2 Sicherungs- und Wiederherstellungsoperationen auf Datenstrombandeinheiten. Verwenden Sie diesen Befehl, um ein Band zurückzuspulen.

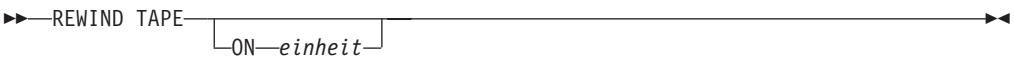
Berechtigung:

Keine

Erforderliche Verbindung:

Keine

Befehlssyntax:



Befehlsparameter:

ON einheit
Gibt einen gültigen Bandeinheitennamen an. Der Standardwert ist `\\.\TAPE0`.

Zugehörige Referenzen:

- „INITIALIZE TAPE" auf Seite 293
- „SET TAPE POSITION" auf Seite 298

SET TAPE POSITION

Bei der Ausführung auf Betriebssystemen, die auf Windows NT basieren, unterstützt DB2 Sicherungs- und Wiederherstellungsoperationen auf Datenstrombandeinheiten. Verwenden Sie diesen Befehl, um ein Band zu positionieren.

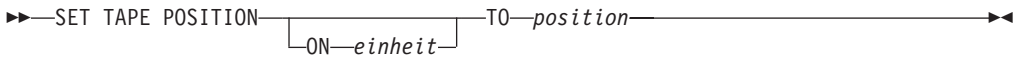
Berechtigung:

Keine

Erforderliche Verbindung:

Keine

Befehlssyntax:



Befehlsparameter:

ON einheit
Gibt einen gültigen Bandeinheitennamen an. Der Standardwert ist `\\.\TAPE0`.

TO position
Gibt die Markierung an, an der Sie das Band positionieren wollen. DB2 für Windows schreibt nach jedem Sicherungsimage eine Bandmarke. Der Wert 1 gibt die erste Position an, der Wert 2 die zweite usw. Sollte das Band an Bandmarke 1 positioniert sein, ist z. B. das Archiv 2 so positioniert, dass es wiederhergestellt wird.

Zugehörige Referenzen:

- „INITIALIZE TAPE" auf Seite 293
- „REWIND TAPE" auf Seite 298

UPDATE HISTORY FILE

Aktualisiert die Position, den Einheitentyp oder den Kommentar in einem Protokolldateieintrag.

Berechtigung:

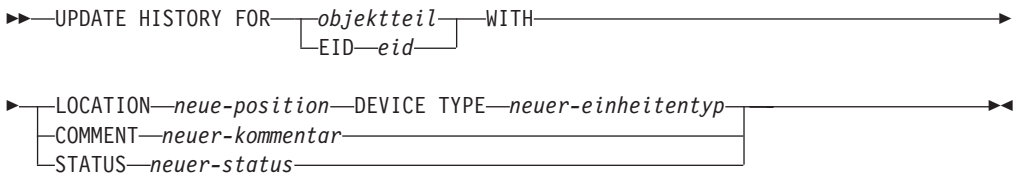
Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Erforderliche Verbindung:

Datenbank

Befehlssyntax:



Befehlsparameter:

EID *eid*
Gibt die ID des Protokolleintrags an.

FOR *objektteil*
Gibt die Kennung für das Sicherungs- oder Kopierimage an. Diese ist eine Zeitmarke mit einer optionalen Folgenummer von 001 bis 999.

LOCATION *neue-position*
Gibt die neue physische Position eines Sicherungsimages an. Die Interpretation dieses Parameters hängt vom Einheitentyp ab.

DEVICE TYPE *neuer-einheitentyp*
Gibt einen neuen Einheitentyp zur Speicherung des Sicherungsimages an. Gültige Einheitentypen:

D Platte
K Diskette
T Band
A TSM
U Benutzerexit
P Pipe
N Nulleinheit
X XBSA
Q SQL-Anweisung
O Sonstiges

COMMENT *neuer-kommentar*
Gibt einen neuen Kommentar mit der Beschreibung des Eintrags an.

STATUS *neuer-status*
Gibt einen neuen Status für einen Eintrag an. Gültige Werte:

A Markiert den Eintrag als aktiv.
I Markiert den Eintrag als inaktiv.

Beispiele:

Geben Sie Folgendes ein, um den Protokolldateieintrag für eine vollständige Datenbanksicherung am 13. April 1997 um 10.00 Uhr zu aktualisieren:

```
db2 update history for 19970413100000001 with
   location /backup/dbbackup.1 device type d
```

Hinweise:

Die Datenbankprotokolldatei dient primär zur Aufzeichnung von Informationen, die im Protokoll enthaltenen Daten werden jedoch direkt von automatischen Wiederherstellungsoperationen verwendet. Wenn bei einer Wiederherstellung die Option AUTOMATIC angegeben wird, verwendet das Wiederherstellungsdienstprogramm das Protokoll der Sicherungsimages und deren Positionen zur automatischen Wiederherstellung. Wenn die automatische Wiederherstellungsfunktion verwendet werden soll und einige Sicherungsimages seit ihrer Erstellung verlagert wurden, wird empfohlen, den Datenbankprotokollsatz für diese Images zu aktualisieren, sodass die aktuelle Position der Images wiedergegeben wird. Wenn die Position der Sicherungsimages im Datenbankprotokoll nicht aktualisiert wird, kann die automatische Wiederherstellung die Sicherungsimages nicht finden, manuelle Wiederherstellungsbefehle können jedoch weiter erfolgreich verwendet werden.

Zugehörige Referenzen:

- „PRUNE HISTORY/LOGFILE" auf Seite 296

Anhang D. Weitere APIs und zugehörige Datenstrukturen

In diesem Anhang werden wiederherstellungsbezogene APIs und ihre Datenstrukturen beschrieben, die in diesem Handbuch nicht ausführlich behandelt werden.

db2ArchiveLog - Aktive Protokolldatei archivieren

Schließt für eine wiederherstellbare Datenbank die aktive Protokolldatei und schneidet sie ab. Wenn der Benutzerexit aktiviert ist, wird eine Archivierungsanforderung abgesetzt.

Berechtigung:

Eine der Folgenden:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Erforderliche Verbindung:

Mit dieser API stellen Sie automatisch eine Verbindung zur angegebenen Datenbank her. Wenn bereits eine Verbindung zu der angegebenen Datenbank existiert, gibt die API einen Fehler zurück.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2ArchiveLog */
/* ... */
SQL_API_RC SQL_API_FN
   db2ArchiveLog (
    db2Uint32 version,
  void *pDB2ArchiveLogStruct,
    struct sqlca *pSqlca);

typedef struct
{
   char                *piDatabaseAlias;
  char                        *piUserName;
  char                        *piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE                   *piNodeList;
   db2Uint32 iOptions;
} db2ArchiveLogStruct
/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2gArchiveLog */
/* ... */
SQL_API_RC SQL_API_FN
   db2gArchiveLog (
    db2Uint32 version,
   void *pDB2ArchiveLogStruct,
    struct sqlca *pSqlca);
typedef struct
{
   db2Uint32 iAliasLen;
   db2Uint32 iUserNameLen;
   db2Uint32 iPasswordLen;
   char                       *piDatabaseAlias;
  char                        *piUserName;
  char                        *piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE          *piNodeList;
   db2Uint32 iOptions;
} db2ArchiveLogStruct
/* ... */
```

API-Parameter:

version
Eingabe. Gibt die Version und den Releasestand der Variablen an, die als zweiter Parameter, *pDB2ArchiveLogStruct*, übergeben wurde.

pDB2ArchiveLogStruct
Eingabe. Ein Zeiger auf die Struktur *db2ArchiveLogStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur *sqlca*.

iAliasLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Datenbankaliasnamens in Byte angibt.

iUserNameLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Benutzernamens in Byte angibt. Wenn kein Benutzername verwendet wird, ist dieser Parameter auf null gesetzt.

iPasswordLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Kennworts in Byte angibt. Wenn kein Kennwort verwendet wird, ist dieser Parameter auf null gesetzt.

piDatabaseAlias
Eingabe. Eine Zeichenfolge, die den Datenbankaliasnamen (wie im Systemdatenbankverzeichnis katalogisiert) enthält, für die das aktive Protokoll archiviert werden soll.

piUserName
Eingabe. Eine Zeichenfolge mit dem Benutzernamen, der beim Versuch, eine Verbindung herzustellen, verwendet wird.

piPassword
Eingabe. Eine Zeichenfolge mit dem Kennwort, das beim Versuch, eine Verbindung herzustellen, verwendet wird.

iAllNodeFlag
Nur Umgebungen mit partitionierten Datenbanken. Eingabe. Diese Markierung zeigt an, ob die Operation für alle Knoten, die in der Datei db2nodes.cfg aufgelistet sind, gelten soll. Gültige Werte:

DB2ARCHIVELOG_NODE_LIST
Auf Knoten anwenden, die in einer Knotenliste angegeben sind, die in *piNodeList* übergeben wird.

DB2ARCHIVELOG_ALL_NODES
Auf alle Knoten anwenden. *piNodeList* sollte NULL sein. Dies ist der Standardwert.

DB2ARCHIVELOG_ALL_EXCEPT
Auf alle Knoten anwenden, *außer* den Knoten, die in einer Knotenliste angegeben sind, die in *piNodeList* übergeben wird.

iNumNodes
Nur Umgebungen mit partitionierten Datenbanken. Eingabe. Gibt die Anzahl der Knoten im Bereich *piNodeList* an.

piNodeList
Nur Umgebungen mit partitionierten Datenbanken. Eingabe. Ein Zeiger auf einen Bereich von Knotennummern, für die das Archivierungsprotokollieren angewendet werden soll.

iOptions
Eingabe. Zur zukünftigen Verwendung reserviert.

Zugehörige Referenzen:

• „ARCHIVE LOG" auf Seite 291

db2HistoryCloseScan - Suche in Protokolldatei beenden

Beendet das Durchsuchen der Protokolldatei und gibt DB2-Ressourcen frei, die für die Suche benötigt wurden. Dieser API muss ein erfolgreicher Aufruf von db2HistoryOpenScan vorausgehen.

Berechtigung:

Keine

Erforderliche Verbindung:

Exemplar. Vor dem Aufruf dieser API müssen Sie nicht sqleatin aufrufen.

API-Includedatei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2HistoryCloseScan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2HistoryCloseScan (
    db2Uint32 version,
    void *piHandle,
  struct sqlca *pSqlca);
/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryCloseScan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2GenHistoryCloseScan (
    db2Uint32 version,
    void *piHandle,
  struct sqlca *pSqlca);
/* ... */
```

API-Parameter:

version
Eingabe. Gibt die Version und den Releasestand des zweiten Parameters, *piHandle*, an.

piHandle
Eingabe. Gibt einen Zeiger auf die Kennung für den Suchzugriff an, die von der API db2HistoryOpenScan zurückgegeben wurde.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

Syntax der REXX-API:

```
CLOSE RECOVERY HISTORY FILE :scanid
```

Parameter der REXX-API:

scanid Hostvariable mit der Such-ID, die von OPEN RECOVERY HISTORY FILE SCAN zurückgegeben wurde.

Hinweise:

Eine detaillierte Beschreibung der Verwendung der APIs für die Protokolldatei finden Sie in den Informationen zu db2HistoryOpenScan.

Zugehörige Referenzen:

- „db2Prune - Protokolldatei bereinigen" auf Seite 314
- „db2HistoryUpdate - Protokolldatei aktualisieren" auf Seite 311
- „db2HistoryOpenScan - Suche in Protokolldatei starten" auf Seite 307
- „db2HistoryGetEntry - Nächsten Eintrag aus der Protokolldatei abrufen" auf Seite 304
- „SQLCA" im Handbuch *Administrative API Reference*

Zugehörige Beispiele:

- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.sqC -- How to recover a database (C++)"

db2HistoryGetEntry - Nächsten Eintrag aus der Protokolldatei abrufen

Ruft den nächsten Eintrag aus der Protokolldatei ab. Dieser API muss ein erfolgreicher Aufruf von db2HistoryOpenScan vorausgehen.

Berechtigung:

Keine

db2HistoryGetEntry - Nächsten Eintrag aus der Protokolldatei abrufen

Erforderliche Verbindung:

Exemplar. Vor dem Aufruf dieser API müssen Sie nicht `sqleatin` aufrufen.

API-Includedatei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2HistoryGetEntry */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryGetEntry (
    db2Uint32 version,
    void *pDB2HistoryGetEntryStruct,
  struct sqlca *pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2HistoryGetEntryStruct;
/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryGetEntry */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryGetEntry (
    db2Uint32 version,
    void *pDB2GenHistoryGetEntryStruct,
  struct sqlca *pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2GenHistoryGetEntryStruct;
/* ... */
```

API-Parameter:

version
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2HistoryGetEntryStruct*, übergeben wird.

pDB2HistoryGetEntryStruct
Eingabe. Ein Zeiger auf die Struktur *db2HistoryGetEntryStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

iHandle
Eingabe. Enthält die Kennung für den Suchzugriff, die von der API db2HistoryOpenScan zurückgegeben wurde.

iCallerAction
Eingabe. Gibt den Typ der auszuführenden Aktion an. Gültige Werte (in `db2ApiDf.h` definiert):

DB2HISTORY_GET_ENTRY
Den nächsten Eintrag abrufen, jedoch ohne jegliche Befehlsdaten.

DB2HISTORY_GET_DDL
Nur die Befehlsdaten aus dem vorherigen Abruf abrufen.

DB2HISTORY_GET_ALL
Den nächsten Eintrag mit allen Daten abrufen.

pioHistData
Eingabe. Ein Zeiger auf die Struktur *db2HistData*.

Syntax der REXX-API:

```
GET RECOVERY HISTORY FILE ENTRY :scanid [USING :value]
```

Parameter der REXX-API:

scanid Hostvariable mit der Such-ID, die von OPEN RECOVERY HISTORY FILE SCAN zurückgegeben wurde.

value Eine zusammengesetzte REXX-Hostvariable, in die Eintragsdaten der Protokolldatei zurückgegeben werden. Im Folgenden steht XXX für den Namen der Hostvariablen:

XXX.0	Anzahl der Elemente der ersten Ebene in der Variablen (immer 15)
XXX.1	Anzahl der Tabellenbereichselemente
XXX.2	Anzahl der verwendeten Tabellenbereichselemente
XXX.3	OPERATION (Typ der ausgeführten Operation)
XXX.4	OBJECT (Granularität der Operation)
XXX.5	OBJECT_PART (Zeitmarke und Folgenummer)
XXX.6	OPTYPE (Qualifikationsmerkmal der Operation)
XXX.7	DEVICE_TYPE (Typ der verwendeten Einheit)
XXX.8	FIRST_LOG (ID des ältesten Protokolls)
XXX.9	LAST_LOG (ID des aktuellen Protokolls)
XXX.10	BACKUP_ID (Kennung für die Sicherung)
XXX.11	SCHEMA (Qualifikationsmerkmal für den Tabellennamen)
XXX.12	TABLE_NAME (Name der geladenen Tabelle)
XXX.13.0	NUM_OF_TABLESPACES (Anzahl der Tabellenbereiche, die an der Sicherung oder Wiederherstellung beteiligt sind)
XXX.13.1	Name des ersten gesicherten/wiederhergestellten Tabellenbereichs
XXX.13.2	Name des zweiten gesicherten/wiederhergestellten Tabellenbereichs
XXX.13.3	und so weiter
XXX.14	LOCATION (Speicherposition der Sicherung oder Kopie)
XXX.15	COMMENT (Text zur Beschreibung des Eintrags)

Hinweise:

Die Datensätze, die zurückgegeben werden, wurden anhand der Werte ausgewählt, die beim Aufruf der API db2HistoryOpenScan angegeben wurden. Eine detaillierte Beschreibung der Verwendung der APIs für die Protokolldatei finden Sie in den Informationen zu db2HistoryOpenScan.

Zugehörige Referenzen:

- „db2Prune - Protokolldatei bereinigen" auf Seite 314
- „db2HistoryUpdate - Protokolldatei aktualisieren" auf Seite 311
- „db2HistoryOpenScan - Suche in Protokolldatei starten" auf Seite 307
- „db2HistoryCloseScan - Suche in Protokolldatei beenden" auf Seite 303
- „SQLCA" im Handbuch *Administrative API Reference*
- „db2HistData" auf Seite 326

Zugehörige Beispiele:

- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.sqC -- How to recover a database (C++)"

db2HistoryOpenScan - Suche in Protokolldatei starten

Startet das Durchsuchen einer Protokolldatei.

Berechtigung:

Keine

Erforderliche Verbindung:

Exemplar. Wenn die Datenbank als fern katalogisiert ist, rufen Sie `sqleatin` auf, bevor Sie diese API aufrufen.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2HistoryOpenScan */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryOpenScan (
    db2Uint32 version,
    void *pDB2HistoryOpenStruct,
  struct sqlca *pSqlca);

typedef struct
{
  char *piDatabaseAlias,
  char *piTimestamp,
  char *piObjectName,
  db2Uint32 oNumRows,
  db2Uint32 oMaxTbspaces,
```

```
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2HistoryOpenStruct;
/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryOpenScan */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryOpenScan (
    db2Uint32 version,
    void *pDB2GenHistoryOpenStruct,
  struct sqlca *pSqlca);

typedef struct
{
  char *piDatabaseAlias,
  char *piTimestamp,
  char *piObjectName,
  db2Uint32 oNumRows,
  db2Uint32 oMaxTbspaces,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2GenHistoryOpenStruct;
/* ... */
```

API-Parameter:

version
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2HistoryOpenStruct*, übergeben wird.

pDB2HistoryOpenStruct
Eingabe. Ein Zeiger auf die Struktur *db2HistoryOpenStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

piDatabaseAlias
Eingabe. Ein Zeiger auf eine Zeichenfolge, die den Datenbankaliasnamen enthält.

piTimestamp
Eingabe. Ein Zeiger auf eine Zeichenfolge, die die Zeitmarke zum Auswählen von Datensätzen angibt. Es werden Datensätze ausgewählt, deren Zeitmarke größer oder gleich diesem Wert ist. Wenn dieser Parameter NULL ist oder auf null zeigt, wird das Filtern der Einträge anhand einer Zeitmarke verhindert.

piObjectName
Eingabe. Ein Zeiger auf eine Zeichenfolge, die den Objektnamen zum Auswählen von Datensätzen angibt. Das Objekt kann eine Tabelle oder ein Tabellenbereich sein. Wenn es eine Tabelle ist, muss der vollständig qualifizierte Tabellenname eingegeben werden. Wenn dieser Parameter NULL ist oder auf null zeigt, wird das Filtern der Einträge anhand eines Objektnamens verhindert.

oNumRows
Ausgabe. Nach der Rückkehr von der API enthält dieser Parameter die Anzahl passender Einträge der Protokolldatei.

oMaxTbspaces
Ausgabe. Die maximale Anzahl Tabellenbereichsnamen, die in einem Protokolleintrag gespeichert werden kann.

iCallerAction
Eingabe. Gibt den Typ der auszuführenden Aktion an. Gültige Werte (in `db2ApiDf.h` definiert):

DB2HISTORY_LIST_HISTORY
Alle Ereignisse auflisten, die momentan in der Protokolldatei aufgezeichnet sind.

DB2HISTORY_LIST_BACKUP
Sicherungs- und Wiederherstellungsoperationen auflisten.

DB2HISTORY_LIST_ROLLFORWARD
Optionen für aktualisierende Wiederherstellung auflisten.

DB2HISTORY_LIST_DROPPED_TABLE
Gelöschte Tabellensätze auflisten. Das DDL-Feld zu einem Eintrag wird nicht zurückgegeben. Zum Abruf der DDL-Daten für einen Eintrag muss db2HistoryGetEntry sofort nach dem Abruf des Eintrags mit der Aktion `DB2HISTORY_GET_DDL` aufgerufen werden.

DB2HISTORY_LIST_LOAD
Ladeoperationen auflisten.

DB2HISTORY_LIST_CRT_TABLESPACE
Auf den Tabellenbereich bezogene Erstellungs- und Löschoperationen auflisten.

DB2HISTORY_LIST_REN_TABLESPACE
Auf den Tabellenbereich bezogene Umbenennungsoperationen auflisten.

DB2HISTORY_LIST_ALT_TABLESPACE
Auf den Tabellenbereich bezogene Änderungsoperationen auflisten. Das DDL-Feld zu einem Eintrag wird nicht zurückgegeben. Zum Abruf der DDL-Daten für einen Eintrag muss db2HistoryGetEntry sofort nach dem Abruf des Eintrags mit der Aktion `DB2HISTORY_GET_DDL` aufgerufen werden.

DB2HISTORY_LIST_REORG
REORGANIZE TABLE-Operationen auflisten. Dieser Wert wird derzeit nicht unterstützt.

oHandle
Ausgabe. Nach der Rückkehr von der API enthält dieser Parameter die Kennung für den Suchzugriff. Er wird nachfolgend in db2HistoryGetEntry und in db2HistoryCloseScan verwendet.

Syntax der REXX-API:

```
OPEN [BACKUP] RECOVERY HISTORY FILE FOR database_alias
[OBJECT objname] [TIMESTAMP :timestamp]
USING :value
```

Parameter der REXX-API:

database_alias
Der Aliasname der Datenbank, deren Protokolldatei aufgelistet werden soll.

objname
Gibt den Objektnamen zum Auswählen von Datensätzen an. Das Objekt

kann eine Tabelle oder ein Tabellenbereich sein. Wenn es eine Tabelle ist, muss der vollständig qualifizierte Tabellenname eingegeben werden. Wenn dieser Parameter auf NULL gesetzt ist, wird das Filtern der Einträge anhand von *objname* verhindert.

timestamp
Gibt die Zeitmarke zum Auswählen von Datensätzen an. Es werden Datensätze ausgewählt, deren Zeitmarke größer oder gleich diesem Wert ist. Wenn dieser Parameter auf NULL gesetzt ist, wird das Filtern der Einträge anhand von *timestamp* verhindert.

value Eine zusammengesetzte REXX-Hostvariable, in die Daten der Protokolldatei zurückgegeben werden. Im Folgenden steht XXX für den Namen der Hostvariablen:

XXX.0 Anzahl der Elemente in der Variablen (immer 2)

XXX.1 Bezeichner (Kennung) für den späteren Suchzugriff

XXX.2 Anzahl der passenden Protokolldateieinträge.

Hinweise:

Die Kombination aus Zeitmarke, Objektname und auszuführender Aktion können Sie zum Filtern von Datensätzen verwenden. Nur Datensätze, die alle angegebenen Filter passieren, werden zurückgegeben.

Die Auswirkung des Filterns des Objektnamens hängt vom angegebenen Wert ab:

- Wenn Sie eine Tabelle angeben, werden Datensätze für Ladeoperationen zurückgegeben, da diese in der Protokolldatei die einzigen Daten für Tabellen sind.
- Wenn Sie einen Tabellenbereich angeben, werden Datensätze für Sicherungs-, Wiederherstellungs- und Ladeoperationen für den Tabellenbereich zurückgegeben.

Anmerkung: Zur Rückgabe von Datensätzen für Tabellen müssen diese als *schema.tabellenname* angegeben sein. Wenn Sie *tabellenname* angeben, erhalten Sie nur Datensätze für Tabellenbereiche.

Maximal acht Protokolldateisuchen sind pro Prozess zulässig.

Zum Auflisten jedes Eintrags in der Protokolldatei führt eine typische Anwendung die folgenden Schritte aus:

1. db2HistoryOpenScan aufrufen, wodurch *oNumRows* zurückgegeben wird.
2. Eine Struktur *db2HistData* mit Speicherbereich für *n oTablespace*-Felder zuordnen, wobei *n* für eine willkürliche Zahl steht.
3. Das Feld *iDB2NumTablespace* der Struktur *db2HistData* auf *n* setzen.
4. Führen Sie in einer Schleife Folgendes aus:
   - Rufen Sie db2HistoryGetEntry für den Abruf aus der Protokolldatei auf.
   - Wenn db2HistoryGetEntry den SQLCODE-Wert `SQL_RC_OK` zurückgibt, verwenden Sie das Feld *sqld* der Struktur *db2HistData*, um die Anzahl der zurückgegebenen Tabellenbereichseinträge zu bestimmen.
   - Wenn db2HistoryGetEntry den SQLCODE-Wert `SQLUH_SQLUHINFO_VARS_WARNING` zurückgibt, wurde nicht genügend Speicherbereich für alle Tabellenbereiche zugeordnet, die DB2 zurückzugeben versucht; geben Sie *db2HistData* frei, ordnen Sie die Struktur erneut mit ausrei-

chend Speicherbereich für *oDB2UsedTablespace* Tabellenbereichseinträge zu, und setzen Sie *iDB2NumTablespace* auf *oDB2UsedTablespace*.
- Wenn db2HistoryGetEntry den SQLCODE-Wert `SQLE_RC_NOMORE` zurückgibt, wurden alle Einträge aus der Protokolldatei abgerufen.
- Alle anderen SQLCODE-Werte weisen auf ein Problem hin.

5. Sobald alle Daten abgerufen worden sind, rufen Sie db2HistoryCloseScan auf, um Ressourcen freizugeben, die durch den Aufruf von db2HistoryOpenScan zugeordnet wurden.

Das Makro SQLUHINFOSIZE(*n*), das in `sqlutil.h` definiert ist, dient zur Bestimmung, wie viel Speicher für eine Struktur *db2HistData* mit Speicherbereich für *n oTablespace*-Felder erforderlich ist.

Zugehörige Referenzen:
- „db2Prune - Protokolldatei bereinigen" auf Seite 314
- „db2HistoryUpdate - Protokolldatei aktualisieren" auf Seite 311
- „db2HistoryGetEntry - Nächsten Eintrag aus der Protokolldatei abrufen" auf Seite 304
- „db2HistoryCloseScan - Suche in Protokolldatei beenden" auf Seite 303
- „SQLCA" im Handbuch *Administrative API Reference*

Zugehörige Beispiele:
- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.sqC -- How to recover a database (C++)"

db2HistoryUpdate - Protokolldatei aktualisieren

Aktualisiert die Position, den Einheitentyp oder den Kommentar in einem Protokolldateieintrag.

Berechtigung:

Eine der folgenden Berechtigungen:
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Erforderliche Verbindung:

Datenbank. Zum Aktualisieren von Einträgen in der Protokolldatei für eine andere Datenbank als die Standarddatenbank müssen Sie eine Verbindung zur Datenbank herstellen, bevor Sie diese API aufrufen.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2HistoryUpdate */
/* ... */
```

```
SQL_API_RC SQL_API_FN
  db2HistoryUpdate (
    db2Uint32 version,
    void *pDB2HistoryUpdateStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2HistoryUpdateStruct
{
   char          *piNewLocation;
   char          *piNewDeviceType;
   char          *piNewComment;
   char          *piNewStatus;
   db2HistoryEID iEID;
} db2HistoryUpdateStruct;

/* Structure db2HistoryEID */
typedef SQL_STRUCTURE db2HistoryEID
{
   SQL_PDB_NODE_TYPE ioNode;
   db2Uint32         ioHID;
} db2HistoryEID;

/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2gHistoryUpdate */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryUpdate (
    db2Uint32 version,
    void *pDB2GenHistoryUpdateStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2gHistoryUpdateStruct
{
   char                                *piNewLocation;
   char                                *piNewDeviceType;
   char                                *piNewComment;
   char                                *piNewStatus;
   db2Uint32                           iNewLocationLen;
   db2Uint32                           iNewDeviceLen;
   db2Uint32                           iNewCommentLen;
   db2Uint32                           iNewStatusLen;
   db2HistoryEID                       iEID;
} db2gHistoryUpdateStruct;
/* ... */
```

API-Parameter:

version
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2HistoryUpdateStruct*, übergeben wird.

pDB2HistoryUpdateStruct
Eingabe. Ein Zeiger auf die Struktur *db2HistoryUpdateStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

piNewLocation
Eingabe. Ein Zeiger auf eine Zeichenfolge, die eine neue Position für das Sicherungs-, Wiederherstellungs- oder Ladekopieimage angibt. Wenn dieser Parameter NULL ist oder auf null zeigt, bleibt der Wert unverändert.

piNewDeviceType
Eingabe. Ein Zeiger auf eine Zeichenfolge, die einen neuen Einheitentyp zur Speicherung des Sicherungs-, Wiederherstellungs- oder Ladekopieimages angibt. Wenn dieser Parameter NULL ist oder auf null zeigt, bleibt der Wert unverändert.

piNewComment
Eingabe. Ein Zeiger auf eine Zeichenfolge, die einen neuen Kommentar mit der Beschreibung des Eintrags angibt. Wenn dieser Parameter NULL ist oder auf null zeigt, bleibt der Kommentar unverändert.

piNewStatus
Eingabe. Ein Zeiger auf eine Zeichenfolge, die einen neuen Statustyp für den Eintrag angibt. Wenn dieser Parameter NULL ist oder auf null zeigt, bleibt der Status unverändert.

iNewLocationLen
Eingabe. Die Länge des Feldes *piNewLocationLen*.

iNewDeviceLen
Eingabe. Die Länge des Feldes *piNewDeviceLen*.

iNewCommentLen
Eingabe. Die Länge des Feldes *piNewCommentLen*.

iNewStatusLen
Eingabe. Die Länge des Feldes *piNewStatusLen*.

iEID Eingabe. Eine eindeutige Kennung, anhand derer ein spezifischer Eintrag in der Protokolldatei aktualisiert wird.

ioNode
Dieser Parameter kann entweder als Eingabe- oder als Ausgabeparameter verwendet werden.

Gibt die Knotennummer an.

ioHID Dieser Parameter kann entweder als Eingabe- oder als Ausgabeparameter verwendet werden.

Gibt die ID des Eintrags in der lokalen Protokolldatei an.

Syntax der REXX-API:

```
UPDATE RECOVERY HISTORY USING :value
```

Parameter der REXX-API:

value Eine zusammengesetzte REXX-Hostvariable, die Daten zur neuen Position eines Eintrags in einer Protokolldatei enthält. Im Folgenden steht XXX für den Namen der Hostvariablen:

- **XXX.0** Anzahl der Elemente in der Variablen (muss zwischen 1 und 4 liegen)
- **XXX.1** OBJECT_PART (Zeitmarke mit einer Folgenummer von 001 bis 999)
- **XXX.2** Neue Position für das Sicherungs- oder Kopierimage (optionaler Parameter)
- **XXX.3** Neue Einheit zur Speicherung des Sicherungs- oder Kopierimage (optionaler Parameter)
- **XXX.4** Neuer Kommentar (optionaler Parameter)

Hinweise:

Dies ist eine Aktualisierungsfunktion, und alle Daten vor der Änderung werden ersetzt und können nicht erneut erstellt werden. Diese Änderungen werden nicht protokolliert.

Die Datenbankprotokolldatei dient primär zur Aufzeichnung von Informationen, die im Protokoll enthaltenen Daten werden jedoch direkt von automatischen Wiederherstellungsoperationen verwendet. Wenn bei einer Wiederherstellung die Option AUTOMATIC angegeben wird, verwendet das Wiederherstellungsdienstprogramm das Protokoll der Sicherungsimages und deren Positionen zur automatischen Wiederherstellung. Wenn die automatische Wiederherstellungsfunktion verwendet werden soll und einige Sicherungsimages seit ihrer Erstellung verlagert wurden, wird empfohlen, den Datenbankprotokollsatz für diese Images zu aktualisieren, sodass die aktuelle Position der Images wiedergegeben wird. Wenn die Position der Sicherungsimages im Datenbankprotokoll nicht aktualisiert wird, kann die automatische Wiederherstellung die Sicherungsimages nicht finden, manuelle Wiederherstellungsbefehle können jedoch weiter erfolgreich verwendet werden.

Zugehörige Referenzen:

- „db2Rollforward - Datenbank aktualisierend wiederherstellen" auf Seite 154
- „db2Prune - Protokolldatei bereinigen" auf Seite 314
- „db2HistoryOpenScan - Suche in Protokolldatei starten" auf Seite 307
- „db2HistoryGetEntry - Nächsten Eintrag aus der Protokolldatei abrufen" auf Seite 304
- „db2HistoryCloseScan - Suche in Protokolldatei beenden" auf Seite 303
- „SQLCA" im Handbuch *Administrative API Reference*
- „UPDATE HISTORY FILE" auf Seite 299
- „db2Backup - Datenbank sichern" auf Seite 85

Zugehörige Beispiele:

- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.sqC -- How to recover a database (C++)"

db2Prune - Protokolldatei bereinigen

Löscht Einträge aus der Protokolldatei oder Protokolldateien aus dem aktiven Protokollpfad.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Erforderliche Verbindung:

Datenbank. Zum Löschen von Einträgen aus der Protokolldatei für eine beliebige andere Datenbank als die Standarddatenbank müssen Sie eine Verbindung zur Datenbank herstellen, bevor Sie diese API aufrufen.

API-Include-Datei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2Prune */
/* ... */
SQL_API_RC SQL_API_FN
  db2Prune (
    db2Uint32 version,
    void *pDB2PruneStruct,
  struct sqlca *pSqlca);
typedef struct
{
  char *piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2PruneStruct;
/* ... */
```

Syntax der generischen API:

```
/* File: db2ApiDf.h */
/* API: db2GenPrune */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenPrune (
    db2Uint32 version,
    void *pDB2GenPruneStruct,
  struct sqlca *pSqlca);
typedef struct
{
  db2Uint32 iStringLen;
  char *piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2GenPruneStruct;
/* ... */
```

API-Parameter:

version
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2PruneStruct*, übergeben wurde.

pDB2PruneStruct
Eingabe. Ein Zeiger auf die Struktur *db2PruneStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

iStringLen
Eingabe. Gibt die Länge von *piString* in Byte an.

piString
Eingabe. Ein Zeiger auf eine Zeichenfolge, die eine Zeitmarke oder eine Protokollfolgenummer angibt. Anhand der Zeitmarke oder des Teils einer Zeitmarke (mindestens *jjjj* bzw. die Jahresangabe) werden Datensätze zum Löschen ausgewählt. Alle Einträge kleiner-gleich der Zeitmarke werden gelöscht. Sie müssen eine gültige Zeitmarke angeben; für einen Parameter mit dem Wert von NULL gibt es keine Standardeinstellung.

Mit diesem Parameter können Sie auch eine Protokollfolgenummer übergeben, so dass inaktive Protokolle entfernt werden können.

iEID Eingabe. Gibt eine eindeutige Kennung an, anhand derer ein einzelner Eintrag aus der Protokolldatei entfernt wird.

iCallerAction
Eingabe. Gibt den Typ der auszuführenden Aktion an. Gültige Werte (definiert in `db2ApiDf.h`) sind:

DB2PRUNE_ACTION_HISTORY
Protokolldateieinträge entfernen.

DB2PRUNE_ACTION_LOG
Protokolldateien aus dem Pfad für aktive Protokolldateien entfernen.

iOptions
Eingabe. Gültige Werte (definiert in `db2ApiDf.h`) sind:

DB2PRUNE_OPTION_FORCE
Das Entfernen der letzten Sicherung erzwingen.

DB2PRUNE_OPTION_DELETE
Protokolldateien löschen, die bei der Bereinigung aus der Protokolldatei entfernt werden.

DB2PRUNE_OPTION_LSNSTRING
Angeben, dass der Wert von *piString* eine Protokollfolgenummer ist, die verwendet wird, wenn die Aktion `DB2PRUNE_ACTION_LOG` angegeben wird.

Syntax der REXX-API:

```
PRUNE RECOVERY HISTORY BEFORE :timestamp [WITH FORCE OPTION]
```

Parameter der REXX-API:

timestamp
Eine Hostvariable, die eine Zeitmarke enthält. Alle Einträge mit Zeitmarken kleiner-gleich der angegebenen Zeitmarke werden aus der Protokolldatei gelöscht.

WITH FORCE OPTION
Wenn dieser Parameter angegeben ist, wird die Protokolldatei gemäß der angegebenen Zeitmarke entfernt, selbst wenn einige Einträge aus der neuesten Wiederherstellungsgruppe aus der Datei gelöscht werden. Wenn dieser Parameter nicht angegeben ist, wird die neueste Wiederherstellungsgruppe beibehalten, selbst wenn die Zeitmarke kleiner-gleich der Zeitmarke ist, die als Eingabe angegeben ist.

Hinweise:

Das Entfernen der Protokolldatei bewirkt nicht, dass die tatsächlichen Sicherungs- oder Ladedateien gelöscht werden. Der Benutzer muss diese Dateien manuell löschen, um den von ihnen auf dem Datenträger belegten Speicherplatz freizugeben.

Achtung:
Wenn die neueste vollständige Datenbanksicherung von den Datenträgern gelöscht ist (zusätzlich zum Entfernen aus der Protokolldatei), muss der Benutzer sicherstellen, dass alle Tabellenbereiche, einschließlich des Katalogtabellenbereichs und der Benutzertabellenbereiche, gesichert werden. Andernfalls erhalten Sie eine Datenbank, die Sie nicht wiederherstellen können, oder ein Teil der Benutzerdaten in der Datenbank geht verloren.

Zugehörige Referenzen:

- „db2HistoryUpdate - Protokolldatei aktualisieren" auf Seite 311
- „db2HistoryOpenScan - Suche in Protokolldatei starten" auf Seite 307
- „db2HistoryGetEntry - Nächsten Eintrag aus der Protokolldatei abrufen" auf Seite 304
- „db2HistoryCloseScan - Suche in Protokolldatei beenden" auf Seite 303
- „SQLCA" im Handbuch *Administrative API Reference*

Zugehörige Beispiele:

- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.sqC -- How to recover a database (C++)"

db2ReadLogNoConn - Protokolldaten ohne Datenbankverbindung lesen

Extrahiert Protokollsätze aus den DB2 UDB-Datenbankprotokollen und fragt den Protokollmanager auf aktuelle Protokollstatusinformationen ab. Bevor Sie diese API verwenden, ordnen Sie mit db2ReadLogNoConnInit den Speicher zu, der als Eingabeparameter an diese API übergeben wird. Geben Sie nach Verwendung dieser API den Speicher mit db2ReadLogNoConnTerm wieder frei.

Berechtigung:

Keine

Erforderliche Verbindung:

Keine

API-Includedatei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2ReadLogNoConn */
/* ... */
SQL_API_RC SQL_API_FN
db2ReadLogNoConn (
  db2Uint32 versionNumber,
  void *pDB2ReadLogNoConnStruct,
  struct sqlca *pSqlca);
typedef SQL_STRUCTURE db2ReadLogNoConnStruct
{
  db2Uint32                   iCallerAction;
  SQLU_LSN                    *piStartLSN;
  SQLU_LSN                    *piEndLSN;
```

```
  char                        *poLogBuffer;
  db2Uint32                   iLogBufferSize;
  char                        *piReadLogMemPtr;
  db2ReadLogNoConnInfoStruct  *poReadLogInfo;
} db2ReadLogNoConnStruct;

typedef SQL_STRUCTURE db2ReadLogNoConnInfoStruct
{
  SQLU_LSN                    firstAvailableLSN;
  SQLU_LSN                    firstReadLSN;
  SQLU_LSN                    nextStartLSN;
  db2Uint32                   logRecsWritten;
  db2Uint32                   logBytesWritten;
  db2Uint32                   lastLogFullyRead;
  db2TimeOfLog                currentTimeValue;
} db2ReadLogNoConnInfoStruct;

/* ... */
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2ReadLogNoConnStruct*, übergeben wird.

pDB2ReadLogNoConnStruct
Eingabe. Ein Zeiger auf die Struktur *db2ReadLogNoConnStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

iCallerAction
Eingabe. Gibt die auszuführende Aktion an. Gültige Werte sind:

DB2READLOG_READ
Das Datenbankprotokoll von der Start-LSN bis zur End-LSN lesen und die Protokollsätze innerhalb dieses Bereichs zurückgeben.

DB2READLOG_READ_SINGLE
Einen einzelnen Protokollsatz lesen (unabhängig davon, ob er weitergegeben werden kann). Dieser Satz wird mit der Start-LSN angegeben.

DB2READLOG_QUERY
Das Datenbankprotokoll abfragen. Ergebnisse der Abfrage über die Struktur db2ReadLogNoConnInfoStruct zurücksenden.

piStartLSN
Eingabe. Die Start-LSN gibt die relative Bytestartadresse für das Lesen des Protokolls an. Dieser Wert muss der Anfang eines tatsächlichen Protokollsatzes sein.

piEndLSN
Eingabe. Die End-LSN gibt die relative Byteendadresse für das Lesen des Protokolls an. Dieser Wert muss größer als *piStartLsn* sein, er muss jedoch nicht das Ende eines tatsächlichen Protokollsatzes angeben.

poLogBuffer
Ausgabe. Puffer zur sequenziellen Speicherung aller Protokollsätze, die weitergegeben werden können und die innerhalb des angegeben Bereichs gelesen wurden. Dieser Puffer muss groß genug für einen einzelnen Protokollsatz sein. Als Richtlinie sollte dieser Puffer mindestens 32 Byte enthalten können. Seine Maximalgröße hängt von der Größe des angefor-

derten Bereichs ab. Die einzelnen Protokollsätze im Puffer weisen als Präfix die sechs Byte lange Protokollfolgenummer des darauffolgenden Protokollsatzes auf.

iLogBufferSize
Eingabe. Gibt die Größe des Protokollpuffers in Byte an.

piReadLogMemPtr
Eingabe. Speicherblock der Größe iReadLogMemoryLimit, der im Initialisierungsaufruf zugeordnet wurde. Dieser Speicher enthält persistente Daten, die die API bei jedem Aufruf benötigt. Dieser Speicherblock darf nicht erneut zugeordnet oder vom Aufrufenden in irgendeiner Weise geändert werden.

poReadLogInfo
Ausgabe. Ein Zeiger auf die Struktur *db2ReadLogNoConnInfoStruct*.

firstAvailableLSN
Die erste verfügbare Protokollfolgenummer in verfügbaren Protokollen.

firstReadLSN
Die erste Protokollfolgenummer, die in diesem Aufruf gelesen wird.

nextStartLSN
Die nächste, lesbare Protokollfolgenummer.

logRecsWritten
Die Anzahl der Protokollsätze, die in das Protokollpufferfeld *poLogBuffer* geschrieben werden.

logBytesWritten
Die Anzahl Byte, die in das Protokollpufferfeld *poLogBuffer* geschrieben werden.

lastLogFullyRead
Die Nummer der letzten, vollständig gelesenen Protokolldatei.

Hinweise:

Die API db2ReadLogNoConn benötigt einen Speicherblock, der mit der API db2ReadLogNoConnInit zugeordnet werden muss. Der Speicherblock muss als Eingabeparameter an alle nachfolgenden Aufrufe der API db2ReadLogNoConn übergeben werden und darf nicht geändert werden.

Bei einer Anforderung eines sequenziellen Protokolllesevorgangs benötigt die API einen Protokollfolgenummernbereich sowie den zugeordneten Speicher. Die API gibt eine Reihe von Protokollsätzen zurück, basierend auf den bei der Initialisierung angegebenen Filteroptionen und dem Protokollfolgenummernbereich. Bei Anforderung einer Abfrage enthält die Informationsstruktur zum Lesen von Protokolldaten eine gültige Start-LSN, die bei einem Leseaufruf verwendet werden kann. Der Wert, der bei einem Lesevorgang als End-LSN verwendet wird, kann einer der folgenden Werte sein:

- Ein Wert größer als die vom Aufrufenden angegebene Start-LSN.
- `FFFF FFFF FFFF`, wobei dieser Wert vom Programm zum asynchronen Lesen von Protokolldaten als Ende der verfügbaren Protokolle interpretiert wird.

Protokollsätze, die weitergegeben werden können und innerhalb des Start- und Endebereichs der Protokollfolgenummern gelesen werden, werden in den Protokollpuffer zurückgegeben. Ein Protokollsatz enthält nicht die zugehörige Protokollfolgenummer; diese steht im Puffer vor dem tatsächlichen Protokollsatz.

Beschreibungen der verschiedenen DB2-Protokollsätze, die von db2ReadLogNoConn zurückgegeben werden, finden Sie im Abschnitt zu den DB2 UDB-Protokollsätzen.

Zum Lesen des nächsten sequenziellen Protokollsatzes nach dem ersten Lesen verwenden Sie den Wert nextStartLSN, der in db2ReadLogNoConnInfoStruct zurückgegeben wird. Wiederholen Sie den Aufruf mit dieser neuen Start-LSN und einer gültigen End-LSN, und der nächste Block Protokollsätze wird gelesen. Der *sqlca*-Code SQLU_RLOG_READ_TO_CURRENT bedeutet, dass bis zum Ende der verfügbaren Protokolldateien gelesen wurde.

Wenn Sie die API nicht länger verwenden, geben Sie den Speicher mit db2ReadLogNoConnTerm wieder frei.

Zugehörige Referenzen:

- „SQLCA" im Handbuch *Administrative API Reference*
- „db2ReadLogNoConnInit - Lesen von Protokolldaten ohne Datenbankverbindung initialisieren" auf Seite 320
- „db2ReadLogNoConnTerm - Lesen von Protokolldaten ohne Datenbankverbindung beenden" auf Seite 322

db2ReadLogNoConnInit - Lesen von Protokolldaten ohne Datenbankverbindung initialisieren

Ordnet den Speicher zu, der von db2ReadLogNoConn zum Extrahieren von Protokollsätzen aus den DB2 UDB-Datenbankprotokollen und zum Abfragen des Protokollmanagers nach aktuellen Protokollstatusinformationen verwendet werden soll.

Berechtigung:

Keine

Erforderliche Verbindung:

Keine

API-Includedatei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2ReadLogNoConnInit */
/* ... */
SQL_API_RC SQL_API_FN
db2ReadLogNoConnInit (
  db2Uint32 versionNumber,
  void * pDB2ReadLogNoConnInitStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnInitStruct
{
  db2Uint32                  iFilterOption;
  char                       *piLogFilePath;
  char                       *piOverflowLogPath;
```

```
  db2Uint32                  iRetrieveLogs;
  char                       *piDatabaseName;
  char                       *piNodeName;
  db2Uint32                  iReadLogMemoryLimit;
  char                       **poReadLogMemPtr;
} db2ReadLogNoConnInitStruct;
/* ... */
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2ReadLogNoConnInitStruct*, übergeben wird.

pDB2ReadLogNoConnInitStruct
Eingabe. Ein Zeiger auf die Struktur *db2ReadLogNoConnInitStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

iFilterOption
Eingabe. Gibt die Ebene der Protokollsatzfilterung an, die beim Lesen der Protokollsätze verwendet werden soll. Gültige Werte sind:

DB2READLOG_FILTER_OFF
Alle Protokollsätze im gegebenen Protokollfolgenummernbereich lesen.

DB2READLOG_FILTER_ON
Nur die Protokollsätze im gegebenen Protokollfolgenummernbereich lesen, die als weitergebbar markiert sind. Dies ist das übliche Verhalten der API zum asynchronen Lesen von Protokolldaten.

piLogFilePath
Eingabe. Der Pfad, in dem sich die zu lesenden Protokolldateien befinden.

piOverflowLogPath
Eingabe. Ein Alternativpfad, in dem sich die zu lesenden Protokolldateien befinden können.

iRetrieveLogs
Eingabe. Option, die angibt, ob ein Benutzerexit aufgerufen werden soll, um die Protokolldateien abzurufen, die weder im Protokolldateipfad noch im Überlaufprotokollpfad gefunden werden können. Gültige Werte sind:

DB2READLOG_RETRIEVE_OFF
Zum Abrufen der fehlenden Protokolldateien keinen Benutzerexit aufrufen.

DB2READLOG_RETRIEVE_LOGPATH
Zum Abrufen der fehlenden Protokolldateien in den angegebenen Protokolldateipfad einen Benutzerexit aufrufen.

DB2READLOG_RETRIEVE_OVERFLOW
Zum Abrufen der fehlenden Protokolldateien in den angegebenen Überlaufprotokollpfad soll ein Benutzerexit aufgerufen werden.

piDatabaseName
Eingabe. Name der Datenbank, der die gelesenen Wiederherstellungsprotokolle gehören. Diese Angabe ist erforderlich, wenn die oben aufgeführte Abrufoption angegeben wird.

piNodeName
Eingabe. Name des Knotens, dem die gelesenen Wiederherstellungsprotokolle gehören. Diese Angabe ist erforderlich, wenn die oben aufgeführte Abrufoption angegeben wird.

iReadLogMemoryLimit
Eingabe. Maximale Anzahl Byte, die die API intern zuordnen kann.

poReadLogMemPtr
Ausgabe. Der von der API zugeordnete Speicherblock der Größe *iReadLogMemoryLimit*. Dieser Speicher enthält persistente Daten, die die API bei jedem Aufruf benötigt. Dieser Speicherblock darf nicht erneut zugeordnet oder vom Aufrufenden in irgendeiner Weise geändert werden.

Hinweise:

Der von db2ReadLogNoConnInit initialisierte Speicher darf nicht geändert werden.

Wenn db2ReadLogNoConn nicht länger verwendet wird, rufen Sie db2ReadLogNoConnTerm auf, um den von db2ReadLogNoConnInit initialisierten Speicher freizugeben.

Zugehörige Referenzen:
- „SQLCA" im Handbuch *Administrative API Reference*
- „db2ReadLogNoConn - Protokolldaten ohne Datenbankverbindung lesen" auf Seite 317
- „db2ReadLogNoConnTerm - Lesen von Protokolldaten ohne Datenbankverbindung beenden" auf Seite 322

db2ReadLogNoConnTerm - Lesen von Protokolldaten ohne Datenbankverbindung beenden

Gibt den von db2ReadLogNoConn verwendeten Speicher frei, der ursprünglich von db2ReadLogNoConnInit initialisiert wurde.

Berechtigung:

Keine

Erforderliche Verbindung:

Keine

API-Includedatei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2ReadLogNoConnTerm */
/* ... */
SQL_API_RC SQL_API_FN
db2ReadLogNoConnTerm (
  db2Uint32 versionNumber,
  void * pDB2ReadLogNoConnTermStruct,
   struct sqlca * pSqlca);
```

```
typedef SQL_STRUCTURE db2ReadLogNoConnTermStruct
{
  char                          **poReadLogMemPtr;
} db2ReadLogNoConnTermStruct;
/* ... */
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2ReadLogNoConnTermStruct*, übergeben wird.

pDB2ReadLogNoConnTermStruct
Eingabe. Zeigt auf die Struktur *db2ReadLogNoConnTermStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

poReadLogMemPtr
Ausgabe. Zeigt auf den Speicherblock, der im Initialisierungsaufruf zugeordnet wurde. Dieser Zeiger wird freigegeben und auf NULL gesetzt.

Zugehörige Referenzen:

- „SQLCA" im Handbuch *Administrative API Reference*
- „db2ReadLogNoConn - Protokolldaten ohne Datenbankverbindung lesen" auf Seite 317
- „db2ReadLogNoConnInit - Lesen von Protokolldaten ohne Datenbankverbindung initialisieren" auf Seite 320

db2ReadLog - Protokolldaten asynchron lesen

Extrahiert Protokollsätze aus den DB2 UDB-Datenbankprotokollen und fragt den Protokollmanager auf aktuelle Protokollstatusinformationen ab. Diese API können Sie nur mit wiederherstellbaren Datenbanken verwenden. Eine Datenbank ist wiederherstellbar, wenn sie so konfiguriert ist, dass *logretain* auf `RECOVERY` oder *userexit* auf `ON` gesetzt ist.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *dbadm*

Erforderliche Verbindung:

Datenbank

API-Includedatei:

db2ApiDf.h

Syntax der C-API:

```
/* File: db2ApiDf.h */
/* API: db2ReadLog */
/* ... */
SQL_API_RC SQL_API_FN
```

```
  db2ReadLog (
  db2Uint32 versionNumber,
    void *pDB2ReadLogStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2ReadLogStruct
{
  db2Uint32                  iCallerAction;
  SQLU_LSN                   *piStartLSN;
  SQLU_LSN                   *piEndLSN;
  char                       *poLogBuffer;
  db2Uint32                  iLogBufferSize;
  db2Uint32                  iFilterOption;
  db2ReadLogInfoStruct       *poReadLogInfo;

typedef SQL_STRUCTURE db2ReadLogInfoStruct
{
SQLU_LSN       initialLSN ;
SQLU_LSN       firstReadLSN ;
  SQLU_LSN                   nextStartLSN;
  db2Uint32                  logRecsWritten;
  db2Uint32                  logBytesWritten;
  SQLU_LSN                   firstReusedLSN;
  db2Uint32                  timeOfLSNReuse;
  db2TimeOfLog               currentTimeValue;
} db2ReadLogInfoStruct;

typedef SQL_STRUCTURE db2TimeOfLog
{
  db2Uint32                  seconds;
  db2Uint32                  accuracy;
} db2TimeOfLog;
/* ... */
```

API-Parameter:

versionNumber
Eingabe. Gibt die Version und den Releasestand der Struktur an, die als zweiter Parameter, *pDB2ReadLogStruct*, übergeben wird.

pDB2ReadLogStruct
Eingabe. Ein Zeiger auf *db2ReadLogStruct*.

pSqlca
Ausgabe. Ein Zeiger auf die *sqlca*-Struktur.

iCallerAction
Eingabe. Gibt die auszuführende Aktion an.

DB2READLOG_READ
Das Datenbankprotokoll von der Start-LSN bis zur End-LSN lesen und die Protokollsätze innerhalb dieses Bereichs zurückgeben.

DB2READLOG_READ_SINGLE
Einen einzelnen Protokollsatz lesen (unabhängig davon, ob er weitergegeben werden kann). Dieser Satz wird mit der Start-LSN angegeben.

DB2READLOG_QUERY
Das Datenbankprotokoll abfragen. Ergebnisse der Abfrage werden über die Struktur *db2ReadLogInfoStruct* zurückgesendet.

piStartLsn
Eingabe. Die Start-LSN gibt die relative Bytestartadresse für das Lesen des Protokolls an. Dieser Wert muss der Anfang eines tatsächlichen Protokollsatzes sein.

piEndLsn
Eingabe. Die End-LSN gibt die relative Byteendadresse für das Lesen des Protokolls an. Dieser Wert muss größer als *startLsn* sein, muss jedoch nicht das Ende des tatsächlichen Protokollsatzes angeben.

poLogBuffer
Ausgabe. Puffer zur sequenziellen Speicherung aller Protokollsätze, die weitergegeben werden können und die innerhalb des angegeben Bereichs gelesen wurden. Dieser Puffer muss groß genug für einen einzelnen Protokollsatz sein. Als Richtlinie sollte dieser Puffer mindestens 32 Byte enthalten können. Seine Maximalgröße hängt von der Größe des angeforderten Bereichs ab. Die einzelnen Protokollsätze im Puffer weisen als Präfix die sechs Byte lange Protokollfolgenummer des darauffolgenden Protokollsatzes auf.

iLogBufferSize
Eingabe. Gibt die Größe des Protokollpuffers in Byte an.

iFilterOption
Eingabe. Gibt die Ebene der Protokollsatzfilterung an, die beim Lesen der Protokollsätze verwendet werden soll. Gültige Werte sind:

DB2READLOG_FILTER_OFF
Alle Protokollsätze im gegebenen Protokollfolgenummernbereich lesen.

DB2READLOG_FILTER_ON
Nur die Protokollsätze im gegebenen Protokollfolgenummernbereich lesen, die als weitergebbar markiert sind. Dies ist das übliche Verhalten der API zum asynchronen Lesen von Protokolldaten.

poReadLogInfo
Ausgabe. Eine Struktur mit zusätzlichen Informationen zum Aufruf und zum Datenbankprotokoll.

Hinweise:

Wenn als Aktion das Lesen des Protokolls angegeben ist, stellt das aufrufende Programm einen Protokollfolgenummernbereich und einen Puffer zum Speichern der Protokollsätze bereit. Die API liest das Protokoll sequenziell im angeforderten Protokollfolgenummernbereich und gibt Protokollsätze zurück, die Tabellen zugeordnet sind, für die DATA CAPTURE auf CHANGES gesetzt ist. Außerdem gibt die API die Struktur db2ReadLogInfoStruct mit den aktuellen Daten des aktiven Protokolls zurück. Wenn die angeforderte Aktion eine Abfrage ist, gibt die API die Struktur db2ReadLogInfoStruct mit den aktuellen Daten des aktiven Protokolls zurück.

Zum asynchronen Lesen von Protokolldaten müssen Sie zuerst das Datenbankprotokoll auf eine gültige Start-LSN abfragen. Nach dem Abfrageaufruf enthält die Informationsstruktur zum Lesen von Protokolldaten (db2ReadLogInfoStruct) eine gültige Start-LSN (im Element initialLSN), die in einem Leseaufruf verwendet werden soll. Der Wert, der bei einem Lesevorgang als End-LSN verwendet wird, kann einer der folgenden Werte sein:

- Ein Wert größer als initialLSN

- FFFF FFFF FFFF, wobei dieser Wert vom Programm zum asynchronen Lesen von Protokolldaten als Ende des aktuellen Protokolls interpretiert wird.

Protokollsätze, die weitergegeben werden können und innerhalb des Start- und Endebereichs der Protokollfolgenummern gelesen werden, werden in den Protokollpuffer zurückgegeben. Ein Protokollsatz enthält nicht die zugehörige Protokollfolgenummer; diese steht im Puffer vor dem eigentlichen Protokollsatz. Beschreibungen der verschiedenen DB2-Protokollsätze, die von db2ReadLog zurückgegeben werden, finden Sie im Abschnitt zu den DB2 UDB-Protokollsätzen.

Zum Lesen des nächsten sequenziellen Protokollsatzes nach dem ersten Lesen verwenden Sie das Feld nextStartLSN, das in der Struktur db2ReadLogStruct zurückgegeben wird. Wiederholen Sie den Aufruf mit dieser neuen Start-LSN und einer gültigen End-LSN. Der nächste Datensatzblock wird anschließend gelesen. Der *sqlca*-Code SQLU_RLOG_READ_TO_CURRENT bedeutet, dass bis zum Ende des aktuellen aktiven Protokolls gelesen wurde.

Zugehörige Referenzen:

- „SQLCA" im Handbuch *Administrative API Reference*
- „db2Reorg - Reorganize" im Handbuch *Administrative API Reference*

Zugehörige Beispiele:

- „dbrecov.sqc -- How to recover a database (C)"
- „dbrecov.sqC -- How to recover a database (C++)"

db2HistData

Diese Struktur können Sie verwenden, um nach einem Aufruf von db2HistoryGetEntry Daten zurückzugeben.

Tabelle 2. Felder in der Struktur db2HistData

Feldname	Datentyp	Beschreibung
ioHistDataID	char(8)	Eine acht Byte lange Strukturkennung für Speicherauszüge. Der einzige gültige Wert lautet „SQLUHINF". Für diese Zeichenfolge ist keine symbolische Definition vorhanden.
oObjectPart	db2Char	Die ersten 14 Zeichen sind eine Zeitmarke mit dem Format *jjjjmmtthhnnss*, die angibt, wann eine Operation begann. Die nächsten drei Zeichen sind eine Folgenummer. Jede Sicherungsoperation kann mehrere Einträge in dieser Datei ergeben, wenn das Sicherungsimage in mehreren Dateien oder auf mehreren Bändern gespeichert ist. Aufgrund der Folgenummer können Sie mehrere Speicherpositionen angeben. Wiederherstellungs- und Ladeoperationen weisen in dieser Datei nur einen einzigen Eintrag auf. Dieser entspricht der Folgenummer '001' der entsprechenden Sicherung. Die Kombination aus Zeitmarke und Folgenummer muss eindeutig sein.
oEndTime	db2Char	Eine Zeitmarke mit dem Format *jjjjmmtthhnnss*, die angibt, wann die Operation endete.

Tabelle 2. Felder in der Struktur db2HistData (Forts.)

Feldname	Datentyp	Beschreibung
oFirstLog	db2Char	Die ID der ältesten Protokolldatei (Bereich von S0000000 bis S9999999): • Erforderlich, um eine aktualisierende Wiederherstellung für eine Onlinesicherung anzuwenden • Erforderlich, um eine aktualisierende Wiederherstellung für eine Offlinesicherung anzuwenden • Angewendet nach der Wiederherstellung einer vollständigen Datenbank oder nach einer Sicherung auf Tabellenbereichsebene, die beim Beginn des Ladens aktuell war
oLastLog	db2Char	Die ID der neuesten Protokolldatei (Bereiche von S0000000 bis S9999999): • Erforderlich, um eine aktualisierende Wiederherstellung für eine Onlinesicherung anzuwenden • Erforderlich, um eine aktualisierende Wiederherstellung bis zum aktuellen Zeitpunkt für eine Offlinesicherung anzuwenden • Angewendet nach der Wiederherstellung einer vollständigen Datenbank oder nach einer Sicherung auf Tabellenbereichsebene, die beim Ende des Ladens aktuell war (wie bei *oFirstLog*, sofern keine aktualisierende Wiederherstellung angewendet wird)
oID	db2Char	Eindeutige Kennung für die Sicherung oder die Tabelle.
oTableQualifier	db2Char	Tabellenqualifikationsmerkmal.
oTableName	db2Char	Tabellenname.
oLocation	db2Char	Für Sicherungs- und Ladekopien zeigt dieses Feld an, wo die Daten gespeichert wurden. Für Operationen, die mehrere Einträge in der Datei erfordern, gibt die durch *oObjectPart* definierte Folgenummer an, welcher Teil der Sicherungskopie sich an der angegebenen Position befindet. Für Wiederherstellungs- und Ladeoperationen gibt die Position immer an, wo der erste Teil der wiederhergestellten oder geladenen Daten (entsprechend der Folgenummer '001' für mehrteilige Sicherungen) gespeichert wurde. Die Daten in *oLocation* werden abhängig von *oDeviceType* interpretiert: • Für eine Platte oder Diskette (`D` oder `K`) als vollständig qualifizierter Dateiname • Für ein Band (`T`) als Datenträgerkennsatz • Für TSM (`A`) als Servername • Für einen Benutzerexit oder für Sonstiges (`U` oder `O`) als Text mit freiem Format
oComment	db2Char	Kommentartext mit freiem Format.
oCommandText	db2Char	Befehlstext oder DDL.
oLastLSN	SQLU_LSN	Letzte Protokollfolgenummer.
oEID	Struktur	Eindeutige Eintragskennung.
poEventSQLCA	Struktur	Ergebnis-*sqlca* des aufgezeichneten Ereignisses.
poTablespace	db2Char	Eine Liste von Tabellenbereichsnamen.

Tabelle 2. Felder in der Struktur db2HistData (Forts.)

Feldname	Datentyp	Beschreibung
ioNumTablespaces	db2Uint32	Anzahl der Einträge in der Liste *poTablespace*. Jede Tabellenbereichssicherung enthält mindestens einen Tabellenbereich. Jede Tabellenbereichswiederherstellung ersetzt mindestens einen Tabellenbereich. Wenn dieses Feld ungleich null ist (Angabe einer Sicherung oder einer Wiederherstellung auf Tabellenbereichsebene), enthalten die nächsten Zeilen in dieser Datei den Namen des gesicherten oder wiederhergestellten Tabellenbereichs als 18 Zeichen lange Zeichenfolge. In jeder Zeile wird ein Name eines Tabellenbereichs angezeigt.
oOperation	char	Siehe Tabelle 3.
oObject	char	Granularität der Operation: `D` für gesamte Datenbank, `P` für Tabellenbereich und `T` für Tabelle.
oOptype	char	Siehe Tabelle 4 auf Seite 329.
oStatus	char	Eintragsstatus: `A` für Aktion, `D` für gelöscht (zukünftige Verwendung), `E` für abgelaufen, `I` für inaktiv, `N` für noch nicht festgeschrieben, `Y` für festgeschrieben oder aktiv, `a` für aktive Sicherung, wobei einige DataLink-Server die Sicherung jedoch noch nicht abgeschlossen haben, `i` für inaktive Sicherung, wobei einige DataLink-Server die Sicherung jedoch noch nicht abgeschlossen haben.
oDeviceType	char	Einheitentyp. Dieses Feld gibt an, wie das Feld *oLocation* interpretiert wird: `A` für TSM, `C` für Client, `D` für Platte, `K` für Diskette, `L` für lokal, `O` für Sonstiges (Unterstützung durch eine Einheit eines anderen Herstellers), `P` für Pipe, `Q` für Cursor, `S` für Server, `T` für Band und `U` für Benutzerexit.

Tabelle 3. Gültige oOperation-Werte in der Struktur db2HistData

Wert	Beschreibung	C-Definition	COBOL/FORTRAN-Definition
A	Tabellenbereich hinzufügen	DB2HISTORY_OP_ADD_ TABLESPACE	DB2HIST_OP_ADD_ TABLESPACE
B	sichern	DB2HISTORY_OP_BACKUP	DB2HIST_OP_BACKUP
C	Kopie laden	DB2HISTORY_OP_LOAD_COPY	DB2HIST_OP_LOAD_COPY
D	gelöschte Tabelle	DB2HISTORY_OP_DROPPED_ TABLE	DB2HIST_OP_DROPPED_TABLE
F	aktualisierend wiederherstellen	DB2HISTORY_OP_ROLLFWD	DB2HIST_OP_ROLLFWD
G	Tabelle reorganisieren	DB2HISTORY_OP_REORG	DB2HIST_OP_REORG
L	laden	DB2HISTORY_OP_LOAD	DB2HIST_OP_LOAD
N	Tabellenbereich umbenennen	DB2HISTORY_OP_REN_ TABLESPACE	DB2HIST_OP_REN_ TABLESPACE
O	Tabellenbereich löschen	DB2HISTORY_OP_DROP_ TABLESPACE	DB2HIST_OP_DROP_ TABLESPACE
Q	in Wartemodus versetzen	DB2HISTORY_OP_QUIESCE	DB2HIST_OP_QUIESCE
R	wiederherstellen	DB2HISTORY_OP_RESTORE	DB2HIST_OP_RESTORE
T	Tabellenbereich ändern	DB2HISTORY_OP_ALT_ TABLESPACE	DB2HIST_OP_ALT_TBS

Tabelle 3. Gültige oOperation-Werte in der Struktur db2HistData (Forts.)

Wert	Beschreibung	C-Definition	COBOL/FORTRAN-Definition
U	aus Tabelle laden	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD

Tabelle 4. Gültige oOptype-Werte in der Struktur db2HistData

oOperation	oOptype	Beschreibung	C/COBOL/FORTRAN-Definition
B	F	offline	DB2HISTORY_OPTYPE_OFFLINE
	N	online	DB2HISTORY_OPTYPE_ONLINE
	I	inkrementell offline	DB2HISTORY_OPTYPE_INCR_ OFF-LINE
	O	inkrementell online	DB2HISTORY_OPTYPE_INCR_ ONLINE
	D	delta offline	DB2HISTORY_OPTYPE_DELTA_ OFF-LINE
	V	delta online	DB2HISTORY_OPTYPE_DELTA_ ONLINE
F	V	Ende der Proto-kolle	DB2HISTORY_OPTYPE_EOL
	P	zu bestimmtem Zeitpunkt	DB2HISTORY_OPTYPE_PIT
G	F	offline	DB2HISTORY_OPTYPE_OFFLINE
	N	online	DB2HISTORY_OPTYPE_ONLINE
L	I	einfügen	DB2HISTORY_OPTYPE_INSERT
	R	ersetzen	DB2HISTORY_OPTYPE_REPLACE
Q	S	Wartemodus: SHARE	DB2HISTORY_OPTYPE_SHARE
	U	Wartemodus: UPDATE	DB2HISTORY_OPTYPE_UPDATE
	X	Wartemodus: EXCLUSIVE	DB2HISTORY_OPTYPE_EXCL
	Z	Wartemodus: RESET	DB2HISTORY_OPTYPE_RESET
R	F	offline	DB2HISTORY_OPTYPE_OFFLINE
	N	online	DB2HISTORY_OPTYPE_ONLINE
	I	inkrementell offline	DB2HISTORY_OPTYPE_INCR_ OFF-LINE
	O	inkrementell online	DB2HISTORY_OPTYPE_INCR_ ONLINE
T	C	Behälter hinzufü-gen	DB2HISTORY_OPTYPE_ADD_CONT
	R	neu ausgleichen	DB2HISTORY_OPTYPE_REB

Tabelle 5. Felder in der Struktur db2Char

Feldname	Datentyp	Beschreibung
pioData	char	Ein Zeiger auf einen Puffer für Zeichendaten. Wenn der Wert NULL ist, werden keine Daten zurückgegeben.
iLength	db2Uint32	Eingabe. Die Größe des Puffers *pioData*.
oLength	db2Uint32	Ausgabe. Die Anzahl gültiger Zeichen im Puffer *pioData*.

Tabelle 6. Felder in der Struktur db2HistoryEID

Feldname	Datentyp	Beschreibung
ioNode	SQL_PDB_NODE_TYPE	Knotennummer.
ioHID	db2Uint32	ID des Eintrags in der lokalen Protokolldatei.

Sprachsyntax:

C-Struktur

```
/* File: db2ApiDf.h */
/* ... */
typedef SQL_STRUCTURE db2HistoryData
{
  char ioHistDataID[8];
  db2Char oObjectPart;
  db2Char oEndTime;
  db2Char oFirstLog;
  db2Char oLastLog;
  db2Char oID;
  db2Char oTableQualifier;
  db2Char oTableName;
  db2Char oLocation;
  db2Char oComment;
  db2Char oCommandText;
  SQLU_LSN oLastLSN;
  db2HistoryEID oEID;
  struct sqlca * poEventSQLCA;
  db2Char * poTablespace;
  db2Uint32 ioNumTablespaces;
  char oOperation;
  char oObject;
  char oOptype;
  char oStatus;
  char oDeviceType
} db2HistoryData;

typedef SQL_STRUCTURE db2Char
{
  char * pioData;
  db2Uint32 ioLength
} db2Char;

typedef SQL_STRUCTURE db2HistoryEID
{
  SQL_PDB_NODE_TYPE ioNode;
  db2Uint32 ioHID
} db2HistoryEID;
/* ... */
```

Zugehörige Referenzen:

- „db2HistoryGetEntry - Nächsten Eintrag aus der Protokolldatei abrufen" auf Seite 304
- „SQLCA" im Handbuch *Administrative API Reference*

SQLU-LSN

Diese Gesamtverknüpfung wird von der API db2ReadLog verwendet und enthält die Definition der Protokollfolgenummer. Eine Protokollfolgenummer ist eine relative Byteadresse innerhalb des Datenbankprotokolls. Alle Protokollsätze sind mit dieser Nummer gekennzeichnet. Sie stellt die relative Byteadresse des Datensatzes ab dem Beginn des Datenbankprotokolls dar.

Tabelle 7. Felder in der Gesamtverknüpfung SQLU-LSN

Feldname	Datentyp	Beschreibung
lsnChar	Matrix von UNSIGNED CHAR	Gibt die Protokollfolgenummer der Zeichenmatrix aus sechs Elementen an.
lsnWord	Matrix von UNSIGNED SHORT	Gibt die Protokollfolgenummer der kurzen Matrix aus drei Elementen an.

Sprachsyntax:

C-Struktur

```
typedef union SQLU_LSN
{
unsigned char  lsnChar  [6] ;
unsigned short lsnWord  [3] ;
} SQLU_LSN;
```

Zugehörige Referenzen:

- „db2ReadLog - Protokolldaten asynchron lesen" auf Seite 323

Anhang E. Beispielwiederherstellungsprogramme

Beispielprogramme mit eingebettetem SQL

Die folgenden Beispielprogramme zeigen, wie Sie mit den DB2-Sicherungs- und -Wiederherstellungs-APIs folgende Aktionen ausführen können:

- Einträge in der Datenbankwiederherstellungsdatei lesen und aktualisieren
- Datenbankprotokolldateien mit einer Datenbankverbindung lesen
- Datenbankprotokolldateien ohne Datenbankverbindung lesen
- Eine Datenbank aus einem Sicherungsimage wiederherstellen
- Nach einer Datenbankwiederherstellungsoperation eine aktualisierende Wiederherstellungsoperation ausführen

Anmerkung: Diese Beispieldateien befinden sich in den Verzeichnissen `sqllib/samples/c` und `sqllib/samples/cpp`.

Beispielprogramm dbredirect:

Die Beispieldateien für `dbredirect` zeigen, wie eine umgeleitete Wiederherstellung einer Datenbank ausgeführt wird.

```
/****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
****************************************************************************
**
** SOURCE FILE NAME: dbredirect.sqc
**
** SAMPLE: How to perform Redirected Restore of a database
**
**         This program ends in ".sqc" even though it does not contain
**         embedded SQL statements. It links in  the embedded SQL utility
**         file for database connection and disconnection, so it needs the
**         embedded SQL extension for the precompiler.
**
**         Note:
**           You must be disconnected from the sample database to run
**           this program. To ensure you are, enter 'db2 connect reset'
**           on the command line prior to running dbredirect. If the target
**           database for the redirected restore already exists, SQLCODE 2529
**           will be displayed.
**
** DB2 API USED:
**        db2CfgSet -- Set Configuration
**        db2Restore -- Restore Database
**
** OUTPUT FILE: dbredirect.out (available in the online documentation)
****************************************************************************
**
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
```

```
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
******************************************************************************/
#include "utilrecov.c"

/* local function prototypes */
int DbBackupAndRedirectedRestore(char *, char *, char *, char *, char *);

/* support function called by DbBackupAndRedirectedRestore() */
int InaccessableContainersRedefine(char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  char redirectedRestoredDbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("\nTHIS SAMPLE SHOWS HOW TO PERFORM A REDIRECTED RESTORE\n");
  printf("FROM A DATABASE BACKUP.\n");

  strcpy(redirectedRestoredDbAlias, "RRDB");

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  printf("\nNOTE: Backup images will be created on the server\n");
  printf("      in the directory %s,\n", serverWorkingPath);
  printf("      and will not be deleted by the program.\n");

  /* call the sample function */
  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);
  CHECKRC(rc, "DbRecoveryHistoryFilePrune");

  rc = DbBackupAndRedirectedRestore(dbAlias,
                                    redirectedRestoredDbAlias,
                                    user, pswd, serverWorkingPath);
  CHECKRC(rc, "DbBackupAndRedirectedRestore");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");
```

```
  return 0;
} /* end main */

int DbBackupAndRedirectedRestore(char dbAlias[],
                        char restoredDbAlias[],
                                        char user[],
                                  char pswd[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  unsigned short logretain = 0;

  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1] = { 0 };

  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32 backupImageSize = 0;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;

  printf("\n*************************\n");
  printf("*** REDIRECTED RESTORE ***\n");
  printf("*************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2CfgSet -- Upate Configuration\n");
  printf("  db2Backup -- Backup Database\n");
  printf("  sqlecrea -- Create Database\n");
  printf("  db2Restore -- Restore Database\n");
  printf("  sqlbmtsq -- Tablespace Query\n");
  printf("  sqlbtcq -- Tablespace Container Query\n");
  printf("  sqlbstsc -- Set Tablespace Containers\n");
  printf("  sqlefmem -- Free Memory\n");
  printf("  sqledrpd -- Drop Database\n");
  printf("TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.\n");

  printf("\n  Update \'%s\' database configuration:\n", dbAlias);
  printf("    - Disable the database configuration parameter LOGRETAIN \n");
  printf("        i.e., set LOGRETAIN = OFF/NO\n");

  /* initialize cfgParameters */
  /* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
     parameter 'logretain'; it is used to update the database configuration
     file */
  cfgParameters[0].flags = 0;
  cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
  cfgParameters[0].ptrvalue = (char *)&logretain;

  /* disable the database configuration parameter 'logretain' */
  logretain = SQLF_LOGRETAIN_DISABLE;

  /* initialize cfgStruct */
  cfgStruct.numItems = 1;
  cfgStruct.paramArray = cfgParameters;
  cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
  cfgStruct.dbname = dbAlias;

  /* get database configuration */
  db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
  DB2_API_CHECK("Db Log Retain -- Disable");

  /*****************************/
  /*    BACK UP THE DATABASE    */
```

```
| /*****************************/
|
| /* Calling up the routine for database backup */
| rc = DbBackup(dbAlias, user, pswd, serverWorkingPath, &backupStruct);
| CHECKRC(rc, "DbBackup");
|
| /*****************************/
| /*    RESTORE THE DATABASE    */
| /*****************************/
|
| strcpy(restoreTimestamp, backupStruct.oTimestamp);
|
| rtablespaceStruct.tablespaces = NULL;
| rtablespaceStruct.numTablespaces = 0;
|
| rmediaListStruct.locations = &serverWorkingPath;
| rmediaListStruct.numLocations = 1;
| rmediaListStruct.locationType = SQLU_LOCAL_MEDIA;
|
| restoreStruct.piSourceDBAlias = dbAlias;
| restoreStruct.piTargetDBAlias = restoredDbAlias;
| restoreStruct.piTimestamp = restoreTimestamp;
| restoreStruct.piTargetDBPath = NULL;
| restoreStruct.piReportFile = NULL;
| restoreStruct.piTablespaceList = &rtablespaceStruct;
| restoreStruct.piMediaList = &rmediaListStruct;
| restoreStruct.piUsername = user;
| restoreStruct.piPassword = pswd;
| restoreStruct.piNewLogPath = NULL;
| restoreStruct.piVendorOptions = NULL;
| restoreStruct.iVendorOptionsSize = 0;
| restoreStruct.iParallelism = 1;
| restoreStruct.iBufferSize = 1024;     /*  1024 x 4KB */
| restoreStruct.iNumBuffers = 2;
| restoreStruct.iOptions = DB2RESTORE_OFFLINE | DB2RESTORE_DB |
| DB2RESTORE_NODATALINK | DB2RESTORE_NOROLLFWD;
|
| printf("\n  Restoring a database ...\n");
| printf("    - source image alias     : %s\n", dbAlias);
| printf("    - source image time stamp: %s\n", restoreTimestamp);
| printf("    - target database        : %s\n", restoredDbAlias);
|
| restoreStruct.iCallerAction = DB2RESTORE_RESTORE_STORDEF;
|
| /* The API db2Restore is used to restore a database that has been backed
|    up using the API db2Backup. */
| db2Restore(db2Version810, &restoreStruct, &sqlca);
|
| /* If restoring to a different database and restoreDbAlias already exists,
|    SQLCODE 2529 is expected. */
| if (strcmp(dbAlias, restoredDbAlias))
| {
|   printf("\n  SQLCODE 2529 is expected if target database '%s' already exists\n",
|         restoredDbAlias);
| }
|
| EXPECTED_WARN_CHECK("database restore -- start");
|
| while (sqlca.sqlcode != 0)
| {
|   /* continue the restore operation */
|   printf("\n  Continuing the restore operation...\n");
|
|   /* depending on the sqlca.sqlcode value, user action may be
|      required, such as mounting a new tape */
|
|   if (sqlca.sqlcode == SQLUD_INACCESSABLE_CONTAINER)
```

```
    {
      /* redefine the table space container layout */
      printf("\n  Find and redefine inaccessable containers.\n");
      rc = InaccessableContainersRedefine(serverWorkingPath);
      CHECKRC(rc, "InaccessableContainersRedefine");
    }

    restoreStruct.iCallerAction = DB2RESTORE_CONTINUE;

    /* restore the database */
  db2Restore(db2Version810, &restoreStruct, &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }

  printf("\n  Restore finished.\n");

  /* drop the restored database */
  rc = DbDrop(restoredDbAlias);
  CHECKRC(rc, "DbDrop");

  return 0;
} /* DbBackupAndRedirectedRestore */

int InaccessableContainersRedefine(char serverWorkingPath[])
{
  struct sqlca sqlca;
  sqluint32 numTablespaces = 0;
  struct SQLB_TBSPQRY_DATA **ppTablespaces = NULL;
  sqluint32 numContainers = 0;
  struct SQLB_TBSCONTQRY_DATA *pContainers = NULL;
  int tspNb = 0;
  int contNb = 0;
  char pathSep[2] = { 0 };

  /* The API sqlbmtsq provides a one-call interface to the table space query
     data. The query data for all table spaces in the database is returned
     in an array. */
  sqlbmtsq(&sqlca,
           &numTablespaces, &ppTablespaces, SQLB_RESERVED1, SQLB_RESERVED2);
  DB2_API_CHECK("tablespaces -- get");

  /* refedine the inaccessable containers */
  for (tspNb = 0; tspNb < numTablespaces; tspNb++)
  {
    /* The API sqlbtcq provides a one-call interface to the table space
       container query data. The query data for all the containers in a table
       space, or for all containers in all table spaces, is returned in an
       array. */
    sqlbtcq(&sqlca, ppTablespaces[tspNb]->id, &numContainers, &pContainers);
    DB2_API_CHECK("tablespace containers -- get");

    for (contNb = 0; contNb < numContainers; contNb++)
    {
      if (!pContainers[contNb].ok)
      {
        /* redefine inaccessable container */
        printf("\n    Redefine inaccessable container:\n");
        printf("      - table space name: %s\n", ppTablespaces[tspNb]->name);
        printf("      - default container name: %s\n",
                   pContainers[contNb].name);
        if (strstr(pContainers[contNb].name, "/"))
        { /* UNIX */
          strcpy(pathSep, "/");
        }
  else
        { /* Intel */
          strcpy(pathSep, "\\");
```

```
                }
                switch (pContainers[contNb].contType)
                {
                  case SQLB_CONT_PATH:
                    printf("      - container type: path\n");

                    sprintf(pContainers[contNb].name, "%s%sSQLT%04d.%d",
                                                      serverWorkingPath, pathSep,
                            ppTablespaces[tspNb]->id, pContainers[contNb].id);
                    printf("      - new container name: %s\n",
                           pContainers[contNb].name);
              break;
                  case SQLB_CONT_DISK:
                  case SQLB_CONT_FILE:
            default:
                    printf("    Unknown container type.\n");
              break;
                }
              }

              /* The API sqlbstsc is used to set or redefine table space containers
                 while performing a 'redirected' restore of the database. */
              sqlbstsc(&sqlca,
                       SQLB_SET_CONT_FINAL_STATE,
                       ppTablespaces[tspNb]->id, numContainers, pContainers);
              DB2_API_CHECK("tablespace containers -- redefine");

              /* The API sqlefmem is used here to free memory allocated by DB2 for use
                 with the API sqlbtcq (Tablespace Container Query). */
              sqlefmem(&sqlca, pContainers);
              DB2_API_CHECK("tablespace containers memory -- free");
            }

            /* The API sqlefmem is used here to free memory allocated by DB2 for
               use with the API sqlbmtsq (Tablespace Query). */
            sqlefmem(&sqlca, ppTablespaces);
            DB2_API_CHECK("tablespaces memory -- free");

            return 0;
          } /* InaccessableContainersRedefine */
```

Beispielprogramm dbhistfile:

Die Beispieldateien für dbhistfile zeigen, wie ein Eintrag in einer Datenbankwiederherstellungsdatei gelesen und aktualisiert wird.

```
/*****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****************************************************************************
**
** SOURCE FILE NAME: dbhistfile.sqc
**
** SAMPLE: How to read and update a database recovery history file entry.
**
**         This program ends in ".sqc" even though it does not contain
**         embedded SQL statements. It links in  the embedded SQL utility
**         file for database connection and disconnection, so it needs the
```

```
** embedded SQL extension for the precompiler.
**
** DB2 APIs USED:
**        db2HistoryCloseScan -- Close Recovery History File Scan
**        db2HistoryGetEntry -- Get Next Recovery History File Entry
**        db2HistoryOpenScan -- Open Recovery History File Scan
**        db2HistoryUpdate -- Update Recovery History File
**
** OUTPUT FILE: dbhistfile.out (available in the online documentation)
*****************************************************************************
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
*****************************************************************************/
#include "utilrecov.c"

/* local function prototypes */
int DbRecoveryHistoryFileRead(char *);
int DbFirstRecoveryHistoryFileEntryUpdate(char *, char *, char *);
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *);
int HistoryEntryDisplay(struct db2HistoryData);
int HistoryEntryDataFieldsFree(struct db2HistoryData *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  sqluint16 savedLogRetainValue = 0;
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("\nTHIS SAMPLE SHOWS HOW TO READ A DATABASE RECOVERY HISTORY FILE \n");
  printf("AND UPDATE A RECOVERY HISTORY FILE ENTRY. \n");

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  rc = DbRecoveryHistoryFileRead(dbAlias);
  CHECKRC(rc, "DbRecoveryHistoryFileRead");

  rc = DbFirstRecoveryHistoryFileEntryUpdate(dbAlias, user, pswd);
  CHECKRC(rc, "DbFirstRecoveryHistoryFileEntryUpdate");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");
```

```
  return 0;
} /* end main */

int DbRecoveryHistoryFileRead(char dbAlias[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint32 numEntries = 0;
  sqluint16 recoveryHistoryFileHandle = 0;
  sqluint32 entryNb = 0;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;

  printf("\n*********************************************\n");
  printf("*** READ A DATABASE RECOVERY HISTORY FILE ***\n");
  printf("*********************************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2HistoryOpenScan -- Open Recovery History File Scan\n");
  printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry\n");
  printf("  db2HistoryCloseScan -- Close Recovery History File Scan\n");
  printf("TO READ A DATABASE RECOVERY HISTORY FILE.\n");

  /* initialize the data structures */
  dbHistoryOpenParam.piDatabaseAlias = dbAlias;
  dbHistoryOpenParam.piTimestamp = NULL;
  dbHistoryOpenParam.piObjectName = NULL;
  dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;
  dbHistoryEntryGetParam.pioHistData = &histEntryData;
  dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;

  rc = HistoryEntryDataFieldsAlloc(&histEntryData);
  CHECKRC(rc, "HistoryEntryDataFieldsAlloc");

  /*****************************************/
  /* OPEN THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("\n  Open recovery history file for '%s' database.\n", dbAlias);

  /* open the recovery history file to scan */
  db2HistoryOpenScan(db2Version810, &dbHistoryOpenParam, &sqlca);
  DB2_API_CHECK("database recovery history file -- open");
  numEntries = dbHistoryOpenParam.oNumRows;

  /* dbHistoryOpenParam.oHandle returns the handle for scan access */
  recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
  dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

  /*********************************************/
  /* READ AN ENTRY IN THE RECOVERY HISTORY FILE */
  /*********************************************/
  for (entryNb = 0; entryNb < numEntries; entryNb++)
  {
    printf("\n  Read entry number %u.\n", entryNb);

    /* get the next entry from the recovery history file */
  db2HistoryGetEntry(db2Version810, &dbHistoryEntryGetParam, &sqlca);
    DB2_API_CHECK("database recovery history file entry -- read")

    /* display the entries in the recovery history file */
    printf("\n  Display entry number %u.\n", entryNb);
  rc = HistoryEntryDisplay(histEntryData);
    CHECKRC(rc, "HistoryEntryDisplay");
  }

  /*****************************************/
```

```
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************/
  printf("\n  Close recovery history file for '%s' database.\n", dbAlias);

  /* The API db2HistoryCloseScan ends the recovery history file scan and
     frees DB2 resources required for the scan. */
  db2HistoryCloseScan(db2Version810, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);
  CHECKRC(rc, "HistoryEntryDataFieldsFree");

  return 0;
} /* DbRecoveryHistoryFileRead */

int DbFirstRecoveryHistoryFileEntryUpdate(char dbAlias[], char user[],
                                          char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint16 recoveryHistoryFileHandle = 0;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;
  char newLocation[DB2HISTORY_LOCATION_SZ + 1] = { 0 };
  char newComment[DB2HISTORY_COMMENT_SZ + 1] = { 0 };
  struct db2HistoryUpdateStruct dbHistoryUpdateParam;

  printf("\n*****************************************************\n");
  printf("***  UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY  ***\n");
  printf("*****************************************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2HistoryOpenScan -- Open Recovery History File Scan\n");
  printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry\n");
  printf("  db2HistoryUpdate -- Update Recovery History File\n");
  printf("  db2HistoryCloseScan -- Close Recovery History File Scan\n");
  printf("TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.\n");

  /* initialize data structures */
  dbHistoryOpenParam.piDatabaseAlias = dbAlias;
  dbHistoryOpenParam.piTimestamp = NULL;
  dbHistoryOpenParam.piObjectName = NULL;
  dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;

  dbHistoryEntryGetParam.pioHistData = &histEntryData;
  dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
  rc = HistoryEntryDataFieldsAlloc(&histEntryData);
  CHECKRC(rc, "HistoryEntryDataFieldsAlloc");

  /*******************************************/
  /* OPEN THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************/
  printf("\n  Open the recovery history file for '%s' database.\n",
         dbAlias);

  /* The API db2HistoryOpenScan starts a recovery history file scan */
  db2HistoryOpenScan(db2Version810, &dbHistoryOpenParam, &sqlca);
  DB2_API_CHECK("database recovery history file -- open");

  /* dbHistoryOpenParam.oHandle returns the handle for scan access */
  recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
  dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

  /*************************************************/
  /* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE */
  /*************************************************/
```

```
printf("\n  Read the first entry in the recovery history file.\n");

/* The API db2HistoryGetEntry gets the next entry from the recovery
   history file. */
db2HistoryGetEntry(db2Version810, &dbHistoryEntryGetParam, &sqlca);
DB2_API_CHECK("first recovery history file entry -- read");
printf("\n  Display the first entry.\n");

/* HistoryEntryDisplay is a support function used to display the entries
   in the recovery history file. */
rc = HistoryEntryDisplay(histEntryData);
CHECKRC(rc, "HistoryEntryDisplay");

/* update the first history file entry */
rc = DbConn(dbAlias, user, pswd);
CHECKRC(rc, "DbConn");

strcpy(newLocation, "this is the NEW LOCATION");
strcpy(newComment, "this is the NEW COMMENT");
printf("\n  Update the first entry in the history file:\n");
printf("    new location = '%s'\n", newLocation);
printf("    new comment = '%s'\n", newComment);

dbHistoryUpdateParam.piNewLocation = newLocation;
dbHistoryUpdateParam.piNewDeviceType = NULL;
dbHistoryUpdateParam.piNewComment = newComment;
dbHistoryUpdateParam.iEID.ioNode = histEntryData.oEID.ioNode;
dbHistoryUpdateParam.iEID.ioHID = histEntryData.oEID.ioHID;

/* The API db2HistoryUpdate can be used to update the location,
   device type, or comment in a history file entry. */

/* Call this API to update the location and comment of the first
   entry in the history file: */
db2HistoryUpdate(db2Version810, &dbHistoryUpdateParam, &sqlca);
DB2_API_CHECK("first history file entry -- update");

rc = DbDisconn(dbAlias);
CHECKRC(rc, "DbDisconn");

/*******************************************/
/* CLOSE THE DATABASE RECOVERY HISTORY FILE */
/*******************************************/
printf("\n  Close recovery history file for '%s' database.\n", dbAlias);

/* The API db2HistoryCloseScan ends the recovery history file scan and
   frees DB2 resources required for the scan. */
db2HistoryCloseScan(db2Version810, &recoveryHistoryFileHandle, &sqlca);
DB2_API_CHECK("database recovery history file -- close");

/*******************************************/
/* RE-OPEN THE DATABASE RECOVERY HISTORY FILE */
/*******************************************/
printf("\n  Open the recovery history file for '%s' database.\n",
       dbAlias);

/* starts a recovery history file scan */
db2HistoryOpenScan(db2Version810, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("database recovery history file -- open");

recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
printf("\n  Read the first recovery history file entry.\n");

/*************************************************************************/
/* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE AFTER MODIFICATION */
/*************************************************************************/
```

```
  db2HistoryGetEntry(db2Version810, &dbHistoryEntryGetParam, &sqlca);
  DB2_API_CHECK("first recovery history file entry -- read");

  printf("\n  Display the first entry.\n");
  rc = HistoryEntryDisplay(histEntryData);
  CHECKRC(rc, "HistoryEntryDisplay");

  /*****************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("\n  Close the recovery history file for '%s' database.\n",
         dbAlias);

  /* ends the recovery history file scan */
  db2HistoryCloseScan(db2Version810, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);
  CHECKRC(rc, "HistoryEntryDataFieldsFree");

  return 0;
} /* DbFirstRecoveryHistoryFileEntryUpdate */

/*****************************************************************************/
/* HistoryEntryDataFieldsAlloc                                               */
/* Allocates memory for all the fields in a database recovery history        */
/* file entry                                                                */
/*****************************************************************************/
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb = 0;

  strcpy(pHistEntryData->ioHistDataID, "SQLUHINF");

  pHistEntryData->oObjectPart.pioData = malloc(DB2HISTORY_OBJPART_SZ + 1);
  pHistEntryData->oObjectPart.iLength = DB2HISTORY_OBJPART_SZ + 1;

  pHistEntryData->oEndTime.pioData = malloc(DB2HISTORY_TIMESTAMP_SZ + 1);
  pHistEntryData->oEndTime.iLength = DB2HISTORY_TIMESTAMP_SZ + 1;

  pHistEntryData->oFirstLog.pioData = malloc(DB2HISTORY_LOGFILE_SZ + 1);
  pHistEntryData->oFirstLog.iLength = DB2HISTORY_LOGFILE_SZ + 1;

  pHistEntryData->oLastLog.pioData = malloc(DB2HISTORY_LOGFILE_SZ + 1);
  pHistEntryData->oLastLog.iLength = DB2HISTORY_LOGFILE_SZ + 1;

  pHistEntryData->oID.pioData = malloc(DB2HISTORY_ID_SZ + 1);
  pHistEntryData->oID.iLength = DB2HISTORY_ID_SZ + 1;

  pHistEntryData->oTableQualifier.pioData =
    malloc(DB2HISTORY_TABLE_QUAL_SZ + 1);
  pHistEntryData->oTableQualifier.iLength = DB2HISTORY_TABLE_QUAL_SZ + 1;

  pHistEntryData->oTableName.pioData = malloc(DB2HISTORY_TABLE_NAME_SZ + 1);
  pHistEntryData->oTableName.iLength = DB2HISTORY_TABLE_NAME_SZ + 1;

  pHistEntryData->oLocation.pioData = malloc(DB2HISTORY_LOCATION_SZ + 1);
  pHistEntryData->oLocation.iLength = DB2HISTORY_LOCATION_SZ + 1;

  pHistEntryData->oComment.pioData = malloc(DB2HISTORY_COMMENT_SZ + 1);
  pHistEntryData->oComment.iLength = DB2HISTORY_COMMENT_SZ + 1;

  pHistEntryData->oCommandText.pioData = malloc(DB2HISTORY_COMMAND_SZ + 1);
  pHistEntryData->oCommandText.iLength = DB2HISTORY_COMMAND_SZ + 1;
```

```
  pHistEntryData->poEventSQLCA =
    (struct sqlca *)malloc(sizeof(struct sqlca));

  pHistEntryData->poTablespace = (db2Char *)malloc(3 * sizeof(db2Char));
  for (tsNb = 0; tsNb < 3; tsNb++)
  {
    pHistEntryData->poTablespace[tsNb].pioData = malloc(18 + 1);
    pHistEntryData->poTablespace[tsNb].iLength = 18 + 1;
  }

  pHistEntryData->iNumTablespaces = 3;

  return 0;
} /* HistoryEntryDataFieldsAlloc */

/******************************************************************************/
/* HistoryEntryDisplay                                                        */
/* Displays the fields of an entry in the database recovery history file      */
/******************************************************************************/
int HistoryEntryDisplay(struct db2HistoryData histEntryData)
{
  int rc = 0;
  int bufLen = 0;
  char *buf = NULL;
  sqluint32 tsNb = 0;

  bufLen =
    MIN(histEntryData.oObjectPart.oLength,
 histEntryData.oObjectPart.iLength);
  buf = malloc(bufLen + 1);
  memcpy(buf, histEntryData.oObjectPart.pioData, bufLen);
  buf[bufLen] = '\0';
  printf("    object part: %s\n", buf);
  free(buf);

  bufLen =
    MIN(histEntryData.oEndTime.oLength, histEntryData.oEndTime.iLength);
  buf = malloc(bufLen + 1);
  memcpy(buf, histEntryData.oEndTime.pioData, bufLen);
  buf[bufLen] = '\0';
  printf("    end time: %s\n", buf);
  free(buf);

  bufLen =
    MIN(histEntryData.oFirstLog.oLength, histEntryData.oFirstLog.iLength);
  buf = malloc(bufLen + 1);
  memcpy(buf, histEntryData.oFirstLog.pioData, bufLen);
  buf[bufLen] = '\0';
  printf("    first log: %s\n", buf);
  free(buf);

  bufLen =
    MIN(histEntryData.oLastLog.oLength, histEntryData.oLastLog.iLength);
  buf = malloc(bufLen + 1);
  memcpy(buf, histEntryData.oLastLog.pioData, bufLen);
  buf[bufLen] = '\0';
  printf("    last log: %s\n", buf);
  free(buf);

  bufLen = MIN(histEntryData.oID.oLength, histEntryData.oID.iLength);
  buf = malloc(bufLen + 1);
  memcpy(buf, histEntryData.oID.pioData, bufLen);
  buf[bufLen] = '\0';
  printf("    ID: %s\n", buf);
  free(buf);

  bufLen =
```

```
                    MIN(histEntryData.oTableQualifier.oLength,
                  histEntryData.oTableQualifier.iLength);
                  buf = malloc(bufLen + 1);
                  memcpy(buf, histEntryData.oTableQualifier.pioData, bufLen);
                  buf[bufLen] = '\0';
                  printf("    table qualifier: %s\n", buf);
                  free(buf);

                  bufLen =
                    MIN(histEntryData.oTableName.oLength, histEntryData.oTableName.iLength);
                  buf = malloc(bufLen + 1);
                  memcpy(buf, histEntryData.oTableName.pioData, bufLen);
                  buf[bufLen] = '\0';
                  printf("    table name: %s\n", buf);
                  free(buf);

                  bufLen =
                    MIN(histEntryData.oLocation.oLength, histEntryData.oLocation.iLength);
                  buf = malloc(bufLen + 1);
                  memcpy(buf, histEntryData.oLocation.pioData, bufLen);
                  buf[bufLen] = '\0';
                  printf("    location: %s\n", buf);
                  free(buf);

                  bufLen =
                    MIN(histEntryData.oComment.oLength, histEntryData.oComment.iLength);
                  buf = malloc(bufLen + 1);
                  memcpy(buf, histEntryData.oComment.pioData, bufLen);
                  buf[bufLen] = '\0';
                  printf("    comment: %s\n", buf);
                  free(buf);

                  bufLen =
                    MIN(histEntryData.oCommandText.oLength,
                  histEntryData.oCommandText.iLength);

                  buf = malloc(bufLen + 1);
                  memcpy(buf, histEntryData.oCommandText.pioData, bufLen);
                  buf[bufLen] = '\0';
                  printf("    command text: %s\n", buf);
                  printf("    history file entry ID: %u\n", histEntryData.oEID.ioHID);
                  printf("    table spaces:\n");
                  free(buf);

                  for (tsNb = 0; tsNb < histEntryData.oNumTablespaces; tsNb++)
                  {
                    bufLen =
                      MIN(histEntryData.poTablespace[tsNb].oLength,
                   histEntryData.poTablespace[tsNb].iLength);
                    buf = malloc(bufLen + 1);
                    memcpy(buf, histEntryData.poTablespace[tsNb].pioData, bufLen);
                    buf[bufLen] = '\0';
                    printf("      %s\n", buf);
                    free(buf);
                  }

                  printf("    type of operation: %c\n", histEntryData.oOperation);
                  printf("    granularity of the operation: %c\n", histEntryData.oObject);
                  printf("    operation type: %c\n", histEntryData.oOptype);
                  printf("    entry status: %c\n", histEntryData.oStatus);
                  printf("    device type: %c\n", histEntryData.oDeviceType);
                  printf("    SQLCA:\n");
                  printf("      sqlcode: %ld\n", histEntryData.poEventSQLCA->sqlcode);

                  bufLen = SQLUDF_SQLSTATE_LEN;
                  buf = malloc(bufLen + 1);
                  memcpy(buf, histEntryData.poEventSQLCA->sqlstate, bufLen);
```

```
  buf[bufLen] = '\0';
  printf("      sqlstate: %s\n", buf);
  free(buf);

  bufLen = histEntryData.poEventSQLCA->sqlerrml;
  buf = malloc(bufLen + 1);
  memcpy(buf, histEntryData.poEventSQLCA->sqlerrmc, bufLen);
  buf[bufLen] = '\0';
  printf("      message: %s\n", buf);
  free(buf);

  return 0;
} /* HistoryEntryDisplay */

/******************************************************************************/
/* HistoryEntryDataFieldsFree                                                 */
/* Deallocates the memory for database recovery history file structures       */
/******************************************************************************/
int HistoryEntryDataFieldsFree(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb = 0;

  free(pHistEntryData->oObjectPart.pioData);
  free(pHistEntryData->oEndTime.pioData);
  free(pHistEntryData->oFirstLog.pioData);
  free(pHistEntryData->oLastLog.pioData);
  free(pHistEntryData->oID.pioData);
  free(pHistEntryData->oTableQualifier.pioData);
  free(pHistEntryData->oTableName.pioData);
  free(pHistEntryData->oLocation.pioData);
  free(pHistEntryData->oComment.pioData);
  free(pHistEntryData->oCommandText.pioData);
  free(pHistEntryData->poEventSQLCA);
  pHistEntryData->oObjectPart.pioData = NULL;
  pHistEntryData->oEndTime.pioData = NULL;
  pHistEntryData->oFirstLog.pioData = NULL;
  pHistEntryData->oLastLog.pioData = NULL;
  pHistEntryData->oID.pioData = NULL;
  pHistEntryData->oTableQualifier.pioData = NULL;
  pHistEntryData->oTableName.pioData = NULL;
  pHistEntryData->oLocation.pioData = NULL;
  pHistEntryData->oComment.pioData = NULL;
  pHistEntryData->oCommandText.pioData = NULL;
  pHistEntryData->poEventSQLCA = NULL;

  for (tsNb = 0; tsNb < 3; tsNb++)
  {
    free(pHistEntryData->poTablespace[tsNb].pioData);
    pHistEntryData->poTablespace[tsNb].pioData = NULL;
  }

  free(pHistEntryData->poTablespace);
  pHistEntryData->poTablespace = NULL;

  return 0;
} /* HistoryEntryDataFieldsFree */
```

Beispielprogramm dblogconn:

Die Beispieldateien für `dblogconn` zeigen, wie Datenbankprotokolldateien gelesen werden, wenn eine Datenbankverbindung besteht.

```
/****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
```

```
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
******************************************************************************
**
** SOURCE FILE NAME: dblogconn.sqc
**
** SAMPLE: How to read database log files asynchronously with a database
**         connection
**
**         Note:
**           You must be initially disconnected from the sample database
**           to run this program. To ensure you are, enter 'db2 connect
**           reset' on the command line prior to running dblogconn.
**
** DB2 API USED:
**        db2CfgSet -- Set Configuration
**        db2ReadLog -- Asynchronous Read Log
**
** SQL STATEMENTS USED:
**         ALTER TABLE
**         COMMIT
**         DELETE
**         INSERT
**         ROLLBACK
**         CONNECT RESET
**
** OUTPUT FILE: dblogconn.out (available in the online documentation)
******************************************************************************
**
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
******************************************************************************/
#include "utilrecov.c"
#include "utilemb.h"

/* local function prototypes */
int DbLogRecordsForCurrentConnectionRead(char *, char *, char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  sqluint16 savedLogRetainValue = 0;
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };
```

```
  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("\nTHIS SAMPLE SHOWS HOW TO READ DATABASE LOGS ASYNCHRONOUSLY\n");
  printf("WITH A DATABASE CONNECTION.\n");

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  /* call the function to do asynchronous log read */
  rc = DbLogRecordsForCurrentConnectionRead(dbAlias,
                                            user, pswd, serverWorkingPath);
  CHECKRC(rc, "DbLogRecordsForCurrentConnectionRead");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");

  return 0;
} /* end main */

int DbLogRecordsForCurrentConnectionRead(char dbAlias[],
                                         char user[],
                         char pswd[],
                                        char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  unsigned short logretain = 0;

  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32 backupImageSize = 0;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;

  SQLU_LSN startLSN;
  SQLU_LSN endLSN;
  char      *logBuffer = NULL;
  sqluint32 logBufferSize = 0;
  db2ReadLogInfoStruct readLogInfo;
  db2ReadLogStruct     readLogInput;
  int i = 0;

  printf("\n*****************************\n");
  printf("*** ASYNCHRONOUS READ LOG ***\n");
  printf("*****************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2CfgSet -- Set Configuration\n");
  printf("  db2Backup -- Backup Database\n");
  printf("  db2ReadLog -- Asynchronous Read Log\n");
  printf("AND THE SQL STATEMENTS:\n");
  printf("  CONNECT\n");
  printf("  ALTER TABLE\n");
  printf("  COMMIT\n");
  printf("  INSERT\n");
```

```
| printf("  DELETE\n");
| printf("  ROLLBACK\n");
| printf("  CONNECT RESET\n");
| printf("TO READ LOG RECORDS FOR THE CURRENT CONNECTION.\n");
|
| printf("\n  Update \'%s\' database configuration:\n", dbAlias);
| printf("    - Enable the database configuration parameter LOGRETAIN \n");
| printf("        i.e., set LOGRETAIN = RECOVERY/YES\n");
|
| /* initialize cfgParameters */
| cfgParameters[0].flags = 0;
| cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
| cfgParameters[0].ptrvalue = (char *)&logretain;
|
| /* enable LOGRETAIN */
| logretain = SQLF_LOGRETAIN_RECOVERY;
|
| /* initialize cfgStruct */
| cfgStruct.numItems = 1;
| cfgStruct.paramArray = cfgParameters;
| cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
| cfgStruct.dbname = dbAlias;
|
| /* get database configuration */
| db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
| DB2_API_CHECK("Db Log Retain -- Enable");
|
| tablespaceStruct.tablespaces = NULL;
| tablespaceStruct.numTablespaces = 0;
|
| mediaListStruct.locations = &serverWorkingPath;
| mediaListStruct.numLocations = 1;
| mediaListStruct.locationType = SQLU_LOCAL_MEDIA;
|
| /* Calling up the routine for database backup */
| rc = DbBackup(dbAlias, user, pswd, serverWorkingPath, &backupStruct);
| CHECKRC(rc, "DbBackup");
|
| /* connect to the database */
| rc = DbConn(dbAlias, user, pswd);
| CHECKRC(rc, "DbConn");
|
| /* invoke SQL statements to fill database log */
| printf("\n  Invoke the following SQL statements:\n"
|        "    ALTER TABLE emp_resume DATA CAPTURE CHANGES;\n"
|        "    COMMIT;\n"
|        "    INSERT INTO emp_resume\n"
|        "      VALUES('000777', 'ascii', 'This is a new resume.');\n"
|        "             ('777777', 'ascii', 'This is another new resume');\n"
|        "    COMMIT;\n"
|        "    DELETE FROM emp_resume WHERE empno = '000777';\n"
|        "    DELETE FROM emp_resume WHERE empno = '777777';\n"
|        "    COMMIT;\n"
|        "    DELETE FROM emp_resume WHERE empno = '000140';\n"
|        "    ROLLBACK;\n"
|        "    ALTER TABLE emp_resume DATA CAPTURE NONE;\n" "    COMMIT;\n");
|
| EXEC SQL ALTER TABLE emp_resume DATA CAPTURE CHANGES;
| EMB_SQL_CHECK("SQL statement 1 -- invoke");
|
| EXEC SQL COMMIT;
| EMB_SQL_CHECK("SQL statement 2 -- invoke");
|
| EXEC SQL INSERT INTO emp_resume
|   VALUES('000777', 'ascii', 'This is a new resume.'),
|         ('777777', 'ascii', 'This is another new resume');
| EMB_SQL_CHECK("SQL statement 3 -- invoke");
```

```
|
|                         EXEC SQL COMMIT;
|                         EMB_SQL_CHECK("SQL statement 4 -- invoke");
|
|                         EXEC SQL DELETE FROM emp_resume WHERE empno = '000777';
|                         EMB_SQL_CHECK("SQL statement 5 -- invoke");
|
|                         EXEC SQL DELETE FROM emp_resume WHERE empno = '777777';
|                         EMB_SQL_CHECK("SQL statement 6 -- invoke");
|
|                         EXEC SQL COMMIT;
|                         EMB_SQL_CHECK("SQL statement 7 -- invoke");
|
|                         EXEC SQL DELETE FROM emp_resume WHERE empno = '000140';
|                         EMB_SQL_CHECK("SQL statement 8 -- invoke");
|
|                         EXEC SQL ROLLBACK;
|                         EMB_SQL_CHECK("SQL statement 9 -- invoke");
|
|                         EXEC SQL ALTER TABLE emp_resume DATA CAPTURE NONE;
|                         EMB_SQL_CHECK("SQL statement 10 -- invoke");
|
|                         EXEC SQL COMMIT;
|                         EMB_SQL_CHECK("SQL statement 11 -- invoke");
|
|                         printf("\n  Start reading database log.\n");
|
|                         logBuffer = NULL;
|                         logBufferSize = 0;
|
|                         /*
|                          * The API db2ReadLog (Asynchronous Read Log) is used to extract
|                          * records from the database logs, and to query the log manager for
|                          * current log state information. This API can only be used on
|                          * recoverable databases.
|                          */
|
|                         /* Query the log manager for current log state information. */
|                         readLogInput.iCallerAction   = DB2READLOG_QUERY;
|                         readLogInput.piStartLSN      = NULL;
|                         readLogInput.piEndLSN        = NULL;
|                         readLogInput.poLogBuffer     = NULL;
|                         readLogInput.iLogBufferSize  = 0;
|                         readLogInput.iFilterOption  = DB2READLOG_FILTER_ON;
|                         readLogInput.poReadLogInfo   = &readLogInfo;
|
|                         db2ReadLog(db2Version810, &readLogInput, &sqlca);
|                         DB2_API_CHECK("database log info -- get");
|
|                         logBufferSize = 64 * 1024;    /* Maximum size of a log buffer */
|                         logBuffer = (char *)malloc(logBufferSize);
|
|                         memcpy(&startLSN, &(readLogInfo.initialLSN), sizeof(startLSN));
|                         memcpy(&endLSN, &(readLogInfo.nextStartLSN), sizeof(endLSN));
|
|                         /*
|                          * Extract a log record from the database logs, and read the first
|                          * log sequence asynchronously.
|                          */
|                         readLogInput.iCallerAction   = DB2READLOG_READ;
|                         readLogInput.piStartLSN      = &startLSN;
|                         readLogInput.piEndLSN        = &endLSN;
|                         readLogInput.poLogBuffer     = logBuffer;
|                         readLogInput.iLogBufferSize  = logBufferSize;
|                         readLogInput.iFilterOption  = DB2READLOG_FILTER_ON;
|                         readLogInput.poReadLogInfo   = &readLogInfo;
|
```

```
  db2ReadLog(db2Version810, &readLogInput, &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
      DB2_API_CHECK("database logs -- read");
  }
  else
  {
    if (readLogInfo.logRecsWritten == 0)
    {
      printf("\n  Database log empty.\n");
    }
  }

    /* display log buffer */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
  CHECKRC(rc, "LogBufferDisplay");

  while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    /* read the next log sequence */

  memcpy(&startLSN, &(readLogInfo.nextStartLSN), sizeof(startLSN));

    /*
     * Extract a log record from the database logs, and read the
     * next log sequence asynchronously.
     */
    db2ReadLog(db2Version810, &readLogInput, &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
    {
      DB2_API_CHECK("database logs -- read");
    }
    /* display log buffer */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
    CHECKRC(rc, "LogBufferDisplay");
  }

  /* free the log buffer */
  free(logBuffer);
  logBuffer = NULL;
  logBufferSize = 0;

  /* disconnect from the database */
  rc = DbDisconn(dbAlias);
  CHECKRC(rc, "DbDisconn");

  return 0;
} /* DbLogRecordsForCurrentConnectionRead */
```

Beispielprogramm dblognoconn:

Die Beispieldateien für dblognoconn zeigen, wie Datenbankprotokolldateien gelesen werden, wenn keine Datenbankverbindung besteht.

```
/****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
****************************************************************************
**
```

```
** SOURCE FILE NAME: dblognoconn.sqc
**
** SAMPLE: How to read database log files asynchronously
**         with no database connection
**
**         This program ends in ".sqc" even though it does not contain
**         embedded SQL statements. It links in  the embedded SQL utility
**         file for database connection and disconnection, so it needs the
**         embedded SQL extension for the precompiler.
**
**         Note:
**           You must be disconnected from the sample database to run
**           this program. To ensure you are, enter 'db2 connect reset'
**           on the command line prior to running dblognoconn.
**
** DB2 API USED:
**        db2CfgSet -- Set Configuration
**         db2ReadLogNoConnInit -- Read log without a db connection
**        db2ReadLog -- Asynchronous Read Log
**
** OUTPUT FILE: dblognoconn.out (available in the online documentation)
******************************************************************************
**
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
******************************************************************************/
#include "utilrecov.c"
#include "utilemb.h"

/* local function prototypes */
int DbReadLogRecordsNoConn(char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  sqluint16 savedLogRetainValue = 0;
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("\nTHIS SAMPLE SHOWS HOW TO READ DATABASE LOGS ASYNCHRONOUSLY\n");
  printf("WITH NO DATABASE CONNECTION.\n");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");
```

```
  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);
  CHECKRC(rc, "DbRecoveryHistoryFilePrune");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");

  rc = DbReadLogRecordsNoConn(dbAlias);
  CHECKRC(rc, "DbReadLogRecordsNoConn");

  return 0;
} /* end main */

int DbReadLogRecordsNoConn(char dbAlias[])
{
  int rc = 0;
  struct sqlca sqlca;
  char logPath[SQL_PATH_SZ + 1] = { 0 };
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  char      nodeName[] = "NODE0000\0";
  db2Uint32 readLogMemSize = 0;
  char      *readLogMemory = NULL;
  struct db2ReadLogNoConnInitStruct readLogInit;
  struct db2ReadLogNoConnInfoStruct readLogInfo;
  struct db2ReadLogNoConnStruct readLogInput;
  SQLU_LSN startLSN;
  SQLU_LSN endLSN;
  char      *logBuffer = NULL;
  db2Uint32 logBufferSize = 0;
  struct db2ReadLogNoConnTermStruct readLogTerm;

  printf("\n*********************************\n");
  printf("***  NO DB CONNECTION READ LOG  ***\n");
  printf("*********************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2ReadLogNoConnInit -- Initialize No Db Connection Read Log\n");
  printf("  db2ReadLogNoConn -- No Db Connection Read Log\n");
  printf("  db2ReadLogNoConnTerm -- Terminate No Db Connection Read Log\n");
  printf("TO READ LOG RECORDS FROM A DATABASE LOG DIRECTORY.\n");

  /* Determine the logpath to read log files from */
  cfgParameters[0].flags = 0;
  cfgParameters[0].token = SQLF_DBTN_LOGPATH;
  cfgParameters[0].ptrvalue =
    (char *)malloc((SQL_PATH_SZ + 1) * sizeof(char));

  /* Initialize cfgStruct */
  cfgStruct.numItems = 1;
  cfgStruct.paramArray = cfgParameters;
  cfgStruct.flags = db2CfgDatabase;
  cfgStruct.dbname = dbAlias;

  db2CfgGet(db2Version810, (void *)&cfgStruct, &sqlca);
  DB2_API_CHECK("log path -- get");

  strcpy(logPath, cfgParameters[0].ptrvalue);
  free(cfgParameters[0].ptrvalue);
  cfgParameters[0].ptrvalue = NULL;

  /*
   * First we must allocate memory for the API's control blocks and log
   * buffer
   */
  readLogMemSize = 4 * 4096;
  readLogMemory = (char*)malloc(readLogMemSize);
```

```
/* Invoke the initialization API to set up the control blocks */
readLogInit.iFilterOption       = DB2READLOG_FILTER_ON;
readLogInit.piLogFilePath       = logPath;
readLogInit.piOverflowLogPath   = NULL;
readLogInit.iRetrieveLogs = DB2READLOG_RETRIEVE_OFF;
readLogInit.piDatabaseName      = dbAlias;
readLogInit.piNodeName          = nodeName;
readLogInit.iReadLogMemoryLimit = readLogMemSize;
readLogInit.poReadLogMemPtr     = &readLogMemory;

db2ReadLogNoConnInit(db2Version810, &readLogInit, &sqlca);
if (sqlca.sqlcode != SQLU_RLOG_LSNS_REUSED)
{
  DB2_API_CHECK("database logs no db conn -- initialization");
}
/* Query for the current log information */
readLogInput.iCallerAction   = DB2READLOG_QUERY;
readLogInput.piStartLSN      = NULL;
readLogInput.piEndLSN        = NULL;
readLogInput.poLogBuffer     = NULL;
readLogInput.iLogBufferSize  = 0;
readLogInput.piReadLogMemPtr = readLogMemory;
readLogInput.poReadLogInfo   = &readLogInfo;

db2ReadLogNoConn(db2Version810, &readLogInput, &sqlca);
if (sqlca.sqlcode != 0)
{
  DB2_API_CHECK("database logs no db conn -- query");
}
/* Read some log records */
logBufferSize = 64 * 1024; /* Maximum size of a log buffer */
logBuffer = (char *)malloc(logBufferSize);

memcpy(&startLSN, &(readLogInfo.nextStartLSN), sizeof(startLSN));
endLSN.lsnWord[0] = 0xffff;
endLSN.lsnWord[1] = 0xffff;
endLSN.lsnWord[2] = 0xffff;

readLogInput.iCallerAction   = DB2READLOG_READ;
readLogInput.piStartLSN      = &startLSN;
readLogInput.piEndLSN        = &endLSN;
readLogInput.poLogBuffer     = logBuffer;
readLogInput.iLogBufferSize  = logBufferSize;
readLogInput.piReadLogMemPtr = readLogMemory;
readLogInput.poReadLogInfo   = &readLogInfo;

db2ReadLogNoConn(db2Version810, &readLogInput, &sqlca);
  if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
{
    DB2_API_CHECK("database logs no db conn -- read");
}
else
{
  if (readLogInfo.logRecsWritten == 0)
  {
    printf("\n  Database log empty.\n");
  }
}

/* Display the log records read */
  rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
CHECKRC(rc, "LogBufferDisplay");

while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
{
  /* read the next log sequence */
```

```
  memcpy(&startLSN, &(readLogInfo.nextStartLSN), sizeof(startLSN));

    /*
     * Extract a log record from the database logs, and read the
     * next log sequence asynchronously.
     */
    db2ReadLogNoConn(db2Version810, &readLogInput, &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
    {
      DB2_API_CHECK("database logs no db conn -- read");
    }
    /* display log buffer */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
    CHECKRC(rc, "LogBufferDisplay");
  }

  printf("\nRead to end of logs.\n\n");
  free(logBuffer);
  logBuffer = NULL;
  logBufferSize = 0;

  readLogTerm.poReadLogMemPtr = &readLogMemory;

  db2ReadLogNoConnTerm(db2Version810, &readLogTerm, &sqlca);
    DB2_API_CHECK("database logs no db conn -- terminate");

  return 0;
} /* DbReadLogRecordsNoConn */
```

Beispielprogramm dbrestore:

Die Beispieldateien für dbrestore zeigen, wie eine Datenbank aus einem Sicherungsimage wiederhergestellt wird.

```
/*****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****************************************************************************
**
** SOURCE FILE NAME: dbrestore.sqc
**
** SAMPLE: How to restore a database from a backup
**
**         This program ends in ".sqc" even though it does not contain
**         embedded SQL statements. It links in  the embedded SQL utility
**         file for database connection and disconnection, so it needs the
**         embedded SQL extension for the precompiler.
**
**         Note:
**           You must be disconnected from the sample database to run
**           this program. To ensure you are, enter 'db2 connect reset'
**           on the command line prior to running dbrestore.
**
** DB2 API USED:
**        db2CfgSet -- Set Configuration
**        db2Restore -- Restore Database
**
** OUTPUT FILE: dbrestore.out (available in the online documentation)
*****************************************************************************
```

```
**
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
******************************************************************************/
#include "utilrecov.c"

/* local function prototypes */
int DbBackupAndRestore(char *, char *, char *, char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  char restoredDbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("\nTHIS SAMPLE SHOWS HOW TO RESTORE A DATABASE FROM A\n");
  printf("BACKUP.\n");

  strcpy(restoredDbAlias, dbAlias);

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  printf("\nNOTE: Backup images will be created on the server\n");
  printf("      in the directory %s,\n", serverWorkingPath);
  printf("      and will not be deleted by the program.\n");

  /* prune the recovery history file */
  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);
  CHECKRC(rc, "DbRecoveryHistoryFilePrune");

  rc = DbBackupAndRestore(dbAlias,
                          restoredDbAlias, user, pswd, serverWorkingPath);
  CHECKRC(rc, "DbBackupAndRestore");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");
```

```
  return 0;
} /* end main */

int DbBackupAndRestore(char dbAlias[],
                       char restoredDbAlias[], char user[],
                                 char pswd[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  unsigned short logretain = 0;
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1] = { 0 };

  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32 backupImageSize = 0;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;

  printf("\n***************************************\n");
  printf("***  BACK UP AND RESTORE A DATABASE  ***\n");
  printf("***************************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2CfgSet -- Set Configuration\n");
  printf("  db2Backup -- Backup Database\n");
  printf("  db2Restore -- Restore Database\n");
  printf("TO BACK UP AND RESTORE A DATABASE.\n");

  printf("\n  Update \'%s\' database configuration:\n", dbAlias);
  printf("    - Disable the database configuration parameter LOGRETAIN\n");
  printf("        i.e., set LOGRETAIN = OFF/NO\n");

  /* initialize cfgParameters */
  /* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
     parameter 'logretain'; it is used to update the database configuration
     file */
  cfgParameters[0].flags = 0;
  cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
  cfgParameters[0].ptrvalue = (char *)&logretain;

  /* disable the database configuration parameter 'logretain' */
  logretain = SQLF_LOGRETAIN_DISABLE;

  /* initialize cfgStruct */
  cfgStruct.numItems = 1;
  cfgStruct.paramArray = cfgParameters;
  cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
  cfgStruct.dbname = dbAlias;

  /* set database configuration */
  db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
  DB2_API_CHECK("Db Log Retain -- Disable");

  /*****************************/
  /*    BACKUP THE DATABASE    */
  /*****************************/

  /* Calling up the routine for database backup */
  rc = DbBackup(dbAlias, user, pswd, serverWorkingPath, &backupStruct);
  CHECKRC(rc, "DbBackup");

  /*****************************/
  /*    RESTORE THE DATABASE   */
  /*****************************/
```

```
                        strcpy(restoreTimestamp, backupStruct.oTimestamp);

                        printf("\n  Restoring a database ...\n");
                        printf("    - source image alias     : %s\n", dbAlias);
                        printf("    - source image time stamp: %s\n", restoreTimestamp);
                        printf("    - target database        : %s\n", restoredDbAlias);

                        rtablespaceStruct.tablespaces = NULL;
                        rtablespaceStruct.numTablespaces = 0;

                        rmediaListStruct.locations = &serverWorkingPath;
                        rmediaListStruct.numLocations = 1;
                        rmediaListStruct.locationType = SQLU_LOCAL_MEDIA;

                        restoreStruct.piSourceDBAlias = dbAlias;
                        restoreStruct.piTargetDBAlias = restoredDbAlias;

                        restoreStruct.piTimestamp = restoreTimestamp;
                        restoreStruct.piTargetDBPath = NULL;
                        restoreStruct.piReportFile = NULL;
                        restoreStruct.piTablespaceList = &rtablespaceStruct;
                        restoreStruct.piMediaList = &rmediaListStruct;
                        restoreStruct.piUsername = user;
                        restoreStruct.piPassword = pswd;
                        restoreStruct.piNewLogPath = NULL;
                        restoreStruct.piVendorOptions = NULL;
                        restoreStruct.iVendorOptionsSize = 0;
                        restoreStruct.iParallelism = 1;
                        restoreStruct.iBufferSize = 1024;     /*  1024 x 4KB */
                        restoreStruct.iNumBuffers = 2;
                        restoreStruct.iCallerAction = DB2RESTORE_RESTORE;
                        restoreStruct.iOptions =
                          DB2RESTORE_OFFLINE | DB2RESTORE_DB | DB2RESTORE_NODATALINK |
                          DB2RESTORE_NOROLLFWD;

                        /* The API db2Restore is used to restore a database that has been backed
                           up using the API db2Backup. */
                        db2Restore(db2Version810, &restoreStruct, &sqlca);
                        EXPECTED_WARN_CHECK("database restore -- start");

                        while (sqlca.sqlcode != 0)
                        {
                          /* continue the restore operation */
                          printf("\n  Continuing the restore operation...\n");

                          /* depending on the sqlca.sqlcode value, user action may be
                             required, such as mounting a new tape */

                          restoreStruct.iCallerAction = DB2RESTORE_CONTINUE;

                          /* restore the database */
                        db2Restore(db2Version810, &restoreStruct, &sqlca);
                          DB2_API_CHECK("database restore -- continue");
                        }

                        printf("\n  Restore finished.\n");

                        return 0;
                      } /* DbBackupAndRestore */
```

Beispielprogramm dbrollfwd:

Die Beispieldateien für dbrollfwd zeigen, wie nach einer Wiederherstellungsoperation eine aktualisierende Wiederherstellung ausgeführt wird.

```
/****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
****************************************************************************
**
** SOURCE FILE NAME: dbrollfwd.sqc
**
** SAMPLE: How to perform rollforward after restore of a database
**
**         This program ends in ".sqc" even though it does not contain
**         embedded SQL statements. It links in  the embedded SQL utility
**         file for database connection and disconnection, so it needs the
**         embedded SQL extension for the precompiler.
**
**         Note:
**           You must be disconnected from the sample database to run
**           this program. To ensure you are, enter 'db2 connect reset'
**           on the command line prior to running dbrollfwd.
**
** DB2 APIs USED:
**        db2CfgSet -- Set Configuration
**        db2Restore -- Restore Database
**        db2Rollforward -- Rollforward Database
**
** OUTPUT FILE: dbrollfwd.out (available in the online documentation)
****************************************************************************
**
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
****************************************************************************/
#include "utilrecov.c"

/* local function prototypes */
int DbBackupRestoreAndRollforward(char *, char *, char *, char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
```

```
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  char rolledForwardDbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("\nTHIS SAMPLE SHOWS HOW TO PERFORM ROLLFORWARD AFTER\n");
  printf("RESTORE OF A DATABASE.\n");
  strcpy(rolledForwardDbAlias, "RFDB");

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  printf("\nNOTE: Backup images will be created on the server\n");
  printf("      in the directory %s,\n", serverWorkingPath);
  printf("      and will not be deleted by the program.\n");

  rc = DbBackupRestoreAndRollforward(dbAlias, rolledForwardDbAlias, user,
                                     pswd, serverWorkingPath);
  CHECKRC(rc, "DbBackupRestoreAndRollforward");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");

  return 0;
} /* end main */

int DbBackupRestoreAndRollforward(char dbAlias[],
                                  char rolledForwardDbAlias[],
                                  char user[], char pswd[],
                                         char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  unsigned short logretain = 0;
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1] = { 0 };
  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32  backupImageSize;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;
  db2RfwdInputStruct   rfwdInput;
  db2RfwdOutputStruct  rfwdOutput;
  db2RollforwardStruct  rfwdStruct;
  char rollforwardAppId[SQLU_APPLID_LEN + 1] = { 0 };
  sqlint32 numReplies = 0;
  struct sqlurf_info nodeInfo;

  printf("\n****************************\n");
  printf("***  ROLLFORWARD RECOVERY  ***\n");
  printf("****************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2CfgSet -- Set Configuration\n");
```

```
printf("  db2Backup -- Backup Database\n");
printf("  sqlecrea -- Create Database\n");
printf("  db2Restore -- Restore Database\n");
printf("  db2Rollforward -- Rollforward Database\n");
printf("  sqledrpd -- Drop Database\n");
printf("TO BACK UP, RESTORE, AND ROLLFORWARD A DATABASE.\n");
printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Enable the configuration parameter LOGRETAIN \n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES\n");

/* initialize cfgParameters */
cfgParameters[0].flags = 0;
cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
cfgParameters[0].ptrvalue = (char *)&logretain;

/* enable the configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* initialize cfgStruct */
cfgStruct.numItems = 1;
cfgStruct.paramArray = cfgParameters;
cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
cfgStruct.dbname = dbAlias;

/* get database configuration */
db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/*****************************/
/*    BACKUP THE DATABASE    */
/*****************************/

/* Calling the routine for database backup */
rc = DbBackup(dbAlias, user, pswd, serverWorkingPath, &backupStruct);
CHECKRC(rc, "DbBackup");

/* To restore a remote database, you will first need to create an empty database
   if the client's code page is different from the server's code page.
   If this is the case, uncomment the call to DbCreate(). It will create
   an empty database on the server with the server's code page. */

/*
rc = DbCreate(dbAlias, rolledForwardDbAlias);
CHECKRC(rc, "DbCreate");
*/

/*****************************/
/*    RESTORE THE DATABASE   */
/*****************************/
strcpy(restoreTimestamp, backupStruct.oTimestamp);
rtablespaceStruct.tablespaces = NULL;
rtablespaceStruct.numTablespaces = 0;
rmediaListStruct.locations = &serverWorkingPath;
rmediaListStruct.numLocations = 1;
rmediaListStruct.locationType = SQLU_LOCAL_MEDIA;
restoreStruct.piSourceDBAlias = dbAlias;
restoreStruct.piTargetDBAlias = rolledForwardDbAlias;
restoreStruct.piTimestamp = restoreTimestamp;
restoreStruct.piTargetDBPath = NULL;
restoreStruct.piReportFile = NULL;
restoreStruct.piTablespaceList = &rtablespaceStruct;
restoreStruct.piMediaList = &rmediaListStruct;
restoreStruct.piUsername = user;
restoreStruct.piPassword = pswd;
restoreStruct.piNewLogPath = NULL;
restoreStruct.piVendorOptions = NULL;
restoreStruct.iVendorOptionsSize = 0;
```

```
restoreStruct.iParallelism = 1;
restoreStruct.iBufferSize = 1024;     /*  1024 x 4KB */
restoreStruct.iNumBuffers = 2;
restoreStruct.iCallerAction = DB2RESTORE_RESTORE;
restoreStruct.iOptions =
DB2RESTORE_OFFLINE | DB2RESTORE_DB | DB2RESTORE_NODATALINK |
DB2RESTORE_ROLLFWD;

printf("\n  Restoring a database ...\n");
printf("    - source image alias     : %s\n", dbAlias);
printf("    - source image time stamp: %s\n", restoreTimestamp);
printf("    - target database        : %s\n", rolledForwardDbAlias);

/* The API db2Restore is used to restore a database that has been backed
   up using the API db2Backup. */
db2Restore(db2Version810, &restoreStruct, &sqlca);
DB2_API_CHECK("database restore -- start");
while (sqlca.sqlcode != 0)

{

  /* continue the restore operation */
  printf("\n  Continuing the restore operation...\n");

  /* Depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */
  restoreStruct.iCallerAction = DB2RESTORE_CONTINUE;

  /* restore the database */
db2Restore(db2Version810, &restoreStruct, &sqlca);
  DB2_API_CHECK("database restore -- continue");
}
printf("\n  Restore finished.\n");

/*****************************/
/*    ROLLFORWARD RECOVERY    */
/*****************************/
printf("\n  Rolling forward database '%s'...\n", rolledForwardDbAlias);
rfwdInput.iVersion = SQLUM_RFWD_VERSION;
rfwdInput.piDbAlias = rolledForwardDbAlias;
rfwdInput.iCallerAction = DB2ROLLFORWARD_RFWD_STOP;
rfwdInput.piStopTime = SQLUM_INFINITY_TIMESTAMP;
rfwdInput.piUserName = user;
rfwdInput.piPassword = pswd;
rfwdInput.piOverflowLogPath = serverWorkingPath;
rfwdInput.iNumChngLgOvrflw = 0;
rfwdInput.piChngLogOvrflw = NULL;
rfwdInput.iConnectMode = DB2ROLLFORWARD_OFFLINE;
rfwdInput.piTablespaceList = NULL;
rfwdInput.iAllNodeFlag = DB2_ALL_NODES;
rfwdInput.iNumNodes = 0;
rfwdInput.piNodeList = NULL;
rfwdInput.piDroppedTblID = NULL;
rfwdInput.piExportDir = NULL;
rfwdInput.iNumNodeInfo = 1;
rfwdInput.iRollforwardFlags = DB2ROLLFORWARD_EMPTY_FLAG;
rfwdOutput.poApplicationId = rollforwardAppId;
rfwdOutput.poNumReplies = &numReplies;
rfwdOutput.poNodeInfo = &nodeInfo;
rfwdStruct.piRfwdInput = &rfwdInput;
rfwdStruct.poRfwdOutput = &rfwdOutput;

/* rollforward database */
/* The API db2Rollforward rollforward recovers a database by
   applying transactions recorded in the database log files. */
db2Rollforward(db2Version810, &rfwdStruct, &sqlca);
DB2_API_CHECK("rollforward -- start");
```

```
|                         printf("  Rollforward finished.\n");
|
|                         /* drop the restored database */
|                         rc = DbDrop(rolledForwardDbAlias);
|                         CHECKRC(rc, "DbDrop");
|                         return 0;
|                       } /* DbBackupRestoreAndRollforward */
```

Anhang F. Tivoli Storage Manager

Wenn Sie die Befehle BACKUP DATABASE oder RESTORE DATABASE aufrufen, können Sie angeben, dass Sie die Sicherungs- oder Wiederherstellungsoperation für die Datenbank bzw. den Tabellenbereich mit dem Produkt Tivoli Storage Manager (TSM) verwalten wollen. Der erforderliche Mindeststand der TSM-Client-API ist Version 4.2.0 außer auf folgenden Systemen:

- 64-Bit-Solaris-Systeme, für die Version 4.2.1 der TSM-Client-API erforderlich ist.
- 64-Bit-Windows NT-Betriebssysteme, für die Version 5.1 der TSM-Client-API erforderlich ist.
- SuSE Linux Enterprise Server 8 für IA64 und LinuxPPC, für die mindestens Version 5.1.5 der TSM-Client-API erforderlich ist.
- 64-Bit-Linux auf AMD-Opteron-Systemen, für die mindestens Version 5.2.0 der TSM-Client-API erforderlich ist.
- 64-Bit-Linux für S/390-Systeme, für die mindestens Version 5.2.2 der TSM-Client-API erforderlich ist.

Konfigurieren eines Tivoli Storage Manager-Clients

Bevor der Datenbankmanager die Option TSM verwenden kann, müssen Sie die folgenden Schritte zur Konfiguration der TSM-Umgebung ausführen:

1. Es muss ein funktionierender TSM-Client und -Server installiert und konfiguriert werden. Außerdem muss auf jedem DB2-Server die TSM-Client-API installiert sein.
2. 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 anderen 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 im Verzeichnis `sqllib/userprofile` angegeben werden.
3. Falls an diesen Umgebungsvariablen Änderungen vorgenommen werden, während der Datenbankmanager ausgeführt wird, sollten Sie folgende Schritte ausführen:
   - Stoppen Sie den Datenbankmanager mit dem Befehl **db2stop**.
   - Starten Sie den Datenbankmanager mit dem Befehl **db2start**.
4. Abhängig von der Konfiguration des Servers benötigt der Tivoli-Client für den Datenaustausch mit dem 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 Exemplareigners installiert. Mit Hilfe 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 die Befehle BACKUP DATABASE oder 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 wenn das Kennwort auf dem TSM-Server zurückgesetzt worden ist.

Aspekte der Verwendung von Tivoli Storage Manager

Bestimmte Funktionen innerhalb von TSM können Sie nur verwenden, indem Sie eventuell den vollständig qualifizierten Pfadnamen des Objekts angeben, das diese Funktion verwendet. (Beachten Sie, dass Sie unter Windows-Betriebssystemen das Zeichen \ statt / eingeben müssen.) Für vollständig qualifizierte Pfadnamen gilt Folgendes:

- Der Pfadname eines Objekts für eine Datenbankgesamtsicherung setzt sich wie folgt zusammen: /<datenbank>/NODEnnnn/FULL_BACKUP.zeitmarke.folgenr_no
- Der Pfadname eines Objekts für eine Datenbankteilsicherung setzt sich wie folgt zusammen: /<datenbank>/NODEnnnn/DB_INCR_BACKUP.zeitmarke.folgenr_no
- Der Pfadname eines Objekts für eine Datenbankdeltateilsicherung setzt sich wie folgt zusammen_
  /<datenbank>/NODEnnnn/DB_DELTA_BACKUP.zeitmarke.folgenr_no
- Der Pfadname eines Objekts für eine Tabellenbereichsgesamtsicherung setzt sich wie folgt zusammen: /<datenbank>/NODEnnnn/TSP_BACKUP.zeitmarke.folgenr_no
- Der Pfadname eines Objekts für eine Tabellenbereichsteilsicherung setzt sich wie folgt zusammen:
  /<datenbank>/NODEnnnn/TSP_INCR_BACKUP.zeitmarke.folgenr_no
- Der Pfadname eines Objekts für eine Tabellenbereichsdeltateilsicherung setzt sich wie folgt zusammen:
  /<datenbank>/NODEnnnn/TSP_DELTA_BACKUP.zeitmarke.folgenr_no

Hierbei steht <datenbank> für den Datenbankaliasnamen und NODEnnnn für die Knotennummer. Die Namen in Großbuchstaben müssen Sie genau wie dargestellt eingeben.

- Wenn Sie mehrere Sicherungsimages 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 Sicherungsversion zu ermitteln.

- Individuelle Sicherungsimages werden in Dateibereichen zusammengefasst, die von TSM verwaltet werden. Einzelne Sicherungsimages können Sie nur über die TSM-APIs oder mit Hilfe des Dienstprogramms **db2adutl**, das diese APIs verwendet, bearbeiten.
- Auf dem TSM-Server kommt es zu einer Zeitlimitüberschreitung einer Sitzung, wenn der Tivoli-Client nicht innerhalb des Zeitraums antwortet, der im Parameter `COMMTIMEOUT` der Konfigurationsdatei des Servers angegeben wurde. Die folgenden drei Faktoren können zum Problem mit der Zeitlimitüberschreitung beitragen:
  - Der Parameter `COMMTIMEOUT` wurde möglicherweise auf dem TSM-Server auf einen zu niedrigen Wert gesetzt. Wenn zum Beispiel bei einer Wiederherstellungsoperation große DMS-Tabellenbereiche erstellt werden, kann es zu einer Zeitlimitüberschreitung kommen. Der empfohlene Wert für diesen Parameter ist 6000 Sekunden.
  - Der DB2-Sicherungs- oder -Wiederherstellungspuffer ist möglicherweise zu groß.
  - Die Datenbankaktivität während einer Onlinesicherung ist möglicherweise zu hoch.
- Wenn Sie den Durchsatz erhöhen wollen, verwenden Sie Mehrfachsitzungen (nur bei ausreichender Hardware auf dem TSM-Server).

Zugehörige Konzepte:

- „Verwalten von Protokolldateien" auf Seite 51
- „Hierarchical Storage Manager (HSM) von Tivoli Space Manager" im Handbuch *IBM Data Links Manager Einstieg*

Zugehörige Referenzen:

- „db2adutl - DB2-Objekte in TSM verwalten" auf Seite 271

Anhang G. Benutzerexit zur Datenbankwiederherstellung

Sie können ein *Benutzerexitprogramm* entwickeln, um die Archivierung und den Abruf von Protokolldateien zu automatisieren. Bevor Sie ein Benutzerexitprogramm zur Archivierung oder zum Abruf von Protokolldateien aufrufen, muss der Datenbankkonfigurationsparameter *logarchmeth1* auf `USEREXIT` gesetzt sein. Dies ermöglicht außerdem die aktualisierende Wiederherstellung der Datenbank.

Wenn Sie ein Benutzerexitprogramm aufrufen, übergibt der Datenbankmanager die Steuerung an die ausführbare Datei `db2uext2`. Der Datenbankmanager übergibt Parameter an `db2uext2`, und das Programm gibt bei seiner Beendigung einen Rückkehrcode an den Datenbankmanager zurück. Da der Datenbankmanager nur eine begrenzte Anzahl von Rückkehrcodes bearbeiten kann, sollte das Benutzerexitprogramm Fehlerbedingungen verarbeiten können (siehe „Fehlerbehandlung" auf Seite 371). Und da Sie innerhalb eines Datenbankmanagerexemplars nur ein Benutzerexitprogramm aufrufen können, muss es für jede Operation, die es möglicherweise ausführen soll, einen Abschnitt aufweisen.

Die folgenden Themen werden behandelt:

Benutzerexitbeispielprogramme

Benutzerexitbeispielprogramme sind für alle unterstützten Plattformen verfügbar. Sie können diese Programme so modifizieren, dass sie an die spezifischen Anforderungen angepasst sind. Die Beispielprogramme sind gut kommentiert, und somit können Sie diese sehr effektiv 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 Datenbankwiederherstellung Fehler verursachen.) DB2® entfernt archivierte Protokolldateien aus dem Pfad der aktiven Protokolldateien, sobald diese Protokolldateien nicht mehr zur Wiederherstellung erforderlich sind.

Es folgt eine Beschreibung der Benutzerexitbeispielprogramme, die mit DB2 ausgeliefert werden.

- **UNIX®-Systeme**

  Die Benutzerexitbeispielprogramme für DB2 für UNIX-Systeme befinden sich im Unterverzeichnis `sqllib/samples/c`. Obwohl die Beispielprogramme in der Programmiersprache C codiert sind, können Sie das Benutzerexitprogramm auch in einer anderen Programmiersprache schreiben.

  Das Benutzerexitprogramm muss eine ausführbare Datei sein, deren Name `db2uext2` lautet.

  Folgende vier Benutzerexitbeispielprogramme für UNIX-Systeme sind verfügbar:

  – `db2uext2.ctsm`

    Dieses Beispielprogramm verwendet Tivoli® Storage Manager, um Datenbankprotokolldateien zu archivieren und abzurufen.

– db2uext2.ctape

  Dieses Beispielprogramm verwendet Bänder, 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 für Windows-Betriebssysteme befinden sich im Unterverzeichnis sqllib\samples\c. Obwohl die Beispielprogramme in der Programmiersprache C codiert sind, können Sie das 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 Festplatten, um Datenbankprotokolldateien zu archivieren und abzurufen.

Aufrufformat

Wenn der Datenbankmanager ein Benutzerexitprogramm aufruft, übergibt er einen Satz von Parametern (mit dem Datentyp CHAR) an das Programm. Das Aufrufformat hängt vom eingesetzten Betriebssystem ab:

```
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 das Exemplar 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.

knotennr Gibt die Nummer des lokalen Knotens an, z. B. 5.

protokollpfad 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. (Sollte der Wert für den Datenbankkonfigurationsparameter *tsm_password* bereits angegeben worden sein, 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

Das Benutzerexitprogramm sollte so konzipiert sein, dass spezifische und aussagekräftige Rückkehrcodes zurückgegeben werden, damit der Datenbankmanager diese richtig interpretieren kann. Da das Benutzerexitprogramm durch den Befehlsprozessor des Betriebssystems aufgerufen wird, gibt das Betriebssystem möglicherweise selbst Fehlercodes zurück. Da diese Fehlercodes nicht erneut zugeordnet werden, können Sie die Hilfe für Betriebssystemnachrichten verwenden, um Informationen zu diesen Fehlercodes abzurufen.

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 dieser so behandelt, als hätte er den Wert 32.

Tabelle 8. Rückkehrcodes von Benutzerexitprogrammen. Nur für Archivierungs- und Abrufoperationen gültig.

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, dass 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.). Nur für Archivierungs- und Abrufoperationen gültig.

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 Wiederherstellungen 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 worden sind, 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 Wiederherstellung 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 Wiederherstellung 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:** Während 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.	

Anhang H. APIs zum Sichern und Wiederherstellen für Produkte anderer Lieferanten

APIs für das Sichern und Wiederherstellen in Speichermanagern

DB2 bietet Schnittstellen für Produkte zur Datenträgerverwaltung von Fremdanbietern, die Sie zum Speichern und Abrufen von Daten für Sicherungs- und Wiederherstellungsoperationen und Protokolldateien verwenden können. Mit dieser Funktion werden den Fremdprodukten die Sicherungs-, Wiederherstellungs- und Protokollarchivierungsdatenziele Diskette, Platte, Band und Tivoli Storage Manager hinzugefügt, die standardmäßig von DB2 unterstützt werden. Diese Produkte zur Datenträgerverwaltung von Fremdanbietern werden in diesem Anhang als Produkte anderer Lieferanten bezeichnet.

DB2 definiert eine Gruppe von Funktionsprototypen, die eine für allgemeine Zwecke verwendbare Datenschnittstelle zur Sicherung, Wiederherstellung und Protokollarchivierung bietet, welche von vielen Produkten anderer Lieferanten verwendet werden kann. Diese Funktionen müssen vom Lieferanten für UNIX-Systeme in einer gemeinsam benutzten Bibliothek und für Windows-Betriebssysteme in einer DLL (Dynamic Link Library) bereitgestellt werden. Wenn DB2 die Funktionen aufruft, wird die gemeinsam benutzte Bibliothek bzw. die DLL geladen, die von der aufrufenden Sicherungs-, Wiederherstellungs- oder Protokollarchivierungsroutine angegeben wird. Die Funktionen des Produkts eines anderen Lieferanten werden zur Ausführung der erforderlichen Tasks aufgerufen.

Die Beispieldateien, die die DB2-Funktionalität für Produkte anderer Lieferanten demonstrieren, befinden sich auf UNIX-Plattformen im Verzeichnis `sqllib/samples/BARVendor` und unter Windows im Verzeichnis `sqllib\samples\BARVendor`.

Funktionsübersicht

Als Schnittstelle zwischen DB2 und Produkten anderer Lieferanten sind die folgenden sieben Funktionen definiert:

- sqluvint - Verbindung zu Einheit initialisieren und herstellen
- sqluvget - Daten von Einheit lesen
- sqluvput - Daten auf Einheit schreiben
- sqluvend - Verbindung zu Einheit aufheben
- sqluvdel - Festgeschriebene Sitzung löschen
- db2VendorQueryApiVersion - Von Einheit unterstützte API-Stufe abfragen
- db2VendorGetNextObj - Nächstes Objekt für Einheit abrufen

DB2 ruft diese Funktionen auf, die vom Produkt eines anderen Lieferanten für UNIX-Systeme in einer gemeinsam benutzten Bibliothek und für Windows-Betriebssysteme in einer DLL bereitgestellt werden müssen.

Anmerkung: Der Code der gemeinsam benutzten Bibliothek oder der DLL wird als Teil des Codes der Datenbanksteuerkomponente ausgeführt. Deshalb muss er simultan verwendbar und gründlich getestet sein. Eine fehlerhafte Funktion kann die Datenintegrität der Datenbank beeinträchtigen.

Die Reihenfolge der Funktionen, die DB2 während einer spezifischen Sicherungs- oder Wiederherstellungsoperation aufruft, hängt von Folgendem ab:

- Anzahl der Sitzungen, die belegt werden
- Typ der Operation (Sicherung, Wiederherstellung, Protokollarchivierung, Protokollabruf)
- PROMPTING-Modus, der für die Sicherungs- und Wiederherstellungsoperation angegeben ist
- Kenndaten der Einheit, auf der die Daten gespeichert sind
- Fehler, die während der Operationsausführung auftreten können

Anzahl Sitzungen

Die Sicherung und Wiederherstellung von Datenbankobjekten wird von DB2 mit mindestens einem Datenstrom oder einer Sitzung unterstützt. Eine Sicherung oder Wiederherstellung in drei Sitzungen würde drei verfügbare physische oder logische Einheiten erfordern. Wenn die Unterstützung von Einheiten anderer Lieferanten verwendet wird, ist es die Aufgabe der Lieferantenfunktionen, die Schnittstelle zu den einzelnen physischen oder logischen Einheiten zu verwalten. DB2 sendet oder empfängt einfach Datenpuffer an die/von den Lieferantenfunktionen.

Die Anzahl der zu verwendenden Sitzungen wird von der Anwendung, die die Datenbankfunktion zu Sicherung oder Wiederherstellung aufruft, als Parameter angegeben. Dieser Wert wird in der Struktur INIT-INPUT bereitgestellt, die von `sqluvint` verwendet wird.

DB2 fährt damit fort, Sitzungen zu initialisieren, bis die angegebene Zahl erreicht ist oder bis DB2 den Warnungsrückkehrcode SQLUV_MAX_LINK_GRANT von einem `sqluvint`-Aufruf empfängt. Zur Warnung an DB2, dass das Maximum der von DB2 unterstützten Sitzungen erreicht ist, ist im Produkt eines anderen Lieferanten Code erforderlich, der die Anzahl aktiver Sitzungen verfolgt. Falls DB2 nicht gewarnt wird, fordert DB2 möglicherweise die Initialisierung der Sitzung an. Diese Anforderung schlägt fehl und führt zur Beendigung aller Sitzungen und zum Fehlschlagen der gesamten Sicherungs- oder Wiederherstellungsoperation.

Wenn die Operation eine Sicherung ist, schreibt DB2 zu Beginn jeder Sitzung einen Datenträgerheadersatz. Dieser Satz enthält Daten, anhand derer DB2 die Sitzung während einer Wiederherstellungsoperation bestimmt. DB2 gibt die einzelnen Sitzungen an, indem an den Namen des Sicherungsimages eine Folgenummer angehängt wird. Die Nummer beginnt mit eins für die erste Sitzung und wird jedes Mal um eins erhöht, wenn eine Sitzung mit einem `sqluvint`-Aufruf für eine Sicherungs- oder Wiederherstellungsoperation eingeleitet wird.

Wenn die Sicherungsoperation erfolgreich beendet ist, schreibt DB2 einen Datenträgertrailer in die letzte Sitzung, die geschlossen wird. Dieser Trailer enthält Daten, an denen DB2 erkennen kann, wie viele Sitzungen für die Sicherungsoperation verwendet wurden. Während einer Wiederherstellungsoperation wird mit diesen Daten sichergestellt, dass alle Sitzungen bzw. Datenströme wiederhergestellt wurden.

Betrieb ohne Fehler, Warnungen oder Bedienerführung

Für Sicherungen setzt DB2 für *jede einzelne* Sitzung die folgende Aufruffolge ab.

```
sqluvint, action = SQLUV_WRITE
```

Anschließend: 1 bis n

```
sqluvput
```

Anschließend: 1

```
sqluvend, action = SQLUV_COMMIT
```

Wenn DB2 einen sqluvend-Aufruf (Aktion SQLUV_COMMIT) absetzt, wird vom Produkt eines anderen Lieferanten erwartet, die Ausgabedaten entsprechend zu speichern. Der Rückkehrcode SQLUV_OK an DB2 zeigt die erfolgreiche Beendigung an.

Die Struktur DB2-INFO, die für den Aufruf von sqluvint verwendet wird, enthält Daten, die zur Kennzeichnung der Sicherung erforderlich sind. Es wird eine Folgenummer angegeben. Möglicherweise speichert das Produkt eines anderen Lieferanten diese Daten. DB2 verwendet die Nummer während der Wiederherstellung, um die Sicherung zu erkennen, die wiederhergestellt wird.

Für eine Wiederherstellung lautet die Aufruffolge wie folgt:

```
sqluvint, action = SQLUV_READ
```

Anschließend: 1 bis n

```
sqluvget
```

Anschließend: 1

```
sqluvend, action = SQLUV_COMMIT
```

In der Struktur DB2-INFO, die für den Aufruf von sqluvint verwendet wird, sind die Daten enthalten, die zur Kennzeichnung der Sicherung erforderlich sind. Es wird keine Folgenummer angegeben. DB2 erwartet, dass alle Sicherungsobjekte (Sitzungsausgaben, die während einer Sicherung festgeschrieben wurden) zurückgegeben werden. Das erste zurückgegebene Sicherungsobjekt ist das Objekt, für das die Folgenummer 1 generiert wurde, und alle übrigen Objekte werden in keiner spezifischen Reihenfolge wiederhergestellt. DB2 überprüft den Datenträgernachsatz, um sicherzustellen, dass alle Objekte verarbeitet wurden.

Anmerkung: Nicht alle Produkte anderer Lieferanten zeichnen die Namen der Sicherungsobjekte auf. Dies ist am wahrscheinlichsten, wenn auf Band oder auf andere Datenträger mit begrenzter Kapazität gesichert wird. Während der Initialisierung von Wiederherstellungssitzungen können mit den Kennzeichnungsdaten die nötigen Sicherungsobjekte zwischengespeichert werden, so dass diese bei Bedarf verfügbar sind. Dies ist am zweckmäßigsten, wenn Sie zur Speicherung der Sicherungskopien automatische Wechselsysteme (Jukeboxes) oder automatische Positionierungssysteme einsetzen. DB2 prüft immer den Datenträgerheader (den ersten Datensatz in der Ausgabe der einzelnen Sitzungen), damit immer die richtigen Daten wiederhergestellt werden.

PROMPTING-Modus

Wenn Sie eine Sicherungs- oder Wiederherstellungsoperation einleiten, sind die folgenden zwei PROMPTING-Modi möglich:

- WITHOUT PROMPTING oder NOINTERRUPT: Das Produkt eines anderen Lieferanten kann weder Nachrichten an den Benutzer ausgeben noch kann der Benutzer auf Eingabeaufforderungen antworten.
- PROMPTING oder INTERRUPT: Der Benutzer kann Nachrichten vom Produkt eines anderen Lieferanten empfangen und auf diese Nachrichten antworten.

Für den PROMPTING-Modus werden von der Sicherung/Wiederherstellung drei mögliche Benutzerantworten definiert:

- Fortsetzen (c - continue)

  Die Lese- oder Schreiboperation von der bzw. auf die Einheit wird fortgesetzt.
- Einheit beenden (d - device terminate)

  Die Einheit empfängt keine weiteren Daten, und die Sitzung wird beendet.
- Beenden (t - terminate)

  Die gesamte Sicherungs- oder Wiederherstellungsoperation wird beendet.

Die Verwendung der Modi PROMPTING und WITHOUT PROMPTING wird in den folgenden Abschnitten behandelt.

Kenndaten von Einheiten

Für die APIs zur Unterstützung von Einheiten anderer Lieferanten sind zwei allgemeine Einheitentypen definiert:

- Einheiten mit begrenzter Kapazität, bei denen der Benutzer den Datenträger wechseln muss, z. B. Bandlaufwerke, Disketten oder CD-ROMs
- Einheiten mit sehr großer Kapazität, bei denen der Benutzer im normalen Betrieb keine Datenträger wechseln muss, z. B. ein automatisches Wechselsystem (Jukebox) oder ein intelligentes automatisches Positionierungssystem zum Wechseln von Datenträgern

Bei einer Einheit mit begrenzter Kapazität wird der Benutzer während der Sicherungs- oder Wiederherstellungsoperation möglicherweise aufgefordert, weitere Datenträger einzulegen. Allgemein ist die Reihenfolge, in der die Datenträger für Sicherungs- oder Wiederherstellungsoperationen eingelegt werden, für DB2 nicht von Bedeutung. Außerdem stellt DB2 Einrichtungen bereit, mit denen Nachrichten von Produkten anderer Lieferanten über die Handhabung von Datenträgern an den Benutzer übertragen werden. Dazu muss die Sicherungs- oder Wiederherstellungsoperation mit aktivierter PROMPTING-Funktion eingeleitet werden. Den Nachrichtentext für die Datenträgerhandhabung geben Sie im Beschreibungsfeld der Rückkehrcodestruktur an.

Wenn PROMPTING aktiv ist und DB2 den Rückkehrcode SQLUV_ENDOFMEDIA oder SQLUV_ENDOFMEDIA_NO_DATA von einem `sqluvput`-Aufruf (Schreiben) oder einem `sqluvget`-Aufruf (Lesen) empfängt, führt DB2 Folgendes aus:

- Bei einem `sqluvput`-Aufruf markiert DB2 den letzten an die Sitzung übertragenen Puffer so, dass er erneut übertragen wird. Er wird später an die Sitzung übertragen.
- DB2 ruft die Sitzung mit `sqluvend` (action = SQLUV_COMMIT) auf. Bei Erfolg (Rückkehrcode SQLUV_OK) führt DB2 Folgendes aus:
  - DB2 sendet an den Benutzer aus der Rückkehrcodestruktur, die das Datenträgerende signalisierte, eine Nachricht vom Produkt eines anderen Lieferanten über die Handhabung von Datenträgern.
  - DB2 fordert den Benutzer auf, anzugeben, ob die Aktion fortgesetzt, die Einheit beendet oder alles beendet wird.
- Will der Benutzer die Aktion *fortsetzen*, initialisiert DB2 eine weitere Sitzung und verwendet dabei den Aufruf `sqluvint`. Wenn dieser erfolgreich beendet ist, beginnt DB2, Daten in die Sitzung zu schreiben oder aus der Sitzung zu lesen. Zur eindeutigen Kennzeichnung der Sitzung beim Schreiben erhöht DB2 die Folgenummer. Die Folgenummer befindet sich in der Struktur DB2-INFO des Aufrufs `sqluvint` im Datenträgerheadersatz. Dies ist der erste Datensatz, der an die Sitzung gesendet wird.

DB2 startet nicht mehr Sitzungen als bei Beginn einer Sicherungs- oder Wiederherstellungsoperation angefordert oder als vom Produkt eines anderen Lieferanten mit der Warnung SQLUV_MAX_LINK_GRANT für den Aufruf `sqluvint` angegeben wurden.

- Will der Benutzer die *Einheit beenden*, versucht DB2 nicht, eine weitere Sitzung zu initialisieren, und die Anzahl aktiver Sitzungen vermindert sich um eins. DB2 lässt nicht zu, dass alle Sitzungen von Antworten zur Beendigung einer Einheit beendet werden. Mindestens eine Sitzung muss aktiv bleiben, bis die Sicherungs- oder Wiederherstellungsoperation beendet ist.
- Will der Benutzer die Aktion *beenden*, beendet DB2 die Sicherungs- oder Wiederherstellungsoperation. Weitere Informationen zum genauen Ablauf, wie DB2 die Sitzungen beendet, finden Sie in „Bei Rückgabe von Fehlermeldungen an DB2" auf Seite 378.

Da die Leistung bei der Sicherung oder Wiederherstellung häufig von der Anzahl der verwendeten Einheiten abhängt, ist es wichtig, die Parallelität beizubehalten. Für Sicherungsoperationen sollten Benutzer angeben, dass sie die Aktion fortsetzen wollen, es sei denn, ihnen ist bekannt, dass die restlichen aktiven Sitzungen die noch zu schreibenden Daten im Speicher halten. Für Wiederherstellungsoperationen sollten die Benutzer ebenfalls angeben, dass sie die Aktion fortsetzen wollen, bis alle Datenträger verarbeitet worden sind.

Wenn der Sicherungs- oder Wiederherstellungsmodus WITHOUT PROMPTING lautet und DB2 von einer Sitzung den Rückkehrcode SQLUV_ENDOFMEDIA oder SQLUV_ENDOFMEDIA_NO_DATA empfängt, beendet DB2 die Sitzung und versucht nicht, eine weitere Sitzung zu öffnen. Wenn alle Sitzungen an DB2 das Datenträgerende zurückgeben, bevor die Sicherungs- oder Wiederherstellungsoperation beendet ist, schlägt die Operation fehl. Daher sollten Sie den Modus WITHOUT PROMPTING bei Einheiten mit begrenzter Kapazität mit Bedacht anwenden. Dennoch kann es sinnvoll sein, Einheiten mit sehr großer Kapazität in diesem Modus zu betreiben.

Das Produkt eines anderen Lieferanten kann Aktionen zum Anhängen und Wechseln der Datenträger vor DB2 verbergen, so dass die Einheit scheinbar eine unendliche Kapazität aufweist. Einige Einheiten mit sehr großer Kapazität arbeiten in diesem Modus. In diesen Fällen ist es kritisch, alle gesicherten Daten in der gleichen Reihenfolge an DB2 zurückzugeben, während die Verarbeitung einer Wiederherstellungsoperation läuft. Andernfalls können Daten verloren gehen. DB2 geht jedoch von einer erfolgreich beendeten Wiederherstellungsoperation aus, da DB2 keine Möglichkeit hat, verlorene Daten zu erkennen.

DB2 schreibt Daten auf das Produkt eines anderen Lieferanten, wobei angenommen wird, dass jeder Puffer auf nur einem einzigen Datenträger enthalten sein wird (z. B. auf einem Band). Das Produkt eines anderen Lieferanten kann diese Puffer auf mehrere Datenträger aufteilen, ohne dass DB2 dies erkennt. In diesem Fall ist die Reihenfolge kritisch, in der die Datenträger während der Wiederherstellungsoperation verarbeitet werden, da das Produkt eines anderen Lieferanten die Aufgabe hat, von mehreren Datenträgern wiederhergestellte Puffer an DB2 zurückzugeben. Andernfalls schlägt die Wiederherstellungsoperation fehl.

Bei Rückgabe von Fehlermeldungen an DB2

Bei einer Sicherungs- oder Wiederherstellungsoperation erwartet DB2, dass alle Sitzungen erfolgreich beendet werden. Andernfalls schlägt die gesamte Sicherungs- oder Wiederherstellungsoperation fehl. Die erfolgreiche Beendigung signalisiert eine Sitzung an DB2 mit dem Rückkehrcode SQLUV_OK des `sqluvend`-Aufrufs (action = SQLUV_COMMIT).

Sollten nicht behebbare Fehler eintreten, beendet DB2 die Sitzung. Dies können DB2-Fehler oder vom Produkt eines anderen Lieferanten an DB2 zurückgegebene Fehler sein. Da alle Sitzungen erfolgreich festgeschrieben werden müssen, damit eine Sicherungs- oder Wiederherstellungsoperation beendet werden kann, beendet DB2, wenn eine Sitzung fehlschlägt, auch die übrigen Sitzungen, die zur Operation gehören.

Wenn das Produkt eines anderen Lieferanten auf einen DB2-Aufruf mit einem Rückkehrcode für einen nicht behebbaren Fehler antwortet, kann das Produkt eines anderen Lieferanten optional zusätzliche Informationen bereitstellen. Dazu kann es den Nachrichtentext im Beschreibungsfeld der Struktur RETURN-CODE verwenden. Diesen Nachrichtentext erhält der Benutzer zusammen mit DB2-Informationen, so dass er den Fehler berichtigen kann.

Es gibt Sicherungsszenarios, in denen eine Sitzung erfolgreich festgeschrieben wurde, während in einer anderen Sitzung der Sicherungsoperation ein nicht behebbarer Fehler eintritt. Da alle Sitzungen erfolgreich beendet werden müssen, bevor eine Sicherungsoperation als erfolgreich gilt, muss DB2 die Ausgabedaten in den festgeschriebenen Sitzungen löschen: DB2 setzt einen `sqluvdel`-Aufruf ab, um das Löschen des Objekts anzufordern. Dieser Aufruf gilt nicht als E/A-Sitzung, und mit ihm werden alle Verbindungen initialisiert und beendet, die möglicherweise zum Löschen des Sicherungsobjekts erforderlich sind.

Die Struktur DB2-INFO enthält keine Folgenummer. `sqluvdel` löscht alle Sicherungsobjekte, die den übrigen Parametern in der Struktur DB2-INFO entsprechen.

Warnungsbedingungen

DB2 erhält vom Produkt eines anderen Lieferanten möglicherweise Warnungsrückkehrcodes, z. B., wenn eine Einheit nicht bereit ist oder wenn andere nicht behebbare Fehler auftraten. Dies gilt sowohl für Schreib- als auch für Leseoperationen.

Bei `sqluvput`- und `sqluvget`-Aufrufen kann das Produkt eines anderen Lieferanten den Rückkehrcode SQLUV_WARNING senden und optional zusätzliche Informationen bereitstellen. Dazu kann es den Nachrichtentext im Beschreibungsfeld der Struktur RETURN-CODE verwenden. Diesen Nachrichtentext erhält der Benutzer, so dass er den Fehler berichtigen kann. Der Benutzer kann die die Aktion fortsetzen, die Einheit beenden oder die Aktion beenden:

- Will der Benutzer die Aktion *fortsetzen*, versucht DB2, den Puffer mit `sqluvput` während der Sicherungsoperation erneut zu schreiben. Bei einer Wiederherstellungsoperation, setzt DB2 einen `sqluvget`-Aufruf ab, um den nächsten Puffer zu lesen.
- Will der Benutzer die *Einheit beenden* oder die Aktion *beenden*, beendet DB2 die gesamte Sicherungs- oder Wiederherstellungsoperation genauso wie nach einem nicht behebbaren Fehler. (Es werden z. B. auch aktive Sitzungen beendet und festgeschriebene Sitzungen gelöscht.)

Hinweise und Tipps für den Betrieb

In diesem Abschnitt finden Sie einige Hinweise und Tipps zur Erstellung von Produkten anderer Lieferanten.

Protokolldatei

Die Protokolldatei können Sie bei Datenbankwiederherstellungen zur Hilfe nehmen. Diese Datei ist jeder Datenbank zugeordnet und wird bei jeder Sicherungs- oder Wiederherstellungsoperation automatisch aktualisiert. Die Daten in dieser Datei können Sie mit den folgenden Einrichtungen anzeigen, aktualisieren oder löschen:

- Steuerzentrale
- Befehlszeilenprozessor (CLP)
  - Befehl LIST HISTORY
  - Befehl UPDATE HISTORY FILE
  - Befehl PRUNE HISTORY
- APIs
  - db2HistoryOpenScan
  - db2HistoryGetEntry
  - db2HistoryCloseScan
  - db2HistoryUpdate
  - db2Prune

Weitere Informationen zum Aufbau der Datei finden Sie im Abschnitt zu db2HistData.

Wenn eine Sicherungsoperation beendet wird, wird mindestens ein Datensatz in die Datei geschrieben. Wurde die Ausgabe der Sicherungsoperation unter Verwendung des Schlüsselworts LOAD auf Einheiten anderer Lieferanten umgeleitet, enthält das Feld DEVICE im Protokollsatz den Wert `0`. Wurde die Ausgabe an TSM geleitet, enthält das Feld DEVICE den Wert `A`. Das Feld LOCATION kann einen der beiden folgenden Werte enthalten:

- Den Namen der Lieferantendatei, der beim Aufruf der Sicherung angegeben wurde
- Den Namen der gemeinsam benutzten Bibliothek, sofern kein Name einer Lieferantendatei angegeben wurde

Weitere Informationen zum Angeben dieser Option finden Sie in „Aufrufen einer Sicherung oder Wiederherstellung mit Produkten anderer Lieferanten" auf Seite 380.

Das Feld LOCATION können Sie über die Steuerzentrale, mit dem CLP oder mit einer API aktualisieren. Die Position der Sicherungsinformationen können Sie aktualisieren, wenn Einheiten mit begrenzter Kapazität (z. B. austauschbare Datenträger) zur Speicherung des Sicherungsimages verwendet wurden und der Datenträger physisch an eine andere Speicherposition (möglicherweise an einen anderen Standort) versetzt wurde. Wenn dies zutrifft, können Sie anhand der Protokolldatei ein Sicherungsimage suchen, sobald eine Wiederherstellungsoperation erforderlich wird.

Aufrufen einer Sicherung oder Wiederherstellung mit Produkten anderer Lieferanten

Sie können Produkte anderer Lieferanten angeben, wenn Sie das DB2-Sicherungsdienstprogramm oder das DB2-Wiederherstellungsdienstprogramm mit Folgendem aufrufen:

- Steuerzentrale
- Befehlszeilenprozessor (CLP)
- Anwendungsprogrammierschnittstelle (API)

Steuerzentrale

Die Steuerzentrale ist die grafische Benutzerschnittstelle für die Datenbankverwaltung. Sie wird mit DB2 ausgeliefert.

Angabe von	**Eingabevariable der Steuerzentrale für Sicherungs- und Wiederherstellungsoperationen**
Verwendung der Einheit und des Bibliotheksnamens des anderen Lieferanten	*Bibliothek verwenden*. Geben Sie den Bibliotheksnamen (für UNIX-Systeme) oder den Namen der DLL (Windows-Betriebssystem) an.
Anzahl der Sitzungen	*Sitzungen*
Lieferantenoptionen	Nicht unterstützt
Lieferantendateiname	Nicht unterstützt
Größe des Übertragungspuffers	*Puffergröße* (für Sicherung) bzw. nicht zutreffend (für Wiederherstellung).

Der Befehlszeilenprozessor (CLP)

Mit dem Befehlszeilenprozessor (CLP - Command Line Processor) können Sie die DB2-Befehle BACKUP DATABASE oder RESTORE DATABASE aufrufen.

Angabe von	**Parameter des Befehlszeilenprozessors**	
	Für die Sicherung	**Für die Wiederherstellung**
Verwendung der Einheit und des Bibliotheksnamens des anderen Lieferanten	*bibliotheksname*	*gemeinsame-bibliothek*
Anzahl der Sitzungen	*anzahl-sitzungen*	*anzahl-sitzungen*
Lieferantenoptionen	Nicht unterstützt	Nicht unterstützt
Lieferantendateiname	Nicht unterstützt	Nicht unterstützt
Größe des Übertragungspuffers	*puffergröße*	*puffergröße*

API-Funktionsaufrufe für Sicherung und Wiederherstellung

Zwei API-Funktionsaufrufe unterstützen Sicherungs- und Wiederherstellungsoperationen: `db2Backup` für Sicherung und `db2Restore` für Wiederherstellung.

Angabe von	API-Parameter (für db2Backup und db2Restore)
Verwendung der Einheit und des Bibliotheksnamens des anderen Lieferanten	In der Struktur *sqlu_media_list* geben Sie den Datenträgertyp SQLU_OTHER_MEDIA an, und in der Struktur *sqlu_vendor* geben Sie eine gemeinsam benutzte Bibliothek oder eine DLL in *shr_lib* an.
Anzahl der Sitzungen	In der Struktur *sqlu_media_list* geben Sie *sessions* an.
Lieferantenoptionen	*PVendorOptions*
Lieferantendateiname	In der Struktur *sqlu_media_list* geben Sie den Datenträgertyp SQLU_OTHER_MEDIA an, und in der Struktur *sqlu_vendor* geben Sie einen Dateinamen in *filename* an.
Größe des Übertragungspuffers	*BufferSize*

Zugehörige Referenzen:

- „sqluvint - Initialisieren und Verbindung zu Einheit herstellen" auf Seite 381
- „sqluvget - Daten von Einheit lesen" auf Seite 385
- „sqluvput - Daten auf Einheit schreiben" auf Seite 386
- „sqluvend - Verbindung zu Einheit aufheben und Ressourcen freigeben" auf Seite 389
- „sqluvdel - Festgeschriebene Sitzung löschen" auf Seite 391
- „DB2-INFO" auf Seite 395
- „VENDOR-INFO" auf Seite 398
- „INIT-INPUT" auf Seite 399
- „INIT-OUTPUT" auf Seite 400
- „DATA" auf Seite 400
- „RETURN-CODE" auf Seite 401
- „db2VendorQueryApiVersion - Von Einheit unterstützte API-Stufe abfragen" auf Seite 392
- „db2VendorGetNextObj - Nächstes Objekt für Einheit abrufen" auf Seite 393

sqluvint - Initialisieren und Verbindung zu Einheit herstellen

Diese Funktion können Sie aufrufen, um Daten zur Initialisierung und zur Herstellung einer logischen Verbindung zwischen DB2 und der Einheit eines anderen Lieferanten anzugeben.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *dbadm*

Erforderliche Verbindung:

Datenbank

API-Includedatei:

sql.h

Syntax der C-API:

```
/* File: sqluvend.h */
/* API: Initialize and Link to Device */
/* ... */
int sqluvint (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API-Parameter:

Init_input
Eingabe. Struktur, die von DB2 gelieferte Daten enthält, mit denen eine logische Verbindung zur Einheit des anderen Lieferanten hergestellt wird.

Init_output
Ausgabe. Struktur, die die von der Einheit des anderen Lieferanten zurückgegebene Ausgabe enthält.

Return_code
Ausgabe. Struktur mit dem Rückkehrcode, der an DB2 übergeben werden soll, und einem kurzen Erläuterungstext.

Hinweise:

Für die einzelnen E/A-Sitzungen der Datenträger ruft DB2 diese Funktion auf, damit es eine Einheitenkennung erhält. Wenn die Lieferantenfunktion aus irgendeinem Grund bei der Initialisierung einen Fehler findet, gibt sie einen entsprechenden Rückkehrcode aus. Wenn der Rückkehrcode einen Fehler angibt, beendet DB2 möglicherweise die Operation mit dem Aufruf der Funktion **sqluvend**. Einzelangaben zu den möglichen Rückkehrcodes und zum jeweiligen Verhalten von DB2 finden Sie in der Tabelle mit Rückkehrcodes (siehe Tabelle 9 auf Seite 383).

Die Struktur INIT-INPUT enthält Elemente, die vom Produkt eines anderen Lieferanten verwendet werden können, um zu bestimmen, ob die Sicherung oder die Wiederherstellung fortgesetzt werden kann:

- size_HI_order und size_LOW_order

  Dies ist die geschätzte Größe der Sicherung. Damit können Sie bestimmen, ob die Einheiten des anderen Lieferanten die Größe des Sicherungsimages bearbeiten können. Mit diesen Angaben kann die Menge austauschbarer Datenträger geschätzt werden, die zum Speichern der Sicherung erforderlich sind. Es kann nützlich sein, wenn der erste Aufruf von **sqluvint** fehlschlägt, sofern Sie auf Fehler vorbereitet sind.
- req_sessions

  Die Anzahl der von Benutzern angeforderten Sitzungen können Sie in Verbindung mit der geschätzten Größe und der Ebene der Bedienerführung verwenden, um zu bestimmen, ob die Sicherungs- oder Wiederherstellungsoperation möglich ist.
- prompt_lvl

  Die Ebene der Bedienerführung zeigt dem Produkt eines anderen Lieferanten an, ob zu Aktionen aufgefordert werden kann, z. B. zum Austauschen von Datenträgern (z. B. dem Einlegen eines anderen Bandes in das Bandlaufwerk). Dies kann

darauf hindeuten, dass die Operation nicht fortgesetzt werden kann, da keine Möglichkeit der Bedienerführung besteht.

Wenn die Ebene der Bedienerführung WITHOUT PROMPTING lautet und die Menge der austauschbaren Datenträger größer ist als die Anzahl der angeforderten Sitzungen, kann DB2 die Operation nicht erfolgreich beenden.

DB2 gibt die zu beschreibende Sicherung oder die zu lesende Wiederherstellung über Felder in der Struktur DB2-INFO an. Wurde action = SQLUV_READ angegeben, muss das Produkt eines anderen Lieferanten auf das Vorhandensein des angegebenen Objekts prüfen. Wenn es nicht gefunden wird, muss der Rückkehrcode auf SQLUV_OBJ_NOT_FOUND gesetzt werden, so dass DB2 die entsprechende Aktion ausführt.

Nach der erfolgreichen Initialisierung fährt DB2 fort und setzt weitere Datenübertragungsfunktionen ab, kann jedoch die Sitzung jederzeit mit einem Aufruf von **sqluvend** beenden.

Rückkehrcodes:

Tabelle 9. Gültige Rückkehrcodes für sqluvint und resultierende DB2-Aktion

Literal in Headerdatei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
SQLUV_OK	Operation erfolgreich	sqluvput, sqluvget (siehe Kommentare)	Wenn action = SQLUV_WRITE ist, ist der nächste Aufruf sqluvput (zur Datensicherung). Wenn action = SQLUV_READ ist, muss das Vorhandensein des angegebenen Objekts geprüft werden, bevor SQLUV_OK zurückgegeben wird; der nächste Aufruf ist sqluvget zum Wiederherstellen von Daten.
SQLUV_LINK_EXIST	Sitzung vorher aktiviert	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_COMM_ ERROR	Fehler bei der Kommunikation mit einer Einheit	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_INV_VERSION	Die DB2-Produkte und Produkte anderer Lieferanten sind inkompatibel.	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_INV_ACTION	Ungültige Aktion wurde angefordert. Damit kann auch angezeigt werden, dass die Kombination der Parameter eine unausführbare Operation ergibt.	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_NO_DEV_ AVAIL	Momentan ist keine Einheit verfügbar.	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_OBJ_NOT_ FOUND	Das angegebene Objekt kann nicht gefunden werden. Dies können Sie verwenden, wenn die Aktion für den sqluvint-Aufruf 'R' (Read) lautet und das angeforderte Objekt gemäß den Bedingungen, die in der Struktur DB2-INFO angeben sind, nicht gefunden werden kann.	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.

Tabelle 9. Gültige Rückkehrcodes für sqluvint und resultierende DB2-Aktion (Forts.)

Literal in Headerdatei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
SQLUV_OBJS_FOUND	Mindestens ein Objekt erfüllt die angegebenen Bedingungen. Dies ergibt sich, wenn die Aktion für den sqluvint-Aufruf 'R' (Read) lautet und mindestens ein Objekt die Bedingungen in der Struktur DB2-INFO erfüllt.	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_INV_USERID	Ungültige Benutzer-ID	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_INV_ PASSWORD	Ungültiges Kennwort	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_INV_OPTIONS	Im Feld für die Lieferantenoptionen wurde eine ungültige Option festgestellt.	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_INIT_FAILED	Die Initialisierung ist fehlgeschlagen, und die Sitzung muss beendet werden.	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_DEV_ERROR	Einheitenfehler	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_MAX_LINK_ GRANT	Maximale Anzahl von Verbindungen hergestellt	sqluvput, sqluvget (siehe Kommentare)	Dies wird von DB2 als Warnung behandelt. Die Warnung teilt DB2 mit, keine zusätzlichen Sitzungen mit dem Produkt eines anderen Lieferanten zu öffnen, da das Maximum unterstützter Sitzungen erreicht wurde. (Anmerkung: Dies könnte an der Verfügbarkeit der Einheit liegen.) Wenn action = SQLUV_WRITE (BACKUP) ist, ist der nächste Aufruf sqluvput. Wenn action = SQLUV_READ ist, muss das Vorhandensein des angegebenen Objekts geprüft werden, bevor SQLUV_MAX_LINK_GRANT; zurückgegeben wird; der nächste Aufruf ist sqluvget zum Wiederherstellen von Daten.
SQLUV_IO_ERROR	E/A-Fehler	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.
SQLUV_NOT_ ENOUGH_SPACE	Es ist nicht genügend Speicherplatz für das gesamte Sicherungsimage vorhanden; die geschätzte Größe wird als 64-Bit-Wert in Byte angegeben.	Keine weiteren Aufrufe	Die Initialisierung der Sitzung schlägt fehl. Geben Sie Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation. Ein sqluvend-Aufruf wird nicht empfangen, da die Sitzung nie eingerichtet worden ist.

sqluvget - Daten von Einheit lesen

Nach der Initialisierung können Sie diese Funktion aufrufen, um Daten von der Einheit zu lesen.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *dbadm*

Erforderliche Verbindung:

Datenbank

API-Includedatei:

sqluvend.h

Syntax der C-API:

```
/* File: sqluvend.h */
/* API: Reading Data from Device */
/* ... */
int sqluvget (
  void * pVendorCB,
  struct Data         *,
  struct Return_code  *);
/* ... */

typedef struct Data
}
   sqlint32  obj_num;
   sqlint32  buff_size;
   sqlint32  actual_buff_size;
   void      *dataptr;
   void      *reserve;
{ Data;
```

API-Parameter:

pVendorCB
Eingabe. Zeiger auf den Speicherbereich, der der Struktur DATA (einschließlich Datenpuffer) und dem Rückkehrcode (Return_code) zugeordnet wurde.

Data Ein-/Ausgabe. Ein Zeiger auf die Struktur *data*.

Return_code
Ausgabe. Der Rückkehrcode aus dem API-Aufruf.

obj_num
Gibt an, welches Sicherungsobjekt abgerufen werden muss.

buff_size
Gibt die zu verwendende Puffergröße an.

actual_buff_size
Gibt die tatsächlichen Byte an, die gelesen oder geschrieben wurden. Dieser Wert muss so gesetzt werden, dass er anzeigt, wie viel Byte tatsächlich gelesen wurden.

dataptr
Ein Zeiger auf den Datenpuffer.

reserve
Zur zukünftigen Verwendung reserviert.

Hinweise:

Diese Funktion wird vom Wiederherstellungsdienstprogramm verwendet.

Rückkehrcodes:

Tabelle 10. Gültige Rückkehrcodes für sqluvget und resultierende DB2-Aktion

Literal in Headerdatei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
SQLUV_OK	Operation erfolgreich	sqluvget	DB2 verarbeitet die Daten.
SQLUV_COMM_ERROR	Fehler bei der Kommunikation mit einer Einheit	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_INV_ACTION	Ungültige Aktion wurde angefordert.	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_INV_DEV_HANDLE	Ungültige Einheitenkennung	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_INV_BUFF_SIZE	Ungültige Puffergröße	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_DEV_ERROR	Einheitenfehler	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_WARNING	Achtung. Dies sollten Sie nicht verwenden, um DB2 das Datenträgerende anzugeben; verwenden Sie dazu SQLUV_ENDOFMEDIA oder SQLUV_ENDOFMEDIA_NO_DATA. Allerdings können Bedingungen *Nicht bereit* von Einheiten mit diesem Rückkehrcode angezeigt werden.	sqluvget oder sqluvend, action = SQLU_ABORT	
SQLUV_LINK_NOT_EXIST	Momentan ist keine Verbindung vorhanden.	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_MORE_DATA	Operation erfolgreich beendet; weitere Daten verfügbar	sqluvget	
SQLUV_ENDOFMEDIA_NO_DATA	Datenträgerende und 0 Byte gelesen (z. B. Ende des Bandes)	sqluvend	
SQLUV_ENDOFMEDIA	Datenträgerende und > 0 Byte gelesen (z. B. Ende des Bandes)	sqluvend	DB2 verarbeitet die Daten und bearbeitet anschließend die Bedingung für das Datenträgerende.
SQLUV_IO_ERROR	E/A-Fehler	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
Nächster Aufruf:			
[a] Wenn der nächste Aufruf sqluvend, action = SQLU_ABORT ist, werden diese Sitzung und alle übrigen aktiven Sitzungen beendet.			

sqluvput - Daten auf Einheit schreiben

Nach der Initialisierung können Sie diese Funktion aufrufen, um Daten auf die Einheit zu schreiben.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *dbadm*

Erforderliche Verbindung:

Datenbank

API-Includedatei:

sqluvend.h

Syntax der C-API:

```
/* File: sqluvend.h */
/* API: Writing Data to Device */
/* ... */
int sqluvput (
  void * pVendorCB,
  struct Data  *,
  struct Return_code  *);
/* ... */

typedef struct Data
}
   sqlint32  obj_num;
   sqlint32  buff_size;
   sqlint32  actual_buff_size;
   void      *dataptr;
   void      *reserve;
{ Data;
```

API-Parameter:

pVendorCB
Eingabe. Zeiger auf den Speicherbereich, der der Struktur DATA (einschließlich Datenpuffer) und dem Rückkehrcode (Return_code) zugeordnet wurde.

Data Ausgabe. Datenpuffer, der mit zu schreibenden Daten gefüllt ist.

Return_code
Ausgabe. Der Rückkehrcode aus dem API-Aufruf.

obj_num
Gibt an, welches Sicherungsobjekt abgerufen werden muss.

buff_size
Gibt die zu verwendende Puffergröße an.

actual_buff_size
Gibt die tatsächlichen Byte an, die gelesen oder geschrieben wurden. Dieser Wert muss so gesetzt werden, dass er anzeigt, wie viel Byte tatsächlich gelesen wurden.

dataptr
Ein Zeiger auf den Datenpuffer.

reserve
Zur zukünftigen Verwendung reserviert.

Hinweise:

Diese Funktion wird vom Sicherungsdienstprogramm verwendet.

Rückkehrcodes:

Tabelle 11. Gültige Rückkehrcodes für sqluvput und resultierende DB2-Aktion

Literal in Headerdatei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
SQLUV_OK	Operation erfolgreich	sqluvput oder sqluvend, sofern beendet (z. B. verfügt DB2 über keine Daten mehr)	Andere Prozesse werden über eine beendete Operation informiert.
SQLUV_COMM_ERROR	Fehler bei der Kommunikation mit einer Einheit	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_INV_ACTION	Ungültige Aktion wurde angefordert.	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_INV_DEV_HANDLE	Ungültige Einheitenkennung	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_INV_BUFF_SIZE	Ungültige Puffergröße	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_ENDOFMEDIA	Datenträgerende erreicht, z. B. Ende des Bandes	sqluvend	
SQLUV_DATA_RESEND	Anforderung von Einheit, den Pufferinhalt erneut zu senden	sqluvput	DB2 überträgt den letzten Pufferinhalt erneut. Dies erfolgt nur einmal.
SQLUV_DEV_ERROR	Einheitenfehler	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_WARNING	Achtung. Dies sollten Sie nicht verwenden, um DB2 das Datenträgerende anzugeben; verwenden Sie dazu SQLUV_ENDOFMEDIA. Allerdings können Bedingungen *Nicht bereit* von Einheiten mit diesem Rückkehrcode angezeigt werden.	sqluvput	
SQLUV_LINK_NOT_EXIST	Momentan ist keine Programmverbindung vorhanden.	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.
SQLUV_IO_ERROR	E/A-Fehler	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.

Nächster Aufruf:

[a] Wenn der nächste Aufruf sqluvend, action = SQLU_ABORT ist, werden diese Sitzung und alle übrigen aktiven Sitzungen beendet. Festgeschriebene Sitzungen werden mit einer Aufruffolge von sqluvint, sqluvdel und sqluvend gelöscht.

sqluvend - Verbindung zu Einheit aufheben und Ressourcen freigeben

Beendet eine Einheit oder hebt die Verbindung zu ihr auf und gibt alle zugehörigen Ressourcen frei. Das Produkt des anderen Lieferanten muss vor der Rückkehr zu DB2 ungenutzte Ressourcen (z. B. zugeordneten Speicherbereich und Dateikennungen) freigeben.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *dbadm*

Erforderliche Verbindung:

Datenbank

API-Includedatei:

sql.h

Syntax der C-API:

```
/* File: sqluvend.h */
/* API: Unlink the Device and Release its Resources */
/* ... */
int sqluvend (
  sqlint32 action,
  void * pVendorCB,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API-Parameter:

action Eingabe. Zur Festschreibung oder zum Abbruch der Sitzung verwendet:

- SQLUV_COMMIT (0 = Festschreibung)
- SQLUV_ABORT (1 = Abbruch)

pVendorCB
Eingabe. Zeiger auf die Struktur Init_output.

Init_output
Ausgabe. Speicherbereich für Init_output wird freigegeben. Die Daten wurden festgeschrieben, damit Speicher für eine Sicherung frei ist, wenn action = SQLUV_COMMIT. Die Daten werden für eine Sicherung freigegeben, wenn action = SQLUV_ABORT.

Return_code
Ausgabe. Der Rückkehrcode aus dem API-Aufruf.

Hinweise:

Diese Funktion wird für jede Sitzung aufgerufen, die geöffnet wurde. Es gibt zwei mögliche Aktionscodes:

- Festschreibung

  Die Ausgabe von Daten in diese Sitzung oder das Lesen von Daten aus der Sitzung ist beendet.

  Schreibsitzung (Sicherung): Wenn das Produkt eines anderen Lieferanten an DB2 den Rückkehrcode SQLUV_OK zurückgibt, geht DB2 davon aus, dass die Ausgabedaten vom Produkt eines anderen Lieferanten in geeigneter Weise gespeichert wurden und dass darauf zugegriffen werden kann, wenn ein späterer Aufruf von **sqluvint** erfolgt.

  Lesesitzung (Wiederherstellung): Wenn das Produkt eines anderen Lieferanten an DB2 den Rückkehrcode SQLUV_OK zurückgibt, sollten die Daten nicht gelöscht werden, da sie später möglicherweise erneut verwendet werden müssen. Wenn das Produkt eines anderen Lieferanten SQLUV_COMMIT_FAILED zurückgibt, geht DB2 davon aus, dass es mit der gesamten Sicherungs- und Wiederherstellungsoperation Probleme gibt. Alle aktiven Sitzungen werden von **sqluvend**-Aufrufen mit action = SQLUV_ABORT beendet. Bei einer Sicherungsoperation empfangen festgeschriebene Sitzungen die Aufruffolge **sqluvint**, **sqluvdel** und **sqluvend**.

- Abbruch

  DB2 erkannte einen Fehler, und in der Sitzung werden keine Daten mehr geschrieben oder gelesen.

  Schreibsitzung (Sicherung): Das Produkt eines anderen Lieferanten sollte den partiell ausgegebenen Datensatz löschen und den Rückkehrcode SQLUV_OK verwenden, wenn die partielle Ausgabe gelöscht wird. DB2 geht davon aus, dass mit der gesamten Sicherung Probleme bestehen. Alle aktiven Sitzungen werden von **sqluvend**-Aufrufen mit action = SQLUV_ABORT beendet, und festgeschriebene Sitzungen empfangen die Aufruffolge **sqluvint**, **sqluvdel** und **sqluvend**. Lesesitzung (Wiederherstellung): Das Produkt eines anderen Lieferanten sollte die Daten nicht löschen (da diese möglicherweise erneut gebraucht werden), sondern bereinigen und an DB2 den Rückkehrcode SQLUV_OK zurückgeben. DB2 beendet alle Wiederherstellungssitzungen mit **sqluvend**-Aufrufen mit action = SQLUV_ABORT. Wenn das Produkt eines anderen Lieferanten an DB2 den Code SQLUV_ABORT_FAILED zurückgibt, erhält das aufrufende Programm keine Nachricht über diesen Fehler, da DB2 den ersten schwer wiegenden Fehler zurückgibt und nachfolgende Fehler ignoriert. Damit DB2 in diesem Fall den Aufruf von **sqluvend** mit action = SQLUV_ABORT veranlasst, muss ein erster schwer wiegender Fehler eingetreten sein.

Rückkehrcodes:

Tabelle 12. Gültige Rückkehrcodes für sqluvend und resultierende DB2-Aktion

Literal in Headerdatei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
SQLUV_OK	Operation erfolgreich	Keine weiteren Aufrufe	Geben Sie den Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation.
SQLUV_COMMIT_FAILED	Festschreibungsanforderung fehlgeschlagen	Keine weiteren Aufrufe	Geben Sie den Speicher frei, der dieser Sitzung zugeordnet ist, und beenden Sie die Operation.
SQLUV_ABORT_FAILED	Abbruchanforderung fehlgeschlagen	Keine weiteren Aufrufe	

sqluvdel - Festgeschriebene Sitzung löschen

Löscht festgeschriebene Sitzungen.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *dbadm*

Erforderliche Verbindung:

Datenbank

API-Includedatei:

sqluvend.h

Syntax der C-API:

```
/* File: sqluvend.h */
/* API: Delete Committed Session */
/* ... */
int sqluvdel (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API-Parameter:

Init_input
Eingabe. Init_input und Return_code zugeordneter Speicherbereich.

Return_code
Ausgabe. Rückkehrcode aus dem API-Aufruf. Das Objekt, auf das die Struktur Init_input zeigt, wird gelöscht.

Hinweise:

Wenn mehrere Sitzungen geöffnet sind und einige Sitzungen festgeschrieben sind, jedoch eine davon fehlschlägt, löscht dieser Funktionsaufruf die festgeschriebenen Sitzungen. Es ist keine Folgenummer angegeben; die Aufgabe von **sqluvdel** ist es, alle Objekte zu finden, die während einer spezifischen Sicherungsoperation erstellt wurden, und diese zu löschen. Daten in der Struktur INIT-INPUT werden zur Angabe der zu löschenden Ausgabedaten verwendet. Der Aufruf **sqluvdel** hat die Aufgabe, alle Verbindungen oder Sitzungen einzurichten, die zum Löschen eines Sicherungsobjekts aus der Lieferanteneinheit erforderlich sind. Wenn der Rückkehrcode aus diesem Aufruf SQLUV_DELETE_FAILED lautet, benachrichtigt DB2 das aufrufende Programm nicht, da DB2 den ersten schwer wiegenden Fehler zurückgibt und nachfolgende Fehler ignoriert. Damit DB2 in diesem Fall den Aufruf von **sqluvdel** veranlasst, muss ein erster schwer wiegender Fehler eingetreten sein.

Rückkehrcodes:

Tabelle 13. Gültige Rückkehrcodes für sqluvdel und resultierende DB2-Aktion

Literal in Headerdatei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
SQLUV_OK	Operation erfolgreich	Keine weiteren Aufrufe	
SQLUV_DELETE_FAILED	Löschanforderung ist fehlgeschlagen	Keine weiteren Aufrufe	

db2VendorQueryApiVersion - Von Einheit unterstützte API-Stufe abfragen

Diese Funktion wird aufgerufen, um zu ermitteln, welche Stufe der Lieferanten-API von der Lieferantenbibliothek unterstützt wird. Ist die Lieferantenbibliothek nicht mit DB2 kompatibel, wird diese Lieferantenbibliothek nicht verwendet.

Wenn eine Lieferantenbibliothek diese API nicht für Protokolle implementiert hat, kann die Lieferantenbibliothek nicht verwendet werden, und DB2 meldet einen Fehler. Dies hat keine Auswirkungen auf die Images, die aktuell mit vorhandenen Lieferantenbibliotheken arbeiten.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *dbadm*

Erforderliche Verbindung:

Datenbank.

API-Include-Datei:

db2VendorApi.h

Syntax der C-API:

```
void db2VendorQueryApiVersion(db2Uint32 *supportedVersion);
```

API-Parameter:

supportedVersion
Ausgabe. Gibt die Version der Lieferanten-API zurück, die die Lieferantenbibliothek unterstützt.

Hinweise:

Diese Funktion wird aufgerufen, bevor andere Lieferanten-APIs aufgerufen werden.

db2VendorGetNextObj - Nächstes Objekt für Einheit abrufen

Diese Funktion wird nach dem Definieren einer Abfrage (mit sqluvint) aufgerufen, um das nächste Objekt abzurufen, das den Suchbedingungen entspricht. Es kann jeweils nur eine Suche für entweder Images oder Protokolldateien definiert werden.

Berechtigung:

Eine der folgenden Berechtigungen:

- *sysadm*
- *dbadm*

Erforderliche Verbindung:

Datenbank.

API-Include-Datei:

db2VendorApi.h

Syntax der C-API:

```
int db2VendorGetNextObj(void *vendorCB,
  struct db2VendorQueryInfo *queryInfo,
  struct Return_code *returnCode);

typedef struct db2VendorQueryInfo
{
   char               db2Instance[SQL_INSTNAME_SZ + 1];
   char               dbname[SQL_DBNAME_SZ + 1];
   char               dbalias[SQL_ALIAS_SZ + 1];
   char               timestamp[SQLU_TIME_STAMP_LEN + 1];
   char               filename[DB2VENDOR_MAX_FILENAME_SZ + 1];
   char               owner[DB2VENDOR_MAX_OWNER_SZ + 1];
   char               mgmtClass[DB2VENDOR_MAX_MGMTCLASS_SZ + 1];
   char               oldestLogfile[DB2_LOGFILE_NAME_LEN + 1];
   db2Uint16          sequenceNum
   SQL_PDB_NODE_TYPE  dbPartitionNum;
   db2Uint32          type;
   db2Uint64          sizeEstimate;
} db2VendorQueryInfo;
```

API-Parameter:

vendorCB
Eingabe. Zeiger auf einen von der Lieferantenbibliothek zugeordneten Speicherbereich.

queryInfo
Ausgabe. Zeiger auf eine Struktur *db2VendorQueryInfo*, die von der der Lieferantenbibliothek ausgefüllt werden soll.

returnCode
Ausgabe. Der Rückkehrcode aus dem API-Aufruf.

db2Instance
Gibt den Namen des Exemplars an, zu dem das Objekt gehört.

dbname
Gibt den Namen der Datenbank an, zu der das Objekt gehört.

dbalias
Gibt den Aliasnamen der Datenbank an, zu der das Objekt gehört.

timestamp
Gibt die Zeitmarke an, mit der das Sicherungsimage angegeben wird. Nur gültig, wenn das Objekt ein Sicherungsimage ist.

dateiname
Gibt den Namen des Objekts an, wenn das Objekt eine Ladekopie oder eine Archivprotokolldatei ist.

owner Gibt den Eigner des Objekts an.

mgmtClass
Gibt die Managementklasse an, unter der das Objekt gespeichert wurde (wird von TSM verwendet).

oldestLogfile
Gibt die älteste Protokolldatei an, die in einem Sicherungsimage gespeichert ist.

sequenceNum
Gibt die Dateierweiterung für das Sicherungsimage an. Nur gültig, wenn das Objekt ein Sicherungsimage ist.

dbPartitionNum
Gibt die Nummer der Datenbankpartition an, zu der das Objekt gehört.

type Gibt den Imagetyp an, wenn das Onjekt ein Sicherungsimage ist.

sizeEstimate
Gibt die geschätzte Größe des Objekts an.

Hinweise:

Nicht alle Felder sind für jedes Objekt oder jeden Hersteller zutreffend. Die Musseingabefelder sind db2Instance, dbname, dbalias, timestamp (für Images), filename (für Protokolle und Ladekopieimages), owner, sequenceNum (für Images) und dbPartitionNum. Die übrigen Felder müssen vom jeweiligen Lieferanten definiert werden. Wenn ein Feld nicht zutrifft, sollte es mit dem Wert "" für Zeichenfolgen und 0 für numerische Typen initialisiert werden.

DB2-INFO

Diese Struktur enthält Daten, die DB2 gegenüber der Einheit des anderen Lieferanten definieren.

Tabelle 14. Felder in der Struktur DB2-INFO. Alle Felder sind auf Null endende Zeichenfolgen.

Feldname	Datentyp	Beschreibung
DB2_id	char	Eine Kennung für das DB2-Produkt. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
version	char	Die aktuelle Version des DB2-Produkts. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
release	char	Das aktuelle Release des DB2-Produkts. Dieser Wert ist auf NULL gesetzt, sofern er irrelevant ist. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
level	char	Die aktuelle Stufe des DB2-Produkts. Dieser Wert ist auf NULL gesetzt, sofern er irrelevant ist. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
action	char	Gibt die Aktion an, die ausgeführt werden muss. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist ein Zeichen.
filename	char	Der Dateiname, mit dem das Sicherungsimage angegeben wird. Sollte der Wert NULL sein, wird das Sicherungsimage mit *server_id, db2instance, dbname* und *timestamp* eindeutig angebeben. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist 255 Zeichen.
server_id	char	Ein eindeutiger Name, der den Server angibt, auf dem sich die Datenbank befindet. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
db2instance	char	Die db2instance-ID. Dies ist die ID des Benutzers, der den Befehl aufruft. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.

type	char	Gibt den Typ der Sicherung oder der Wiederherstellung an, die gerade ausgeführt wird. Mögliche Werte:
		
		Wenn action = SQLUV_WRITE ist:
		`0 - Datenbankgesamtsicherung`
		`3 - Sicherung auf Tabellenbereichsebene`
		
		Wenn action = SQLUV_READ ist:
		`0 - Vollständige Wiederherstellung`
		`3 - Onlinewiederherstellung eines Tabellenbereichs`
		`4 - Wiederherstellung eines Tabellenbereichs`
		`5 - Wiederherstellung einer Protokolldatei`

Tabelle 14. Felder in der Struktur DB2-INFO (Forts.). Alle Felder sind auf Null endende Zeichenfolgen.

Feldname	Datentyp	Beschreibung
dbname	char	Der Name der Datenbank, die gesichert oder wiederhergestellt werden soll. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
alias	char	Der Aliasname der Datenbank, die gesichert oder wiederhergestellt werden soll. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
timestamp	char	Die Zeitmarke, mit der das Sicherungsimage angegeben wird. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist 26 Zeichen.
sequence	char	Gibt die Dateierweiterung für das Sicherungsimage an. Für Schreiboperationen lautet der Wert für die erste Sitzung 1, und jedes Mal, wenn mit einem sqluvint-Aufruf eine weitere Sitzung eingeleitet wird, erhöht sich der Wert um 1. Für Leseoperation ist der Wert immer null. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist 3 Zeichen.
obj_list	struct sqlu_gen_list	Zur zukünftigen Verwendung reserviert.
max_bytes_per_txn	sqlint32	Gibt für das Produkt eines anderen Lieferanten die Übertragungspuffergröße, die der Benutzer angab, in Byte an.
image_filename	char	Zur zukünftigen Verwendung reserviert.
reserve	void	Zur zukünftigen Verwendung reserviert.
nodename	char	Name des Knotens, auf dem die Sicherung generiert wurde.
password	char	Kennwort des Knotens, auf dem die Sicherung generiert wurde.
owner	char	ID des Erstellers der Sicherung.
mcNameP	char	Verwaltungsklasse.
nodeNum	SQL_PDB_NODE_TYPE	Knotennummer. Die Lieferantenschnittstelle unterstützt Zahlen größer als 255.

Mit *filename* oder *server_id*, *db2instance*, *type*, *dbname* und *timestamp* ist das Sicherungsimage eindeutig angegeben. Die Folgenummer (*sequence*) gibt die Dateierweiterung an. Wenn Sie ein Sicherungsimage wiederherstellen müssen, müssen Sie zum Abruf des Sicherungsimages dieselben Werte angeben. Wenn Sie *filename* verwenden, können Sie abhängig vom Produkt eines anderen Lieferanten die übrigen Parameter auf NULL setzen und umgekehrt.

Sprachsyntax:

C-Struktur

```
/* File: sqluvend.h */
/* ... */
typedef struct DB2_info
{
  char                  *DB2_id;
  char                  *version;
  char                  *release;
  char                  *level;
  char                  *action;
  char                  *filename;
  char                  *server_id;
  char                  *db2instance;
  char                  *type;
  char                  *dbname;
  char                  *alias;
  char                  *timestamp;
  char                  *sequence;
  struct sqlu_gen_list  *obj_list;
  long                  max_bytes_per_txn;
  char                  *image_filename;
  void                  *reserve;
  char                  *nodename;
  char                  *password;
  char                  *owner;
  char                  *mcNameP;
  SQL_PDB_NODE_TYPE     nodeNum;
} DB2_info;
/* ... */
```

VENDOR-INFO

Diese Struktur enthält Daten, die den Lieferanten und die Version der Einheit angeben.

Tabelle 15. Felder in der Struktur VENDOR-INFO. Alle Felder sind auf Null endende Zeichenfolgen.

Feldname	Datentyp	Beschreibung
vendor_id	char	Eine Kennung für den Lieferanten. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist 64 Zeichen.
version	char	Die aktuelle Version des Produkts eines anderen Lieferanten. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
release	char	Das aktuelle Release des Produkts eines anderen Lieferanten. Dieser Wert ist auf NULL gesetzt, sofern er irrelevant ist. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
level	char	Die aktuelle Stufe des Produkts eines anderen Lieferanten. Dieser Wert ist auf NULL gesetzt, sofern er irrelevant ist. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
server_id	char	Ein eindeutiger Name, der den Server angibt, auf dem sich die Datenbank befindet. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist acht Zeichen.
max_bytes_per_txn	sqlint32	Die maximal unterstützte Größe des Übertragungspuffers. Der Lieferant gibt diesen Wert in Byte an. Dies wird nur verwendet, wenn der Rückkehrcode aus der Initialisierungsfunktion des anderen Lieferanten SQLUV_BUFF_SIZE lautet, d. h., dass eine ungültige Puffergröße angegeben wurde.
num_objects_in_backup	sqlint32	Die Anzahl der Sitzungen, die zur Erstellung einer vollständigen Sicherung verwendet wurden. Dies dient der Bestimmung, wann alle Sicherungsimages bei einer Wiederherstellungsoperation verarbeitet worden sind.
reserve	void	Zur zukünftigen Verwendung reserviert.

Sprachsyntax:

C-Struktur

```
typedef struct Vendor_info
{
  char       *vendor_id;
  char                  *version;
  char                  *release;
  char                  *level;
  char       *server_id;
  sqlint32   max_bytes_per_txn;
  sqlint32   num_objects_in_backup;
  void                  *reserve;
} Vendor_info;
```

INIT-INPUT

Diese Struktur enthält von DB2 gelieferte Daten, mit denen eine logische Verbindung zu der Einheit des anderen Lieferanten eingerichtet und hergestellt wird.

Tabelle 16. Felder in der Struktur INIT-INPUT. Alle Felder sind auf Null endende Zeichenfolgen.

Feldname	Datentyp	Beschreibung
DB2_session	struct DB2_info	Eine Beschreibung der Sitzung aus der Perspektive von DB2.
size_options	unsigned short	Die Länge des Feldes für Optionen. Wenn Sie die DB2-Sicherungs- oder die DB2-Wiederherstellungsfunktion verwenden, werden die Daten in diesem Feld direkt vom Parameter *VendorOptionsSize* übergeben.
size_HI_order	sqluint32	Höherwertige 32 Bit der geschätzten Datenbankgröße in Byte. Die Gesamtgröße ist 64 Bit.
size_LOW_order	sqluint32	Niederwertige 32 Bit der geschätzten Datenbankgröße in Byte. Die Gesamtgröße ist 64 Bit.
options	void	Diese Daten werden von der Anwendung übergeben, wenn die Sicherungs- oder die Wiederherstellungsfunktion aufgerufen wird. Diese Datenstruktur muss unstrukturiert sein, d. h., Zwischenstufen werden nicht unterstützt. Es findet keine Bytefolgeumkehrung statt, und die Codepage für diese Daten wird nicht überprüft. Wenn Sie die DB2-Sicherungs- oder die DB2-Wiederherstellungsfunktion verwenden, werden die Daten in diesem Feld direkt vom Parameter *pVendorOptions* übergeben.
reserve	void	Zur zukünftigen Verwendung reserviert.
prompt_lvl	char	Ebene der Bedienerführung, die der Benutzer anfordert, wenn er eine Sicherungs- oder Wiederherstellungsoperation aufruft. Die maximale Länge der Zeichenfolge, auf die dieser Wert zeigt, ist ein Zeichen.
num_sessions	unsigned short	Anzahl der Sitzungen, die der Benutzer anfordert, wenn er eine Sicherungs- oder Wiederherstellungsoperation aufruft.

Sprachsyntax:

C-Struktur

```
typedef struct Init_input
{
  struct DB2_info  *DB2_session;
  unsigned short   size_options;
  sqluint32        size_HI_order;
  sqluint32        size_LOW_order;
  void             *options;
  void             *reserve;
  char             *prompt_lvl;
  unsigned short   num_sessions;
} Init_input;
```

INIT-OUTPUT

Diese Struktur enthält die von der Einheit des anderen Lieferanten zurückgegebene Ausgabe.

Tabelle 17. Felder in der Struktur INIT-OUTPUT

Feldname	Datentyp	Beschreibung
vendor_session	struct Vendor_info	Enthält Angaben zum Lieferanten für DB2.
pVendorCB	void	Lieferantensteuerblock.
reserve	void	Zur zukünftigen Verwendung reserviert.

Sprachsyntax:

C-Struktur

```
typedef struct Init_output
{
  struct Vendor_info  *vendor_session;
  void                *pVendorCB;
  void             *reserve;
} Init_output;
```

DATA

Diese Struktur enthält Daten, die zwischen DB2 und der Einheit des anderen Lieferanten ausgetauscht werden.

Tabelle 18. Felder in der Struktur DATA

Feldname	Datentyp	Beschreibung
obj_num	sqlint32	Die Folgenummer, die DB2 während einer Sicherung zuordnet.
buff_size	sqlint32	Die Größe des Puffers.
actual_buf_size	sqlint32	Die tatsächliche Anzahl Byte, die übertragen oder empfangen wurden. Dieser Wert darf den Wert für *buff_size* nicht überschreiten.
dataptr	void	Zeiger auf den Datenpuffer. DB2 ordnet dem Puffer einen Speicherbereich zu.
reserve	void	Zur zukünftigen Verwendung reserviert.

Sprachsyntax:

C-Struktur

```
typedef struct Data
{
   sqlint32  obj_num;
   sqlint32  buff_size;
   sqlint32  actual_buff_size;
   void      *dataptr;
  void             *reserve;
} Data;
```

RETURN-CODE

Diese Struktur enthält den Rückkehrcode und eine kurze Erläuterung des Fehlers, der gerade an DB2 zurückgegeben wurde.

Tabelle 19. Felder in der Struktur RETURN-CODE

Feldname	Datentyp	Beschreibung
return_code[a]	sqlint32	Rückkehrcode aus der Funktion des Produkts eines anderen Lieferanten.
description	char	Eine kurze Beschreibung des Rückkehrcodes.
reserve	void	Zur zukünftigen Verwendung reserviert.
[a] Dies ist ein herstellerspezifischer Rückkehrcode, der nicht derselbe ist wie der Wert, der von verschiedenen DB2-APIs zurückgegeben wird. Weitere Informationen zu den Rückkehrcodes, die von Produkten anderer Lieferanten akzeptiert werden, finden Sie in den einzelnen API-Beschreibungen.		

Sprachsyntax:

C-Struktur

```
typedef struct Return_code
{
  sqlint32  return_code,
  char      description[30],
  void      *reserve,
} Return_code;
```

APIs für komprimierte Sicherungen

Schnittstelle für Komprimierungs-Plug-in

DB2 stellt die Definition für die Struktur COMPR_DB2INFO zur Verfügung, die Definitionen für die anderen nachstehend aufgeführten Strukturen und APIs werden jeweils vom Hersteller bereitgestellt. Die folgenden Strukturen, Prototypen und Konstanten sind in der Datei `sqlucompr.h` definiert, die im Lieferumfang von DB2 enthalten ist.

Beschreibung der DB2-Umgebung - COMPR_DB2INFO:

```
struct COMPR_DB2INFO {
   char    tag[16];
   db2Uint32    version;
   db2Uint32    size;
   char         dbalias[SQLU_ALIAS_SZ+1];
   char         instance[SQL_INSTNAME_SZ+1];
   SQL_PDB_NODE_TYPE node;
   SQL_PDB_NODE_TYPE catnode;
   char         timestamp[SQLU_TIME_STAMP_LEN+1];
   db2Uint32    bufferSize;
   db2Uint32    options;
   db2Uint32    bkOptions;

   db2Uint32    db2Version;
   db2Uint32    platform;
   db2int32     comprOptionsByteOrder;
   db2Uint32    comprOptionsSize;
   void        *comprOptions;
   db2Uint32    savedBlockSize;
   void        *savedBlock;
};
```

COMPR_DB2INFO

DB2 wird diese Struktur zuordnen und definieren und als einen Parameter an die APIs InitCompression und InitDecompression übergeben. Diese Struktur beschreibt die Datenbank, die gesichert oder wiederhergestellt wird, und stellt Informationen zur DB2-Umgebung bereit, in der diese Operation stattfindet. Die Struktur enthält die folgenden Felder:

tag[16]
Wird als Strukturkennung verwendet. Dieses Feld wird immer auf die Zeichenfolge ″COMPR_DB2INFO \0″ gesetzt.

version
Gibt die Version der Struktur an, sodass APIs die Präsenz zusätzlicher Felder anzeigen können. Die derzeitige Version ist 1. Dieser Struktur werden in Zukunft möglicherweise weitere Felder hinzugefügt.

size Gibt die Größe der Struktur COMPR_DB2INFO in Byte an.

dbalias[SQLU_ALIAS_SZ+1]

instance[SQL_INSTNAME_SZ+1]

node

catnode

timestamp[SQLU_TIME_STAMP_LEN+1]
Beschreibt die Datenbank, die gesichert oder wiederhergestellt wird. Diese Felder werden zum Benennen des Sicherungsimages verwendet. Für Wiederherstellungsoperationen bezieht sich *dbalias* auf den Aliasnamen der Quellendatenbank.

bufferSize
Gibt die Größe eines Übertragungspuffers an (in 4-KB-Seiten).

options
Das Feld *iOptions*, das in der API db2Backup oder db2Restore angegeben wird.

bkOptions
Gibt für Wiederherstellungsoperationen das Feld *iOptions* an, das beim Erstellen der Sicherung in der API db2Backup verwendet wurde. Für Sicherungsoperationen ist dieses Feld auf null gesetzt.

db2Version
Gibt die Version der DB2-Steuerkomponente an.

platform
Gibt die Plattform an, auf der die DB2-Steuerkomponente aktiv ist. Der Wert ist einer der in <sqlmon.h> aufgelisteten Werte.

comprOptionsByteOrder
Gibt die Byteanordnung an, die auf dem Client verwendet wird, auf dem die API ausgeführt wird. DB2 nimmt keine Interpretation oder Konvertierung der als *comprOptions* weitergegebenen Daten vor. Daher sollte mit diesem Feld festgelegt werden, ob für die Daten vor der Verwendung eine Bytefolgeumkehrung erforderlich ist oder nicht. Eine gegebenenfalls erforderliche Konvertierung muss von der Plug-in-Bibliothek selbst vorgenommen werden.

comprOptionsSize
Gibt den Wert des Parameters *piComprOptionsSize* in den APIs db2Backup und db2Restore an.

***comprOptions**
Gibt den Wert des Felds *piComprOptions* in den APIs db2Backup und db2Restore an.

savedBlockSize

***savedBlock**
DB2 ermöglicht es der Plug-in-Bibliothek, einen beliebigen Datenblock im Sicherungsimage zu speichern. Wenn in einem Sicherungsimage ein solcher Datenblock gespeichert wurde, wird er bei der Wiederherstellungsoperation in diesen Feldern zurückgegeben. Für Sicherungsoperationen sind diese Felder auf null gesetzt.

Beschreibung des Plug-ins - COMPR_PIINFO:

```
struct COMPR_PIINFO {
   char    tag[16];
   db2Uint32    version;
   db2Uint32    size;
   db2Uint32    useCRC;
   db2Uint32    useGran;
   db2Uint32    useAllBlocks;
   db2Uint32    savedBlockSize;
};
```

COMPR_PIINFO
Diese Struktur wird von der Plug-in-Bibliothek verwendet, um sich DB2 gegenüber selbst zu beschreiben. Diese Struktur wird von DB2 zugeordnet und initialisiert, und die Schlüsselfelder werden von der Plug-in-Bibliothek beim Aufruf von InitCompression mit Daten gefüllt.

tag[16]
Wird als Strukturkennung verwendet. (Wird von DB2 festgelegt.) Dieses Feld wird immer auf die Zeichenfolge "COMPR_PIINFO \0" gesetzt.

version
Gibt die Version der Struktur an, sodass APIs die Präsenz zusätzlicher Felder anzeigen können. Die derzeitige Version ist 1. (Wird von DB2 festgelegt.) Dieser Struktur werden in Zukunft möglicherweise weitere Felder hinzugefügt.

size Gibt die Größe der Struktur COMPR_PIINFO an (in Byte). (Wird von DB2 festgelegt.)

useCRC
DB2 ermöglicht es, Komprimierungs-Plug-ins, die Integrität der komprimierten oder dekomprimierten Daten mit einer 32-Bit-CRC oder einem Kontrollsummenwert zu prüfen. Wenn die Bibliothek eine solche Prüfung verwendet, setzt sie dieses Feld auf 1. Andernfalls wird das Feld auf 0 gesetzt.

useGran
Wenn die Komprimierungsroutine in der Lage ist, Daten in Inkrementen beliebiger Größe zu komprimieren, setzt die Bibliothek dieses Feld auf 1. Wenn die Komprimierungsroutine Daten nur in Inkrementen in Bytegröße komprimieren kann, setzt die Bibliothek dieses Feld auf 0. Details zur Festlegung dieses Werts finden Sie in

der Beschreibung des Parameters *useGran* der API Compress. Für Wiederherstellungsoperationen wird dieses Feld ignoriert.

useAllBlocks
Gibt an, ob DB2 einen komprimierten Datenblock sichern soll, der größer als der ursprüngliche, nicht komprimierte Block ist. Standardmäßig speichert DB2 Daten ohne Komprimierung, wenn die komprimierte Version größer ist, aber unter bestimmten Umständen wird die Plug-in-Bibliothek eine Speicherung der Daten in komprimierter Form bevorzugen. Wenn DB2 die komprimierte Version der Daten für alle Blöcke speichern soll, setzt die Bibliothek diesen Wert auf 1. Wenn DB2 nur dann eine komprimierte Version der Daten speichern soll, wenn diese kleiner als die ursprünglichen Daten ist, setzt die Bibliothek diesen Wert auf 0. Für Wiederherstellungsoperationen wird dieses Feld ignoriert.

savedBlockSize
DB2 ermöglicht es der Plug-in-Bibliothek, einen beliebigen Datenblock im Sicherungsimage zu speichern. Wenn in einem Sicherungsimage ein solcher Datenblock gespeichert wurde, setzt die Bibliothek dieses Feld auf die Größe des Blocks, der für diese Daten zugeordnet werden soll. (Die eigentlichen Daten werden in einem anschließenden API-Aufruf an DB2 übergeben.) Wenn keine Daten gespeichert werden sollen, setzt die Plug-in-Bibliothek dieses Feld auf null. Für Wiederherstellungsoperationen wird dieses Feld ignoriert.

Beschreibung des Steuerblocks - COMPR_CB:

```
struct COMPR_CB;

extern "C" {

int InitCompression(
      const COMPR_DB2INFO *db2Info,
      COMPR_PIINFO  *piInfo,
      COMPR_CB  **pCB);

int GetSavedBlock(
   COMPR_CB               *pCB,
   db2Uint32               blockSize,
      void       *data);

int Compress(
   COMPR_CB               *pCB,
      const char *src,
      db2int32    srcLen,
      db2Uint32   srcGran,
      char       *tgt,
      db2int32    tgtSize,
      db2int32   *srcAct,
      db2int32   *tgtAct,
      db2Uint32  *tgtCRC);

int GetMaxCompressedSize(
   COMPR_CB               *pCB,
      db2Uint32   srcLen);

int TermCompression(
      COMPR_CB   *pCB);

int InitDecompression(
      const COMPR_DB2INFO *db2Info,
```

```
      COMPR_CB  **pCB);

int Decompress(
   COMPR_CB              *pCB,
      const char *src,
      db2int32    srcLen,
      char       *tgt,
      db2int32    tgtSize,
      db2int32   *tgtAct,
      db2Uint32  *tgtCRC);

int TermDecompression(
      COMPR_CB   *pCB);
}
```

COMPR_CB

Dies ist eine Struktur, die von der Plug-in-Bibliothek intern verwendet wird. Sie enthält Daten, die intern von Komprimierungs- und Dekomprimierungsroutinen verwendet werden. DB2 übergibt diese Struktur an jeden Aufruf, den es an die Plug-in-Bibliothek richtet. Alle Aspekte der Struktur, einschließlich der Definition der Strukturfelder und der Speicherverwaltung der Struktur, werden jedoch von der Bibliothek bestimmt.

```
int InitCompression(
      const COMPR_DB2INFO    *db2Info,
      COMPR_PIINFO           *piInfo,
      COMPR_CB              **pCB);
```

Initialisiert die Komprimierungsbibliothek. DB2 übergibt die Strukturen *db2Info* und *piInfo*. Die Bibliothek füllt die entsprechenden Felder von piInfo aus und wird *pCB* zuordnen und einen Zeiger auf den zugeordneten Speicher setzen.

```
int GetSavedBlock(
      COMPR_CB              *pCB,
      db2Uint32              blockSize,
      void                  *data);
```

Ruft den herstellerspezifischen Datenblock ab, der im Sicherungsimage gespeichert werden soll. Wenn die Bibliothek einen Wert ungleich Null für piInfo->savedBlockSize zurückgibt, wird DB2 GetSavedBlock aufrufen und diesen Wert für *blockSize* verwenden. Die Plug-in-Bibliothek schreibt Daten der angegebenen Größe in den von *data* angegebenen Speicher. Diese Funktion wird während der Anfangsdatenverarbeitung in BM1 nur für Sicherungen aufgerufen. Auch wenn eine Parallelität > 1 in der API db2Backup angegeben wird, wird diese Funktion bei jeder Sicherung nur einmal aufgerufen.

```
int Compress(
      COMPR_CB              *pCB,
      const char            *src,
      db2int32               srcLen,
      db2int32               srcGran,
      char                  *tgt,
      db2int32               tgtSize,
      db2int32              *srcAct,
      db2int32              *tgtAct,
      db2Uint32             *tgtCRC);
```

Einen Datenblock komprimieren. *src* zeigt auf einen Datenblock, der *srcLen* Byte groß ist. *tgt* zeigt auf einen Puffer, der *tgtSize* Byte groß ist. Die Plug-in-Bibliothek komprimiert die Daten an Adresse *src* und schreibt die komprimierten Daten in den Puffer an Adresse *tgt*. Die tatsächliche Größe der komprimierten dekomprimierten Daten wird in *srcAct* gespeichert. Die tatsächliche Größe der komprimierten Daten wird als *tgtAct* zurückgegeben.

Wenn die Bibliothek den Wert 1 für piInfo->useCRC zurückgegeben hat, wird der CRC-Wert des dekomprimierten Blocks als *tgtCRC* zurückgegeben. Wenn die Bibliothek den Wert 0 für piInfo->useCRC zurückgegeben hat, ist *tgtCRC* ein Nullzeiger.

Wenn die Bibliothek den Wert 1 für piInfo->useGran zurückgegeben hat, gibt *srcGran* den log2 der Seitengröße der Daten an. (Wenn die Seitengröße der Daten beispielsweise 4096 Byte beträgt, hat *srcGran* den Wert 12.) Die Bibliothek stellt sicher, dass das Volumen der tatsächlich komprimierten Daten (*srcAct*) ein genaues Vielfaches dieser Seitengröße ist. Wenn die Bibliothek die Markierung *useGran* setzt, kann DB2 einen effizienteren Algorithmus auswählen, um die komprimierten Daten in das Sicherungsimage einzufügen. Dies verbessert nicht nur die Leistung des Plug-ins, sondern verringert auch die Größe des komprimierten Sicherungsimages. Wenn die Bibliothek den Wert 0 für piInfo->srcGran zurückgegeben hat, beträgt die Granularität 1 Byte.

```
int GetMaxCompressedSize(
      COMPR_CB              *pCB,
      db2Uint32              srcLen,
      db2Uint32             *tgtLen);
```

Schätzt die maximal mögliche Größe des Puffers, der zum Komprimieren eines Datenblocks erforderlich ist. *srcLen* gibt die Größe eines Datenblocks an, der komprimiert werden soll. Die Bibliothek gibt die theoretische Maximalgröße des Puffers nach der Komprimierung als *tgtLen* zurück.

DB2 verwendet den als *tgtLen* zurückgegebenen Wert zur Optimierung der internen Speicherverwendung. Wenn kein Wert oder ein ungünstiger Wert berechnet wird, hat dies den Nachteil, dass DB2 die API Compress für einen einzelnen Datenblock mehrmals aufrufen muss oder dass Speicher des Zwischenspeichers für Dienstprogramme verschwendet wird. Das Sicherungsimage wird jedoch korrekt erstellt, unabhängig von den zurückgegebenen Werten.

```
int TermCompression(
      COMPR_CB              *pCB);
```

Beendet die Komprimierungsbibliothek. Die Bibliothek gibt den für *pCB* verwendeten Speicher frei.

```
int InitDecompression(
      const COMPR_DB2INFO   *db2Info,
      COMPR_CB             **pCB);
```

Initialisiert die Dekomprimierungsbibliothek. DB2 übergibt die Struktur db2Info. Die Bibliothek ordnet *pCB* zu und gibt einen Zeiger auf den zugeordneten Speicher zurück.

```
int Decompress(
      COMPR_CB              *pCB,
      const char            *src,
      db2int32               srcLen,
      char                  *tgt,
      db2int32               tgtSize,
      db2int32              *tgtLen,
      db2Uint32             *tgtCRC);
```

Dekomprimiert einen Datenblock. *src* zeigt auf einen Datenblock, der *srcLen* Byte groß ist. *tgt* zeigt auf einen Puffer, der *tgtSize* Byte groß ist. Die Plug-in-Bibliothek dekomprimiert die Daten an Adresse *src* und schreibt die dekomprimierten Daten in den Puffer an Adresse *tgt*. Die tatsächliche Größe der dekomprimierten Daten wird als *tgtLen* zurückgegeben. Wenn die Bibliothek den Wert 1 für piInfo->useCRC zurückgegeben hat, wird der CRC-Wert des dekomprimierten Blocks als *tgtCRC*

zurückgegeben. Wenn die Bibliothek den Wert 0 für piInfo->useCRC zurückgegeben hat, ist *tgtLen* ein Nullzeiger.

```
int TermDecompression(
      COMPR_CB   *pCB);
```

Beendet die Dekomprimierungsbibliothek. Die Bibliothek gibt den für *pCB* verwendeten Speicher frei. Der gesamte von diesen APIs intern verwendete Speicher wird vom Hersteller verwaltet. Die Plug-in-Bibliothek verwaltet den von der Struktur COMPR_CB verwendeten Speicher. DB2 verwaltet den für die Datenpuffer (die Parameter *src* und *tgt* in den APIs) verwendeten Speicher.

Rückkehrcodes der Plug-in-Schnittstelle:

Dies sind die Rückkehrcodes, die von den APIs zurückgegeben werden können. Wenn nicht anders angegeben, beendet DB2 die Sicherung oder Wiederherstellung, wenn ein Rückkehrcode ungleich Null zurückgegeben wird.

SQLUV_OK	0	Operation erfolgreich
SQLUV_BUFFER_TOO_SMALL	100	Der Zielpuffer ist zu klein. Wenn bei der Sicherung angegeben, gibt das Feld *tgtAct* die geschätzte Größe an, die zum Komprimieren des Objekts erforderlich ist. DB2 wiederholt die Operation mit einem Puffer, der mindestens die angegebene Größe hat. Wird das Feld bei einer Wiederherstellung angegeben, schlägt die Operation fehl.
SQLUV_PARTIAL_BUFFER	101	Ein Puffer wurde teilweise komprimiert. Wenn bei der Sicherung angegeben, gibt das Feld *srcAct* die tatsächliche Größe des tatsächlich komprimierten Datenvolumens an und das Feld *tgtAct* die tatsächliche Größe der komprimierten Daten. Werden die Felder bei einer Wiederherstellung angegeben, schlägt die Operation fehl.
SQLUV_NO_MEMORY	102	Nicht genügend Hauptspeicher vorhanden.
SQLUV_EXCEPTION	103	Im Code trat ein Signal oder eine Ausnahmebedingung auf.
SQLUV_INTERNAL_ERROR	104	Es ist ein interner Fehler aufgetreten.

Der Unterschied zwischen SQLUV_BUFFER_TOO_SMALL und SQLUV_PARTIAL_BUFFER liegt darin, dass DB2 bei Rückgabe von SQLUV_PARTIAL_BUFFER die Daten im Ausgabepuffer als gültig betrachtet.

Zugehörige Referenzen:

- „db2Backup - Datenbank sichern" auf Seite 85
- „db2Restore - Datenbank wiederherstellen" auf Seite 113

Anhang I. Technische Informationen zu DB2 Universal Database

DB2-Dokumentation und Hilfe

Die technischen Informationen zu DB2® stehen über die folgenden Tools und Methoden zur Verfügung:

- DB2 Information - Unterstützung
  - Themen
  - Hilfe für DB2-Tools
  - Beispielprogramme
  - Lernprogramme
- Für den Download verfügbare PDF-Dateien, PDF-Dateien auf CD und gedruckte Bücher
  - Handbücher
  - Referenzhandbücher
- Befehlszeilenhilfe
  - Hilfe für Befehle
  - Hilfe für Nachrichten
  - Hilfe für SQL-Anweisungen
- Installierter Quellcode
  - Beispielprogramme

Darüber hinaus können Sie auf zusätzliche technische Informationen zu DB2 Universal Database™, wie beispielsweise technische Hinweise (Technotes), White Papers und Redbooks™, online über ibm.com® zugreifen. Rufen Sie die Website 'DB2 Information Management - Library' unter www.ibm.com/software/data/pubs/ auf.

Aktualisierungen der DB2-Dokumentation

In bestimmten Fällen stellt IBM® in regelmäßigen Abständen Dokumentations-Fix-Paks und andere Dokumentationsaktualisierungen für 'DB2 Information - Unterstützung' zur Verfügung. Wenn Sie über http://publib.boulder.ibm.com/infocenter/db2help/ auf 'DB2 Information - Unterstützung' zugreifen, erhalten Sie stets die neuesten Informationen. Falls Sie 'DB2 Information - Unterstützung' lokal installiert haben, müssen Sie alle Aktualisierungen manuell installieren, bevor Sie sie anzeigen können. Diese Dokumentationsaktualisierungen ermöglichen Ihnen, die Informationen, die Sie von der CD mit *DB2 Information - Unterstützung* installiert haben, auf den neuesten Stand zu bringen, sobald neue Informationen verfügbar sind.

'DB2 Information - Unterstützung' wird häufiger aktualisiert als die PDF- und Hardcopy-Bücher. Um stets die jeweils neuesten technischen Informationen zu DB2 zur Verfügung zu haben, sollten Sie die Dokumentationsaktualisierungen installieren, sobald sie verfügbar sind, oder 'DB2 Information - Unterstützung' über die Website www.ibm.com aufrufen.

Zugehörige Konzepte:

- „CLI sample programs" im Handbuch *CLI Guide and Reference, Volume 1*
- „Java sample programs" im Handbuch *Application Development Guide: Building and Running Applications*
- „DB2 Information - Unterstützung" auf Seite 410

Zugehörige Tasks:

- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 429
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 421
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 430
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 431
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 431

Zugehörige Referenzen:

- „DB2-Dokumentation in PDF-Format und gedrucktem Format" auf Seite 422

DB2 Information - Unterstützung

Die DB2®-Komponente 'DB2 Information - Unterstützung' bietet Ihnen die Möglichkeit, auf alle Informationen zuzugreifen, die Sie zur optimalen Nutzung der Produkte innerhalb der DB2-Produktfamilie, wie z. B. DB2 Universal Database™, DB2 Connect™, DB2 Information Integrator und DB2 Query Patroller™, benötigen. 'DB2 Information - Unterstützung' dokumentiert auch die wichtigsten DB2-Funktionen und -Komponenten, einschließlich der Funktionen für die Replikation, das Data Warehousing und die DB2 Extender. Wenn Sie für die Anzeige von 'DB2 Information - Unterstützung' Mozilla ab Version 1.0 oder Microsoft® Internet Explorer ab Version 5.5 verwenden, stehen Ihnen die folgenden Funktionen zur Verfügung. Für bestimmte Funktionen muss die JavaScript™-Unterstützung aktiviert werden :

Flexible Installationsoptionen

Wählen Sie für die Anzeige der DB2-Dokumentation die Option, die Ihren Anforderungen am besten entspricht:

- Stellen Sie ohne großen Aufwand sicher, dass Ihre Dokumentation stets auf dem neuesten Stand ist, indem Sie auf die gesamte Dokumentation direkt über 'DB2 Information - Unterstützung' auf der IBM® Website unter http://publib.boulder.ibm.com/infocenter/db2help/ zugreifen.
- Reduzieren Sie den Aktualisierungsaufwand auf ein Minimum und begrenzen Sie den Datenaustausch auf Ihr Intranet, indem Sie die DB2-Dokumentation auf einem einzigen Server innerhalb Ihres Intranets installieren.
- Erzielen Sie maximale Flexibilität und reduzieren Sie die Abhängigkeit von Netzwerkverbindungen, indem Sie die DB2-Dokumentation auf dem eigenen Computer installieren.

Suchen

Sie können alle Themen in 'DB2 Information - Unterstützung' durchsuchen, indem Sie einen Suchbegriff im Textfeld **Suchen** eingeben. Schließen Sie Begriffe in Anführungszeichen ein, wenn Sie nach exakten Übereinstimmungen suchen möchten. Mit Hilfe von Platzhalterzeichen (*, ?) und Booleschen Operatoren (AND, NOT, OR) können Sie die Suche eingrenzen.

Aufgabenorientiertes Inhaltsverzeichnis

Die Themen in der DB2-Dokumentation können über ein zentrales Inhaltsverzeichnis lokalisiert werden. Das Inhaltsverzeichnis ist primär auf der Basis übergeordneter Aufgabenbereiche aufgebaut, enthält jedoch auch Einträge für Produktübersichten, Ziele, Referenzinformationen sowie einen Index und ein Glossar.

- Produktübersichten beschreiben die Beziehung zwischen den in der DB2-Produktfamilie verfügbaren Produkten sowie die von den einzelnen Produkten bereitgestellten Funktionen und enthalten darüber hinaus die neuesten Release-Informationen für diese Produkte.
- Aufgabenkategorien, wie z. B. Installation, Verwaltung und Entwicklung, umfassen Themen, mit deren Hilfe Sie die einzelnen Aufgaben schnell ausführen und sich außerdem genauere Kenntnisse über die Hintergrundinformationen zu diesen Aufgaben verschaffen können.
- In den Referenzthemen finden Sie detaillierte Informationen zu einem Thema, einschließlich der Anweisungs- und Befehlssyntax, der Hilfetexte zu Nachrichten und der Konfigurationsparameter.

Anzeigen des aktuellen Themas im Inhaltsverzeichnis

Wenn Sie sehen möchten, welchem Bereich des Inhaltsverzeichnisses das aktuelle Thema zugeordnet ist, klicken Sie den Knopf **Aktualisieren / aktuelles Thema anzeigen** im Teilfenster des Inhaltsverzeichnisses oder den Knopf **Im Inhaltsverzeichnis anzeigen** im Inhaltsteilfenster an. Diese Funktion ist zum Beispiel dann von Nutzen, wenn Sie mehreren Links zu zugehörigen Themen in verschiedenen Dateien gefolgt sind oder ein Thema über das Ergebnis einer Suche aufgerufen haben.

Index Über den Index können Sie auf die gesamte Dokumentation zugreifen. Der Index ist alphabetisch nach Indexeinträgen sortiert.

Glossar

Im Glossar finden Sie Definitionen zu Termini, die in der DB2-Dokumentation verwendet werden. Das Glossar ist alphabetisch nach Glossareinträgen sortiert.

Integrierte übersetzte Informationen

Die Informationen in 'DB2 Information - Unterstützung' werden in der Sprache angezeigt, die Sie in den Benutzervorgaben des verwendeten Browsers festgelegt haben. Ist ein Thema nicht in der bevorzugten Sprache verfügbar, wird die englische Version des Themas angezeigt.

Technische Informationen zu iSeries™ finden Sie im Informationszentrum von IBM eServer™ iSeries unter www.ibm.com/eserver/iseries/infocenter/.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 412

Zugehörige Tasks:

- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 421
- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 422
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 419
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 414
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 417

DB2 Information - Unterstützung: Installationsszenarios

Je nach Arbeitsumgebung kann es unterschiedliche Anforderungen hinsichtlich des Zugriffs auf DB2®-Informationen geben. Sie können auf 'DB2 Information - Unterstützung' entweder auf der IBM® Website zugreifen oder auf einem Server im unternehmensinternen Netzwerk oder auf eine auf dem lokalen Computer installierte Version. In allen drei Fällen befindet sich die Dokumentation in 'DB2 Information - Unterstützung', einem strukturierten System themenbasierter Informationen, die über einen Browser angezeigt werden können. Standardmäßig greifen DB2-Produkte auf 'DB2 Information - Unterstützung' auf der IBM Website zu. Wenn Sie jedoch auf 'DB2 Information - Unterstützung' auf einem Intranet-Server oder auf dem eigenen Computer zugreifen möchten, müssen Sie 'DB2 Information - Unterstützung' mit Hilfe der entsprechenden CD installieren, die sich im Programmpaket des Produkts befindet. Anhand der nachfolgenden Übersicht über die verfügbaren Optionen für den Zugriff auf die DB2-Dokumentation und mit Hilfe der drei Installationsszenarios können Sie ermitteln, welche Methode für den Zugriff auf 'DB2 Information - Unterstützung' für Ihre Anforderungen und Arbeitsumgebung am besten geeignet ist und welche Aspekte Sie bei der Installation berücksichtigen müssen.

Übersicht über die verfügbaren Optionen für den Zugriff auf die DB2-Dokumentation:

Die folgende Tabelle enthält Empfehlungen hinsichtlich der für Ihre Arbeitsumgebung geeigneten Optionen für den Zugriff auf die DB2-Produktdokumentation in 'DB2 Information - Unterstützung'.

Internetzugriff	Intranetzugriff	Empfehlung
Ja	Ja	Greifen Sie entweder über die IBM Website auf 'DB2 Information - Unterstützung' zu oder auf die auf einem Intranet-Server installierte Version von 'DB2 Information - Unterstützung'.
Ja	Nein	Greifen Sie über die IBM Website auf 'DB2 Information - Unterstützung' zu.
Nein	Ja	Greifen Sie auf die auf einem Intranet-Server installierte Version von 'DB2 Information - Unterstützung' zu.
Nein	Nein	Greifen Sie auf die auf einem lokalen Computer installierte Version von 'DB2 Information - Unterstützung' zu.

Szenario: Zugriff auf 'DB2 Information - Unterstützung' auf Ihrem Computer:

Tsu-Chen besitzt eine Fabrik in einer Kleinstadt, in der es vor Ort keinen Anbieter für einen Internetzugang gibt. Für die Verwaltung des Lagerbestands, der Produktbestellungen, der Betriebsausgaben und seines Bankkontos hat Tsu-Chen DB2 Universal Database™ gekauft. Da er zuvor noch nie ein DB2-Produkt verwendet hat, muss er anhand der DB2-Produktdokumentation lernen, wie die Verwaltung funktioniert.

Nachdem er DB2 Universal Database mit der Option für die Standardinstallation auf seinem Computer installiert hat, versucht Tsu-Chen, auf die DB2-Dokumentation zuzugreifen. Sein Browser zeigt jedoch eine Fehlernachricht mit der Information an, dass die Seite, die geöffnet werden sollte, nicht gefunden werden kann. Tsu-Chen überprüft das Installationshandbuch für sein DB2-Produkt und findet

heraus, dass er 'DB2 Information - Unterstützung' zunächst installieren muss, um auf seinem Computer auf die DB2-Dokumentation zugreifen zu können. Im Programmpaket findet er die *CD für DB2 Information - Unterstützung* und installiert sie.

Über das Programm zum Aufrufen von Anwendungen für sein Betriebssystem hat Tsu-Chen nun Zugriff auf 'DB2 Information - Unterstützung', um sich mit der Verwendung seines DB2-Produkts vertraut zu machen und so einen wertvollen Beitrag zum Erfolg seines Unternehmens leisten.

Szenario: Zugriff auf 'DB2 Information - Unterstützung' über die IBM Website:

Colin ist IT-Berater bei einer Schulungsfirma. Er ist auf Datenbanktechnologie und SQL spezialisiert und hält Seminare zu diesen Themen für Unternehmen aus ganz Nordamerika ab. Hierfür verwendet er DB2 Universal Database. Im Rahmen seiner Seminare verwendet Colin die DB2-Dokumentation als Unterrichtsmaterial. Für SQL-Kurse beispielsweise verwendet Colin die DB2-Dokumentation zu SQL, um die grundlegende und erweiterte Syntax für Datenbankabfragen zu unterrichten.

Die meisten Unternehmen, bei denen Colin unterrichtet, verfügen über einen Internetzugang. Aus diesem Grund entschied sich Colin, seinen tragbaren Computer für den Zugriff auf 'DB2 Information - Unterstützung' über die Website von IBM zu konfigurieren, als er die letzte Version von DB2 Universal Database installiert hat. Diese Konfiguration ermöglicht es Colin, während seiner Seminare online auf die neueste DB2-Dokumentation zuzugreifen.

Wenn er auf Reisen ist, hat Colin bisweilen allerdings keinen Internetzugang. Dieser Umstand war für ihn recht problematisch, insbesondere dann, wenn er Zugriff auf die DB2-Dokumentation benötigte, um sich auf seine Seminare vorzubereiten. Um Situationen wie diese zu vermeiden, installierte Colin eine Kopie von 'DB2 Information - Unterstützung' auf seinem tragbaren Computer.

Auf diese Weise hat Colin nun jederzeit eine Kopie der DB2-Dokumentation zur Verfügung und ist dadurch wesentlich flexibler. Mit dem Befehl **db2set** kann Colin ohne Schwierigkeiten die Registrierdatenbankvariablen auf seinem tragbaren Computer so konfigurieren, dass er den jeweiligen Umständen entsprechend entweder über die Website von IBM oder über seinen tragbaren Computer auf 'DB2 Information - Unterstützung' zugreifen kann.

Szenario: Zugriff auf 'DB2 Information - Unterstützung' über einen Intranet-Server:

Eva arbeitet als leitende Datenbankadministratorin für eine Lebensversicherung. In ihre Zuständigkeit fallen auch das Installieren und Konfigurieren der neuesten Version von DB2 Universal Database auf den UNIX®-basierten Datenbankservern des Unternehmens. Vor Kurzem hat das Unternehmen seine Mitarbeiter darüber informiert, dass sie aus Sicherheitsgründen während der Arbeitszeit keinen Internetzugriff erhalten würden. Da ihr Unternehmen in einer Netzwerkumgebung arbeitet, beschließt Eva, eine Kopie von 'DB2 Information - Unterstützung' auf einem Intranet-Server zu installieren, damit alle Mitarbeiter, die das Data Warehouse des Unternehmens regelmäßig verwenden (Vertriebsbeauftragte, Vertriebsleiter und Geschäftsanalysten), Zugriff auf die DB2-Dokumentation haben.

Eva weist ihr Datenbankteam an, die neueste Version von DB2 Universal Database auf allen Computern der Mitarbeiter mit Hilfe einer Antwortdatei zu installieren, um sicherzustellen, dass die Konfiguration des Zugriffs auf 'DB2 Information - Unterstützung' auf allen Computern mit dem Hostnamen und der Portnummer des Intranet-Servers erfolgt.

Durch ein Missverständnis installiert jedoch Migual, ein Datenbankadministrator in Evas Team, eine Kopie von 'DB2 Information - Unterstützung' auf mehreren Mitarbeitercomputern, anstatt DB2 Universal Database für den Zugriff auf 'DB2 Information - Unterstützung' über den Intranet-Server zu konfigurieren. Um diesen Fehler zu korrigieren, weist Eva Migual an, mit dem Befehl **db2set** die Registrierdatenbankvariablen von 'DB2 Information - Unterstützung' (DB2_DOCHOST für den Hostnamen und DB2_DOCPORT für die Portnummer) auf allen entsprechenden Computern zu ändern. Anschließend haben nun alle erforderlichen Computer im Netzwerk Zugriff auf 'DB2 Information - Unterstützung', und die Mitarbeiter können mit Hilfe der DB2-Dokumentation Antworten auf ihre Fragen zu DB2 finden.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 410

Zugehörige Tasks:

- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 421
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 414
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 417

Zugehörige Referenzen:

- „db2set - DB2 Profile Registry Command" im Handbuch *Command Reference*

Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)

Es gibt drei Möglichkeiten, auf die DB2-Produktdokumentation zuzugreifen: auf der IBM Website, auf einem Intranet-Server oder auf eine auf dem lokalen Computer installierte Version. Standardmäßig greifen DB2-Produkte auf die DB2-Dokumentation auf der IBM Website zu. Wenn Sie jedoch auf die DB2-Dokumentation auf einem Intranet-Server oder auf dem eigenen Computer zugreifen möchten, müssen Sie die Dokumentation von der *CD 'DB2 Information - Unterstützung'* aus installieren. Mit dem DB2-Installationsassistenten können Sie Ihre Installationseinstellungen definieren und 'DB2 Information - Unterstützung' auf einem Computer installieren, der das Betriebssystem UNIX verwendet.

Voraussetzungen:

Dieser Abschnitt erläutert die Voraussetzungen für Hardware, Betriebssystem, Software und Kommunikation zum Installieren von 'DB2 Information - Unterstützung' auf UNIX-Computern.

- **Hardwarevoraussetzungen**

  Sie benötigen einen der folgenden Prozessoren:

  – PowerPC (AIX)

- HP 9000 (HP-UX)
- Intel 32-Bit (Linux)
- Solaris UltraSPARC-Computer (Solaris-Betriebsumgebung)

- **Betriebssystemvoraussetzungen**

  Sie benötigen eines der folgenden Betriebssysteme:
  - IBM AIX 5.1 (auf PowerPC)
  - HP-UX 11i (auf HP 9000)
  - Red Hat Linux 8.0 (auf Intel 32–Bit)
  - SuSE Linux 8.1 (auf Intel 32–Bit)
  - Sun Solaris Version 8 (auf UltraSPARC-Computern in der Solaris-Betriebsumgebung)

  Anmerkung: 'DB2 Information - Unterstützung' kann unter einem Teil der UNIX-Betriebssysteme ausgeführt werden, unter denen DB2-Clients unterstützt werden. Daher wird empfohlen, entweder über die IBM Website auf 'DB2 Information - Unterstützung' zuzugreifen oder 'DB2 Information - Unterstützung' auf einem Intranet-Server zu installieren und dort auf die Dokumentation zuzugreifen.

- **Softwarevoraussetzungen**
  - Unterstützte Browser:
    - Mozilla Version 1.0 oder höher

- Beim DB2-Installationsassistenten handelt es sich um ein grafisches Installationsprogramm. Um den DB2-Installationsassistenten auf Ihrem Computer ausführen zu können, benötigen Sie eine Implementierung der X Window System-Software zur Wiedergabe einer grafischen Benutzerschnittstelle (GUI). Bevor Sie den DB2-Installationsassistenten ausführen können, müssen Sie die entsprechende Anzeigefunktion (DISPLAY) unbedingt ordnungsgemäß exportieren. Geben Sie hierzu beispielsweise den folgenden Befehl an der Eingabeaufforderung ein:

  ```
  export DISPLAY=9.26.163.144:0.
  ```

- **Kommunikationsvoraussetzungen**
  - TCP/IP

Vorgehensweise:

Um 'DB2 Information - Unterstützung' mit Hilfe des DB2-Installationsassistenten zu installieren, gehen Sie wie folgt vor:

1. Melden Sie sich am System an.
2. Legen Sie die Produkt-CD von 'DB2 Information - Unterstützung' in das CD-Laufwerk ein, und hängen Sie die CD an Ihr System an.
3. Wechseln Sie in das Verzeichnis, in dem die CD angehängt ist. Geben Sie hierzu den folgenden Befehl ein:

   ```
   cd /cd
   ```

   Hierbei steht */cd* für den Mountpunkt der CD.
4. Geben Sie den Befehl **./db2setup** ein, um den DB2-Installationsassistenten zu starten.
5. Die IBM DB2-Klickstartleiste wird geöffnet. Um direkt mit der Installation von 'DB2 Information - Unterstützung' fortzufahren, klicken Sie **Produkt installieren** an. Die Onlinehilfe enthält Informationen, die Sie durch die verbleibenden

Schritte der Installation führen. Um die Onlinehilfe aufzurufen, klicken Sie **Hilfe** an. Sie können jederzeit **Abbrechen** anklicken, um die Installation zu beenden.

6. Klicken Sie im Fenster **Wählen Sie das zu installierende Produkt aus** den Knopf **Weiter** an.
7. Klicken Sie **Weiter** im Fenster **Willkommen beim DB2-Installationsassistenten** an. Der DB2-Installationsassistent leitet Sie durch die erforderlichen Schritte zum Installieren des Programms.
8. Um mit der Installation fortfahren zu können, müssen Sie die Lizenzvereinbarung akzeptieren. Wählen Sie auf der Seite **Lizenzvereinbarung** die Option **Bedingungen in der Lizenzvereinbarung anerkennen** aus, und klicken Sie **Weiter** an.
9. Wählen Sie **DB2 Information - Unterstützung auf diesem Computer installieren** auf der Seite **Installationsaktion auswählen** aus. Wenn Sie 'DB2 Information - Unterstützung' zu einem späteren Zeitpunkt auf diesem Computer oder anderen Computern mit Hilfe einer Antwortdatei installieren möchten, wählen Sie **Ihre Einstellungen in einer Antwortdatei speichern** aus. Klicken Sie **Weiter** an.
10. Wählen Sie auf der Seite **Zu installierende Sprachen auswählen** die Sprachen aus, in denen 'DB2 Information - Unterstützung' installiert werden soll. Klicken Sie den Knopf **Weiter** an.
11. Konfigurieren Sie 'DB2 Information - Unterstützung' auf der Seite **Port von DB2 Information - Unterstützung angeben** für eingehende Kommunikation. Klicken Sie **Weiter** an, um mit der Installation fortzufahren.
12. Überprüfen Sie auf der Seite **Kopieren der Dateien starten** noch einmal die von Ihnen ausgewählten Installationseinstellungen. Wenn Sie die Einstellungen ändern möchten, klicken Sie **Zurück** an. Klicken Sie **Installieren** an, um die Dateien von 'DB2 Information - Unterstützung' auf Ihren Computer zu kopieren.

Sie können 'DB2 Information - Unterstützung' auch mit Hilfe einer Antwortdatei installieren.

Die Installationsprotokolldateien `db2setup.his`, `db2setup.log` und `db2setup.err` befinden sich standardmäßig im Verzeichnis `/tmp`.

Die Datei `db2setup.log` erfasst alle Installationsinformationen zu DB2-Produkten, einschließlich Fehlern. Die Datei `db2setup.his` zeichnet alle DB2-Produktinstallationen auf Ihrem Computer auf. DB2 hängt die Datei `db2setup.log` an die Datei `db2setup.his` an. Die Datei `db2setup.err` erfasst die gesamte Fehlerausgabe, die von Java zurückgegeben wird, wie beispielsweise Informationen zu Ausnahmebedingungen und Traps.

Nach Abschluss der Installation ist 'DB2 Information - Unterstützung' je nach UNIX-Betriebssystem in einem der folgenden Verzeichnisse installiert:

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris-Betriebsumgebung: /opt/IBM/db2/V8.1

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 410
- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 412

Zugehörige Tasks:

- „Installieren von DB2 mit Hilfe einer Antwortdatei (UNIX)" im Handbuch *Installation und Konfiguration Ergänzung*
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 421
- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 422
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 419
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 417

Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)

Es gibt drei Möglichkeiten, auf die DB2-Produktdokumentation zuzugreifen: auf der IBM Website, auf einem Intranet-Server oder auf eine auf dem lokalen Computer installierte Version. Standardmäßig greifen DB2-Produkte auf die DB2-Dokumentation auf der IBM Website zu. Wenn Sie jedoch auf die DB2-Dokumentation auf einem Intranet-Server oder auf dem eigenen Computer zugreifen möchten, müssen Sie die DB2-Dokumentation von der *CD 'DB2 Information - Unterstützung'* aus installieren. Mit dem DB2-Installationsassistenten können Sie Ihre Installationseinstellungen definieren und 'DB2 Information - Unterstützung' auf einem Computer installieren, der ein Windows-Betriebssystem verwendet.

Voraussetzungen:

Dieser Abschnitt erläutert die Voraussetzungen für Hardware, Betriebssystem, Software und Kommunikation zum Installieren von 'DB2 Information - Unterstützung' unter Windows.

- **Hardwarevoraussetzungen**

  Sie benötigen einen der folgenden Prozessoren:

  – 32-Bit-Computer: eine Pentium- oder mit Pentium kompatible CPU
- **Betriebssystemvoraussetzungen**

  Sie benötigen eines der folgenden Betriebssysteme:

  – Windows 2000

  – Windows XP

  Anmerkung: 'DB2 Information - Unterstützung' kann unter einem Teil der Windows-Betriebssysteme ausgeführt werden, unter denen DB2-Clients unterstützt werden. Daher wird empfohlen, entweder über die IBM Website auf 'DB2 Information - Unterstützung' zuzugreifen oder 'DB2 Information - Unterstützung' auf einem Intranet-Server zu installieren und dort auf die Dokumentation zuzugreifen.
- **Softwarevoraussetzungen**

  – Unterstützte Browser:

    - Mozilla 1.0 oder höher

    - Internet Explorer Version 5.5 oder 6.0 (Version 6.0 für Windows XP)
- **Kommunikationsvoraussetzungen**

  – TCP/IP

Einschränkungen:

- Sie benötigen einen Benutzereintrag mit Administratorberechtigung, um 'DB2 Information - Unterstützung' zu installieren.

Vorgehensweise:

Um 'DB2 Information - Unterstützung' mit Hilfe des DB2-Installationsassistenten zu installieren, gehen Sie wie folgt vor:

1. Melden Sie sich mit dem für die Installation von 'DB2 Information - Unterstützung' definierten Benutzereintrag am System an.
2. Legen Sie die CD in das Laufwerk ein. Die IBM DB2 Setup-Klickstartleiste wird von der Funktion für automatische Ausführung gestartet, sofern diese Funktion aktiviert ist.
3. Der DB2-Installationsassistent ermittelt die Systemsprache und startet das Installationsprogramm für diese Sprache. Wenn Sie das Installationsprogramm nicht in Englisch ausführen möchten oder wenn beim automatischen Starten des Programms ein Fehler aufgetreten ist, können Sie den DB2-Installationsassistenten auch manuell starten.

   Um den DB2-Installationsassistenten manuell zu starten, gehen Sie wie folgt vor:

   a. Klicken Sie **Start** an, und wählen Sie die Option **Ausführen** aus.
   b. Geben Sie im Feld **Öffnen** den folgenden Befehl ein:

      ```
      x:\setup.exe /i zweistellige sprachenkennung
      ```

      Hierbei steht *x:* für das CD-Laufwerk und *zweistellige sprachenkennung* für die Sprache, in der das Installationsprogramm ausgeführt werden soll.
   c. Klicken Sie **OK** an.
4. Die IBM DB2-Klickstartleiste wird geöffnet. Um direkt mit der Installation von 'DB2 Information - Unterstützung' fortzufahren, klicken Sie **Produkt installieren** an. Die Onlinehilfe enthält Informationen, die Sie durch die verbleibenden Schritte der Installation führen. Um die Onlinehilfe aufzurufen, klicken Sie **Hilfe** an. Sie können jederzeit **Abbrechen** anklicken, um die Installation zu beenden.
5. Klicken Sie im Fenster **Wählen Sie das zu installierende Produkt aus** den Knopf **Weiter** an.
6. Klicken Sie **Weiter** im Fenster **Willkommen beim DB2-Installationsassistenten** an. Der DB2-Installationsassistent leitet Sie durch die erforderlichen Schritte zum Installieren des Programms.
7. Um mit der Installation fortfahren zu können, müssen Sie die Lizenzvereinbarung akzeptieren. Wählen Sie auf der Seite **Lizenzvereinbarung** die Option **Bedingungen in der Lizenzvereinbarung anerkennen** aus, und klicken Sie **Weiter** an.
8. Wählen Sie **DB2 Information - Unterstützung auf diesem Computer installieren** auf der Seite **Installationsaktion auswählen** aus. Wenn Sie 'DB2 Information - Unterstützung' zu einem späteren Zeitpunkt auf diesem Computer oder anderen Computern mit Hilfe einer Antwortdatei installieren möchten, wählen Sie **Ihre Einstellungen in einer Antwortdatei speichern** aus. Klicken Sie **Weiter** an.
9. Wählen Sie auf der Seite **Zu installierende Sprachen auswählen** die Sprachen aus, in denen 'DB2 Information - Unterstützung' installiert werden soll. Klicken Sie den Knopf **Weiter** an.

10. Konfigurieren Sie 'DB2 Information - Unterstützung' auf der Seite **Port von DB2 Information - Unterstützung angeben** für eingehende Kommunikation. Klicken Sie **Weiter** an, um mit der Installation fortzufahren.
11. Überprüfen Sie auf der Seite **Kopieren der Dateien starten** noch einmal die von Ihnen ausgewählten Installationseinstellungen. Wenn Sie die Einstellungen ändern möchten, klicken Sie **Zurück** an. Klicken Sie **Installieren** an, um die Dateien von 'DB2 Information - Unterstützung' auf Ihren Computer zu kopieren.

Sie haben die Möglichkeit, 'DB2 Information - Unterstützung' mit Hilfe einer Antwortdatei zu installieren. Sie können auch den Befehl **db2rspgn** verwenden, um eine Antwortdatei auf der Grundlage einer vorhandenen Installation zu generieren.

Die Dateien `db2.log` und `db2wi.log` im Verzeichnis `'Eigene Dateien'\DB2LOG\` enthalten Informationen zu Fehlern, die während der Installation aufgetreten sind. Die Position des Verzeichnisses `'Eigene Dateien'` hängt von den Einstellungen Ihres Computers ab.

Die Datei `db2wi.log` erfasst die neuesten DB2-Installationsinformationen. Die Datei `db2.log` erfasst die Protokollinformationen von DB2-Produktinstallationen.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 410
- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 412

Zugehörige Tasks:

- „Installieren eines DB2-Produkts mit Hilfe einer Antwortdatei (Windows)" im Handbuch *Installation und Konfiguration Ergänzung*
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 421
- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 422
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 419
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 414

Zugehörige Referenzen:

- „db2rspgn - Response File Generator Command (Windows)" im Handbuch *Command Reference*

Aufrufen von 'DB2 Information - Unterstützung'

'DB2 Information - Unterstützung' bietet Ihnen die Möglichkeit, auf alle Informationen zuzugreifen, die Sie zur Verwendung der DB2-Produkte für die Betriebssysteme Linux, UNIX und Windows, wie z. B. DB2 Universal Database, DB2 Connect, DB2 Information Integrator und DB2 Query Patroller, benötigen.

Rufen Sie 'DB2 Information - Unterstützung' auf eine der folgenden Arten auf:

- Von einem Computer aus, auf dem ein DB2 UDB-Client oder -Server installiert ist

- Von einem Intranet-Server oder einem lokalen Computer aus, auf dem 'DB2 Information - Unterstützung' installiert ist
- Über die IBM Website

Voraussetzungen:

Führen Sie vor dem Aufrufen von 'DB2 Information - Unterstützung' folgende Schritte aus:

- *Optional*: Konfigurieren des Browsers für die Anzeige der Themen in der gewünschten Landessprache
- *Optional*: Konfigurieren des DB2-Clients für die Verwendung der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'

Vorgehensweise:

Gehen Sie wie folgt vor, um 'DB2 Information - Unterstützung' auf einem Computer aufzurufen, auf dem ein DB2 UDB-Client oder -Server installiert ist:

- Wählen Sie (unter Windows) **Start → Programme → IBM DB2 → Information → DB2 Information - Unterstützung** aus.
- Geben Sie in der Befehlszeile Folgendes ein:
  - Unter Linux und UNIX: Geben Sie den Befehl **db2icdocs** ein.
  - Unter Windows: Geben Sie den Befehl **db2icdocs.exe** ein.

Gehen Sie wie folgt vor, um die auf einem Intranet-Server oder lokalen Computer installierte Komponente 'DB2 Information - Unterstützung' in einem Webbrowser zu öffnen:

- Öffnen Sie die Webseite unter http://<hostname>:<portnummer>/. Dabei stellt <hostname> den Namen des Hosts dar und <portnummer> die Nummer des Ports, an dem 'DB2 Information - Unterstützung' verfügbar ist.

Gehen Sie wie folgt vor, um 'DB2 Information - Unterstützung' auf der IBM Website in einem Webbrowser zu öffnen:

- Öffnen Sie die Webseite unter publib.boulder.ibm.com/infocenter/db2help/.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 410

Zugehörige Tasks:

- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 422
- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 429
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 421
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 430
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 431
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 431

Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'

Die Komponente 'DB2 Information - Unterstützung', auf die Sie über http://publib.boulder.ibm.com/infocenter/db2help/ zugreifen können, wird in regelmäßigen Abständen durch neue oder geänderte Dokumentationen aktualisiert. IBM stellt in bestimmten Fällen auch Aktualisierungen von 'DB2 Information - Unterstützung' zum Download bereit, die Sie auf Ihrem Computer oder Intranet-Server installieren können. Durch die Aktualisierung von 'DB2 Information - Unterstützung' werden keine DB2-Client- oder -Serverprodukte aktualisiert.

Voraussetzungen:

Sie benötigen Zugriff auf einen Computer, der über eine Verbindung zum Internet verfügt.

Vorgehensweise:

Gehen Sie wie folgt vor, um die auf Ihrem Computer bzw. Intranet-Server installierte Komponente 'DB2 Information - Unterstützung' zu aktualisieren:

1. Öffnen Sie 'DB2 Information - Unterstützung' auf der IBM Website unter http://publib.boulder.ibm.com/infocenter/db2help/.
2. Klicken Sie im Downloadbereich der Eingangsseite den Link **DB2 Universal Database-Dokumentation** unter der Überschrift für Service und Unterstützung an.
3. Stellen Sie fest, ob die Version der installierten Komponente 'DB2 Information - Unterstützung' veraltet ist, indem Sie die Stufe des neuesten aktualisierten Dokumentationsimage mit der installierten Dokumentationsstufe vergleichen. Die installierte Dokumentationsstufe ist auf der Eingangsseite von 'DB2 Information - Unterstützung' aufgeführt.
4. Wenn eine neuere Version von 'DB2 Information - Unterstützung' verfügbar ist, laden Sie das neueste aktualisierte Image für *DB2 Information - Unterstützung* für das von Ihnen verwendete Betriebssystem herunter.
5. Befolgen Sie zur Installation des aktualisierten Image für *DB2 Information - Unterstützung* die Anweisungen auf der Webseite.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 412

Zugehörige Tasks:

- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 419
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 414
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 417

Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'

In 'DB2 Information - Unterstützung werden Themen, wenn möglich, in der Sprache angezeigt, die in den Vorgaben Ihres Browsers angegeben ist. Falls ein Thema nicht in die gewünschte Sprache übersetzt wurde, wird es in 'DB2 Information - Unterstützung' in Englisch angezeigt.

Vorgehensweise:

Um Themen in der gewünschten Sprache im Browser 'Internet Explorer' anzuzeigen, gehen Sie wie folgt vor:

1. Klicken Sie im Internet Explorer **Extras** —> **Internetoptionen...** —> **Sprachen...** an. Das Fenster **Spracheinstellung** wird geöffnet.
2. Stellen Sie sicher, dass die gewünschte Sprache als erster Eintrag in der Liste angegeben ist.
   - Klicken Sie den Knopf **Hinzufügen...** an, um eine neue Sprache zur Liste hinzuzufügen.

     Anmerkung: Das Hinzufügen einer Sprache bedeutet nicht zwangsläufig, dass der Computer über die erforderlichen Schriftarten verfügt, um die Themen in der gewünschten Sprache anzuzeigen.
   - Um eine Sprache an den Anfang der Liste zu verschieben, wählen Sie zunächst die gewünschte Sprache und anschließend den Knopf **Nach oben** aus, bis die Sprache an erster Stelle in der Liste steht.
3. Aktualisieren Sie die Seite, um 'DB2 Information - Unterstützung' in der gewünschten Sprache anzuzeigen.

Um Themen in der gewünschten Sprache im Browser 'Mozilla' anzuzeigen, gehen Sie wie folgt vor:

1. Wählen Sie in Mozilla **Bearbeiten** —> **Einstellungen** —> **Sprachen** aus. Die Anzeige für die Auswahl der Sprache wird im Fenster mit den Einstellungen aufgerufen.
2. Stellen Sie sicher, dass die gewünschte Sprache als erster Eintrag in der Liste angegeben ist.
   - Wenn Sie eine neue Sprache hinzufügen möchten, klicken Sie den Knopf **Hinzufügen...** an, um eine Sprache im entsprechenden Fenster auszuwählen.
   - Um eine Sprache an den Anfang der Liste zu verschieben, wählen Sie zunächst die gewünschte Sprache und anschließend den Knopf **Nach oben** aus, bis die Sprache an erster Stelle in der Liste steht.
3. Aktualisieren Sie die Seite, um 'DB2 Information - Unterstützung' in der gewünschten Sprache anzuzeigen.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 410

DB2-Dokumentation in PDF-Format und gedrucktem Format

In den folgenden Tabellen sind die offiziellen Buchtitel, Formularnummern und PDF-Dateinamen aufgeführt. Zum Bestellen von Hardcopybüchern benötigen Sie den offiziellen Buchtitel. Zum Drucken der PDF-Version benötigen Sie den PDF-Dateinamen.

Die DB2-Dokumentation ist in die folgenden Kategorien unterteilt:

- DB2-Kerninformationen
- Verwaltungsinformationen
- Informationen zur Anwendungsentwicklung
- Informationsmanagement
- Informationen zu DB2 Connect
- Einführungsinformationen
- Lernprogramminformationen
- Informationen zu Zusatzkomponenten
- Release-Informationen

In den folgenden Tabellen wird für die einzelnen Bücher der DB2-Bibliothek beschrieben, welche Informationen zum Bestellen von Hardcopys bzw. zum Drucken oder Anzeigen der PDF-Versionen erforderlich sind. Eine vollständige Beschreibung der in der DB2-Bibliothek verfügbaren Bücher finden Sie im IBM Publications Center unter folgender Adresse: www.ibm.com/shop/publications/order.

DB2-Kerninformationen

Diese Bücher enthalten grundlegende Informationen für alle DB2-Benutzer. Diese Informationen sind sowohl für Programmierer als auch für Datenbankadministratoren geeignet und unterstützen Sie bei der Arbeit mit DB2 Connect, DB2 Warehouse Manager und anderen DB2-Produkten.

Tabelle 20. DB2-Kerninformationen

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0e81
IBM DB2 Universal Database Glossar	Keine Formnummer	db2t0g81
IBM DB2 Universal Database Fehlernachrichten, Band 1	GC12-3043, nicht als Hardcopy verfügbar	db2m1g81
IBM DB2 Universal Database Fehlernachrichten, Band 2	GC12-3042, nicht als Hardcopy verfügbar	db2m2g81
IBM DB2 Universal Database Neue Funktionen	SC12-3044	db2q0g81

Verwaltungsinformationen

Die Informationen in diesen Büchern umfassen die Themen, die zum effektiven Entwerfen, Implementieren und Verwalten von DB2-Datenbanken, Data Warehouses und Systemen zusammengeschlossener Datenbanken erforderlich sind.

Tabelle 21. Verwaltungsinformationen

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Systemverwaltung: Konzept	SC12-3057	db2d1g81
IBM DB2 Universal Database Systemverwaltung: Implementierung	SC12-3059	db2d2g81

Tabelle 21. Verwaltungsinformationen (Forts.)

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Systemverwaltung: Optimierung	SC12-3058	db2d3g81
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0e81
IBM DB2 Universal Database Dienstprogramme für das Versetzen von Daten Handbuch und Referenz	SC12-3055	db2dmg81
IBM DB2 Universal Database Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz	SC12-3054	db2hag81
IBM DB2 Universal Database Data Warehouse-Zentrale Verwaltung	SC12-3068	db2ddg81
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1e81
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2e81
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0e81

Informationen zur Anwendungsentwicklung

Die Informationen in diesen Büchern sind besonders für Anwendungsentwickler und Programmierer von Interesse, die mit DB2 Universal Database (DB2 UDB) arbeiten. Sie finden hier Informationen zu den unterstützten Programmiersprachen und Compilern sowie die Dokumentation, die für den Zugriff auf DB2 UDB über die verschiedenen unterstützten Programmierschnittstellen, z. B. eingebettetes SQL, ODBC, JDBC, SQLJ und CLI, erforderlich ist. Wenn Sie die Komponente 'DB2 Information - Unterstützung' verwenden, können Sie auch auf HTML-Versionen des Quellcodes für die Beispielprogramme zugreifen.

Tabelle 22. Informationen zur Anwendungsentwicklung

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Application Development Guide: Building and Running Applications	SC09-4825	db2axe81
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1e81
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2e81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1e81

Tabelle 22. Informationen zur Anwendungsentwicklung (Forts.)

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2e81
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2ade81
IBM DB2 XML Extender Verwaltung und Programmierung	SC12-3062	db2sxg81

Informationsmanagement

Die Informationen in diesen Büchern beschreiben den Einsatz von Komponenten, mit denen Sie die Data Warehousing- und Analysefunktionen von DB2 Universal Database erweitern können.

Tabelle 23. Informationsmanagement

Name	IBM Form	PDF-Dateiname
IBM DB2 Warehouse Manager Standard Edition Informationskatalogzentrale Verwaltung	SC12-3070	db2dig81
IBM DB2 Warehouse Manager Standard Edition Installation	GC12-3069	db2idg81
IBM DB2 Warehouse Manager Standard Edition Managing ETI Solution Conversion Programs with DB2 Warehouse Manager	SC18-7727	iwhe1mste80

Informationen zu DB2 Connect

Die Informationen in dieser Kategorie beschreiben den Zugriff auf Daten auf großen und mittleren Serversystemen mit Hilfe von DB2 Connect Enterprise Edition oder DB2 Connect Personal Edition.

Tabelle 24. Informationen zu DB2 Connect

Name	IBM Form	PDF-Dateiname
IBM Konnektivität Ergänzung	Keine Formnummer	db2h1g81
IBM DB2 Connect Enterprise Edition Einstieg	GC12-3051	db2c6g81
IBM DB2 Connect Personal Edition Einstieg	GC12-3049	db2c1g81
IBM DB2 Connect Benutzerhandbuch	SC12-3048	db2c0g81

Einführungsinformationen

Die Informationen in dieser Kategorie unterstützen Sie beim Installieren und Konfigurieren von Servern, Clients und anderen DB2-Produkten.

Tabelle 25. Einführungsinformationen

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database für DB2-Clients Einstieg	GC12-3052, nicht als Hardcopy verfügbar	db2itg81
IBM DB2 Universal Database für DB2-Server Einstieg	GC12-3047	db2isg81
IBM DB2 Universal Database Personal Edition Einstieg	GC12-3045	db2i1g81
IBM DB2 Universal Database Installation und Konfiguration Ergänzung	GC12-3046, nicht als Hardcopy verfügbar	db2iyg81
IBM DB2 Universal Database Data Links Manager Einstieg	GC12-3056	db2z6g81

Lernprogramminformationen

In den Lernprogramminformationen werden DB2-Funktionen vorgestellt. Darüber hinaus wird die Ausführung verschiedener Tasks beschrieben.

Tabelle 26. Lernprogramminformationen

Name	IBM Form	PDF-Dateiname
Lernprogramm für das Informationsmanagement: Data Warehouse - Einführung	Keine Formnummer	db2tug81
Lernprogramm für das Informationsmanagement: Data Warehouse - Weiterführende Informationen	Keine Formnummer	db2tag81
Lernprogramm für die Informationskatalogzentrale	Keine Formnummer	db2aig81
Video Central für e-business Lernprogramm	Keine Formnummer	db2twg81
Lernprogramm für Visual Explain	Keine Formnummer	db2tvg81

Informationen zu Zusatzkomponenten

Die Informationen in dieser Kategorie beschreiben das Arbeiten mit den DB2-Zusatzkomponenten.

Tabelle 27. Informationen zu Zusatzkomponenten

Name	IBM Form	PDF-Dateiname
IBM DB2 Cube Views Handbuch und Referenz	n/v	db2aag81
IBM DB2 Query Patroller-Handbuch: Installation, Verwaltung und Verwendung	GC12-3225	db2dwg81
IBM DB2 Spatial Extender und Geodetic Extender Benutzer- und Referenzhandbuch	SC12-3063	db2sbg81

Tabelle 27. Informationen zu Zusatzkomponenten (Forts.)

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0e82
DB2 Net Search Extender Verwaltung und Benutzerhandbuch **Anmerkung:** Die HTML-Version dieses Dokuments wird *nicht* von der HTML-Dokumentations-CD installiert.	SH12-3021	n/v

Release-Informationen

Die Release-Informationen enthalten zusätzliche Informationen für das verwendete Produktrelease und die verwendete FixPak-Stufe. Die Release-Informationen enthalten außerdem Zusammenfassungen der Dokumentationsaktualisierungen in den verschiedenen Releases, Aktualisierungen und FixPaks.

Tabelle 28. Release-Informationen

Name	IBM Form	PDF-Dateiname
DB2 Release-Informationen	Siehe Anmerkung.	Siehe Anmerkung.
DB2 Installationsinformationen	Nur auf der Produkt-CD-ROM verfügbar.	n/v

Anmerkung: Die Release-Informationen stehen in den folgenden Formaten zur Verfügung:

- XHTML und Textformat auf den Produkt-CDs
- PDF-Format auf der CD mit der PDF-Dokumentation

Darüber hinaus sind die Abschnitte zu *bekannten Problemen und Fehlerumgehungen* sowie zur *Inkompatibilität zwischen einzelnen Releases*, die Teil der Release-Informationen sind, auch über 'DB2 Information - Unterstützung' verfügbar.

Informationen zum Anzeigen der Release-Informationen in Textformat auf UNIX-Plattformen finden Sie in der Datei `Release.Notes`. Diese Datei befindet sich im Verzeichnis `DB2DIR/Readme/%L`. Hierbei steht `%L` für die länderspezifische Angabe und `DB2DIR` für eine der folgenden Angaben:

- Für AIX-Betriebssysteme: `/usr/opt/db2_08_01`
- Für alle anderen UNIX-Betriebssysteme: `/opt/IBM/db2/V8.1`

Zugehörige Konzepte:

- „DB2-Dokumentation und Hilfe" auf Seite 409

Zugehörige Tasks:

- „Drucken von DB2-Büchern mit PDF-Dateien" auf Seite 428
- „Bestellen gedruckter DB2-Bücher" auf Seite 428
- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 429

Drucken von DB2-Büchern mit PDF-Dateien

DB2-Bücher können mit Hilfe der PDF-Dateien auf der CD mit der *DB2-PDF-Dokumentation* gedruckt werden. Mit Adobe Acrobat Reader können Sie entweder das gesamte Handbuch oder bestimmte Seitenbereiche des Handbuchs ausdrucken.

Voraussetzungen:

Stellen Sie sicher, dass Adobe Acrobat Reader installiert ist. Falls Sie Adobe Acrobat Reader noch nicht installiert haben, finden Sie das Produkt auf der Adobe-Website unter folgender Adresse: www.adobe.com

Vorgehensweise:

Gehen Sie wie folgt vor, um ein DB2-Buch mit einer PDF-Datei auszudrucken:

1. Legen Sie die CD mit der *DB2-PDF-Dokumentation* in das CD-ROM-Laufwerk ein. Hängen Sie unter UNIX-Betriebssystemen die CD mit der DB2-PDF-Dokumentation an. Informationen zum Anhängen einer CD unter UNIX-Betriebssystemen finden Sie im Handbuch *Einstieg* für das jeweilige Betriebssystem.
2. Öffnen Sie index.htm. Die Datei wird in einem Browserfenster geöffnet.
3. Klicken Sie den Titel der PDF an, die Sie aufrufen möchten. Die PDF wird in Acrobat Reader geöffnet.
4. Wählen Sie **Datei → Drucken** aus, um einen beliebigen Teil des gewünschten Buches zu drucken.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 410

Zugehörige Tasks:

- „Anhängen der CD-ROM (AIX)" im Handbuch *DB2 Universal Database für DB2-Server Einstieg*
- „Anhängen der CD-ROM (HP-UX)" im Handbuch *DB2 Universal Database für DB2-Server Einstieg*
- „Anhängen der CD-ROM (Linux)" im Handbuch *DB2 Universal Database für DB2-Server Einstieg*
- „Bestellen gedruckter DB2-Bücher" auf Seite 428
- „Anhängen der CD-ROM (Solaris-Betriebsumgebung)" im Handbuch *DB2 Universal Database für DB2-Server Einstieg*

Zugehörige Referenzen:

- „DB2-Dokumentation in PDF-Format und gedrucktem Format" auf Seite 422

Bestellen gedruckter DB2-Bücher

Wenn Sie die Hardcopyversion der Bücher bevorzugen, können Sie sie auf eine der nachfolgend aufgeführten Arten bestellen.

Vorgehensweise:

In bestimmten Ländern oder Regionen können gedruckte Bücher bestellt werden. Auf der Website mit IBM Veröffentlichungen für das jeweilige Land bzw. die jeweilige Region finden Sie Informationen darüber, ob dieser Service im betreffenden

Land bzw. in der betreffenden Region angeboten wird. Wenn die Veröffentlichungen bestellt werden können, haben Sie folgende Möglichkeiten:

- Wenden Sie sich an den zuständigen IBM Vertragshändler oder Vertriebsbeauftragten. Informationen zum lokalen IBM Ansprechpartner finden Sie im globalen IBM Verzeichnis für Kontakte unter folgender Adresse: www.ibm.com/planetwide.
- Weitere Informationen enthält das IBM Publications Center unter http://www.ibm.com/shop/publications/order. Die Möglichkeit, Bücher über das IBM Publications Center zu bestellen, besteht möglicherweise nicht in allen Ländern.

Die gedruckten Bücher sind zu dem Zeitpunkt, an dem das DB2-Produkt verfügbar gemacht wird, identisch mit den PDF-Versionen auf der CD mit der *DB2-PDF-Dokumentation*. Darüber hinaus stimmt der Inhalt der gedruckten Bücher mit den entsprechenden Informationen auf der CD für *DB2 Information - Unterstützung* überein. Diese CD enthält jedoch zusätzliche Informationen, die in den PDF-Büchern nicht enthalten sind (wie beispielsweise SQL-Verwaltungsroutinen und HTML-Beispiele). Nicht alle Bücher, die auf der CD mit der DB2-PDF-Dokumentation verfügbar sind, können als Hardcopy bestellt werden.

Anmerkung: 'DB2 Information - Unterstützung' wird häufiger aktualisiert als die PDF- oder die Hardcopyversion der Bücher. Installieren Sie die Dokumentationsupdates, sobald diese verfügbar sind, oder greifen Sie über 'DB2 Information - Unterstützung' unter http://publib.boulder.ibm.com/infocenter/db2help/ auf die neuesten Informationen zu.

Zugehörige Tasks:

- „Drucken von DB2-Büchern mit PDF-Dateien" auf Seite 428

Zugehörige Referenzen:

- „DB2-Dokumentation in PDF-Format und gedrucktem Format" auf Seite 422

Aufrufen der Kontexthilfe über ein DB2-Tool

Die Kontexthilfe bietet Informationen zu den Tasks bzw. Steuerelementen, die einem bestimmten Fenster, Notizbuch, Assistenten oder Advisor zugeordnet sind. Die Kontexthilfe steht in allen DB2-Verwaltungs- und -entwicklungstools zur Verfügung, die über eine grafische Benutzerschnittstelle verfügen. Zwei Arten der Kontexthilfe stehen zur Verfügung:

- Die über den Knopf **Hilfe** aufgerufenen Hilfetexte, der in jedem Fenster bzw. Notizbuch zur Verfügung steht.
- Die Kurzhilfe. Hierbei handelt es sich um Informationsfenster, die angezeigt werden, wenn sich der Mauszeiger auf einem Feld oder Steuerelement befindet oder wenn bei der Auswahl eines Feldes oder Steuerelements in einem Fenster, Notizbuch, Assistenten oder Advisor die Taste F1 gedrückt wird.

Über den Knopf **Hilfe** können Sie auf Übersichtsinformationen, Informationen zu Voraussetzungen sowie Informationen zu Tasks zugreifen. In der Kurzhilfe werden die einzelnen Felder und Steuerelemente beschrieben.

Vorgehensweise:

Gehen Sie wie folgt vor, um Kontexthilfe aufzurufen:

- Hilfe zu Fenstern und Notizbüchern können Sie anzeigen, indem Sie eines der DB2-Tools aufrufen und anschließend ein beliebiges Fenster oder Notizbuch öffnen. Klicken Sie den Knopf **Hilfe** in der rechten unteren Ecke des Fensters bzw. Notizbuchs an, um die Kontexthilfe aufzurufen.

  Zugriff auf die Kontexthilfe besteht darüber hinaus über den Menüpunkt **Hilfe** am oberen Rand jeder Zentrale der DB2-Tools.

  Innerhalb von Assistenten und Advisorfunktionen klicken Sie den Link für die Taskübersicht auf der ersten Seite an, um die Kontexthilfe aufzurufen.
- Kurzhilfe zu einzelnen Steuerelementen eines Fensters oder Notizbuchs können Sie aufrufen, indem Sie das gewünschte Steuerelement anklicken und anschließend **F1** drücken. Die Kurzhilfeinformationen mit Details zum jeweiligen Steuerelement werden in einem gelben Fenster angezeigt.

  Anmerkung: Wenn die Kurzhilfe angezeigt werden soll, sobald sich der Mauszeiger auf einem Feld oder Steuerelement befindet, wählen Sie das Markierungsfeld **Kurzhilfe automatisch anzeigen** auf der Seite **Dokumentation** des Notizbuchs 'Tools - Einstellungen' aus.

  Ähnlich wie die Kurzhilfe sind auch Dialogfenster mit Diagnoseinformationen eine Form der kontextbezogenen Hilfe; sie enthalten Regeln für die Dateneingabe. Diese Diagnoseinformationen werden in einem violetten Fenster angezeigt, das aufgerufen wird, wenn die eingegebenen Daten nicht gültig oder nicht ausreichend sind. Die Kontexthilfe mit Diagnoseinformationen kann für folgende Felder angezeigt werden:
  - Musseingabefelder
  - Felder, in denen die Daten einem bestimmten Format entsprechen müssen, wie z. B. Datumsfelder

Zugehörige Tasks:

- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 419
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 430
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 431
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 431
- „Verwenden der DB2 UDB-Hilfe: Gemeinsame GUI - Hilfe"

Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor

Die Hilfe für Nachrichten beschreibt die Ursache von Nachrichten und die Aktionen, die der Benutzer zur Behebung des aufgetretenen Fehlers ausführen sollte.

Vorgehensweise:

Zum Aufrufen der Hilfe für Nachrichten müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? XXXnnnnn
```

Dabei ist *XXXnnnnn* eine gültige Nachrichtenkennung.

So kann beispielsweise durch die Eingabe von `? SQL30081` die Hilfe zur Nachricht SQL30081 angezeigt werden.

Zugehörige Konzepte:

- „Nachrichten - Einführung" im Handbuch *Fehlernachrichten Band 1*

Zugehörige Referenzen:

- „db2 - Command Line Processor Invocation Command" im Handbuch *Command Reference*

Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor

Die Hilfe für Befehle erläutert die Syntax von Befehlen im Befehlszeilenprozessor.

Vorgehensweise:

Zum Aufrufen der Hilfe für Befehle müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? command
```

Dabei stellt *command* ein Schlüsselwort bzw. den vollständigen Befehl dar.

So kann beispielsweise durch die Eingabe von `? catalog` Hilfe für alle CATALOG-Befehle angezeigt werden, während mit `? catalog database` nur Hilfe für den Befehl CATALOG DATABASE angezeigt wird.

Zugehörige Tasks:

- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 429
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 419
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 430
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 431

Zugehörige Referenzen:

- „db2 - Command Line Processor Invocation Command" im Handbuch *Command Reference*

Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor

DB2 Universal Database gibt für Bedingungen, die auf Grund 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 Aufrufen der Hilfe für SQL-Statuswerte müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? sqlstate oder ? klassencode
```

Hierbei steht *sqlstate* für einen gültigen fünfstelligen SQL-Statuswert und *klassencode* für die ersten beiden Ziffern dieses Statuswertes.

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.

Zugehörige Tasks:

- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 419
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 430
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 431

DB2-Lernprogramme

Die Lernprogramme von DB2® unterstützen Sie bei der Einarbeitung in die verschiedenen Themenbereiche von DB2 Universal Database. Sie umfassen Übungen mit in einzelne Arbeitsschritte untergliederten Anweisungen zum Entwickeln von Anwendungen, Optimieren der SQL-Abfrageleistung, Arbeiten mit Data Warehouses, Verwalten von Metadaten und Entwickeln von Webservices mit Hilfe von DB2.

Vorbereitungen:

Die XHTML-Version der Lernprogramme kann über 'DB2 Information - Unterstützung' unter http://publib.boulder.ibm.com/infocenter/db2help/ angezeigt werden.

In einigen der Lernprogrammübungen werden Beispieldaten und Codebeispiele verwendet. Informationen zu den spezifischen Voraussetzungen zur Ausführung der Tasks finden Sie in der Beschreibung des jeweiligen Lernprogramms.

Lernprogramme von DB2 Universal Database:

Klicken Sie einen der Lernprogrammtitel in der folgenden Liste an, um das entsprechende Lernprogramm aufzurufen.

Lernprogramm für das Informationsmanagement: Data Warehouse - Einführung
: Ausführung grundlegender Data Warehousing-Tasks mit Hilfe der Data Warehouse-Zentrale.

Lernprogramm für das Informationsmanagement: Data Warehouse - Weiterführende Informationen
: Ausführung weiterführender Data Warehousing-Tasks mit Hilfe der Data Warehouse-Zentrale.

Lernprogramm für die Informationskatalogzentrale
: Erstellen und Verwalten eines Informationskatalogs zum Lokalisieren und Verwenden von Metadaten mit Hilfe der Informationskatalogzentrale.

Lernprogramm für Visual Explain
: Analysieren, Optimieren und Anpassen von SQL-Anweisungen zur Leistungsverbesserung mit Hilfe von Visual Explain.

Informationen zur Fehlerbehebung in DB2

Eine breite Palette verschiedener Informationen zur Fehlerbestimmung und Fehlerbehebung steht zur Verfügung, um Sie bei der Verwendung von DB2®-Produkten zu unterstützen.

DB2-Dokumentation

Informationen zur Fehlerbehebung stehen in der gesamten Komponente 'DB2 Information - Unterstützung' sowie in den PDF-Büchern der DB2-Bibliothek zur Verfügung. Folgen Sie der Verzweigung 'Unterstützung und Fehlerbehebung' in der Navigationsbaumstruktur von 'DB2 Information - Unterstützung' (im linken Teilfenster des Browserfensters), um eine umfassende Liste der DB2-Dokumentationen zur Fehlerbehebung aufzurufen.

DB2-Website mit technischer Unterstützung

Auf der DB2-Website mit technischer Unterstützung finden Sie Informationen zu Problemen und den möglichen Ursachen und Fehlerbehebungsmaßnahmen. Die DB2-Website mit technischer Unterstützung stellt Links zu den neuesten DB2-Veröffentlichungen, technischen Hinweisen (TechNotes), APARs (Authorized Program Analysis Reports), FixPaks, den neuesten Listen mit internen DB2-Fehlercodes sowie weiteren Ressourcen zur Verfügung. Sie können diese Wissensbasis nach möglichen Lösungen für aufgetretene Probleme durchsuchen.

Rufen Sie die DB2-Website mit technischer Unterstützung unter http://www.ibm.com/software/data/db2/udb/winos2unix/support auf.

DB2-Lernprogramme zur Fehlerbestimmung

Auf der Website mit den DB2-Lernprogrammen zur Fehlerbestimmung finden Sie Informationen dazu, wie Sie Fehler, die bei der Verwendung von DB2-Produkten möglicherweise auftreten, rasch identifizieren und beheben können. Eines der Lernprogramme bietet eine Einführung in die verfügbaren DB2-Einrichtungen und -Tools zur Fehlerbestimmung sowie Entscheidungshilfen für deren Verwendung. Andere Lernprogramme befassen sich mit zugehörigen Themen, wie beispielsweise der Fehlerbestimmung für die Datenbanksteuerkomponente, der Fehlerbestimmung für die Leistung und der Fehlerbestimmung für Anwendungen.

Die vollständige Liste der DB2-Lernprogramme zur Fehlerbestimmung finden Sie auf der DB2-Website mit technischer Unterstützung unter http://www.ibm.com/software/data/support/pdm/db2tutorials.html.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 410
- „Einführung in die Fehlerbestimmung - Lernprogramm für die technische Unterstützung in DB2" im Handbuch *Fehlerbehebung*

Eingabehilfen

Eingabehilfen unterstützen Benutzer mit körperlichen Behinderungen, wie z. B. eingeschränkter Bewegungsfähigkeit oder Sehkraft, beim erfolgreichen Einsatz von Softwareprodukten. Im Folgenden sind die wichtigsten Eingabehilfen aufgeführt, die in den Produkten von DB2® Version 8 zur Verfügung stehen:

- Die gesamte DB2-Funktionalität kann sowohl über die Maus als auch über die Tastatur gesteuert werden. Weitere Informationen hierzu finden Sie unter „Tastatureingabe und Navigation" auf Seite 434.
- Sie können die Größe und Farbe der verwendeten Schriftarten in den DB2-Schnittstellen anpassen. Weitere Informationen hierzu finden Sie unter „Eingabehilfen für Bildschirme" auf Seite 434.
- DB2-Produkte unterstützen Anwendungen mit Eingabehilfen, die mit der Java™ Accessibility API arbeiten. Weitere Informationen hierzu finden Sie unter „Kompatibilität mit Unterstützungseinrichtungen" auf Seite 434.

- Die DB2-Dokumentation steht in behindertengerechtem Format zur Verfügung. Weitere Informationen hierzu finden Sie unter „Dokumentation im behindertengerechten Format".

Tastatureingabe und Navigation

Tastatureingabe

Die verfügbaren DB2-Tools können unter ausschließlicher Benutzung der Tastatur verwendet werden. Mit entsprechenden Tasten oder Tastenkombinationen können Operationen ausgeführt werden, die auch über die Maus verfügbar sind. Die Standardtastenkombinationen des Betriebssystems werden für die entsprechenden Standardoperationen des Betriebssystems verwendet.

Weitere Informationen zur Verwendung von Tasten oder Tastenkombinationen für die Ausführung von Operationen finden Sie unter '' 'Direktaufrufe über die Tastatur: Gemeinsame GUI - Hilfe'.

Navigation über die Tastatureingabe

Sie können in den Benutzerschnittstellen der DB2-Tools mit Hilfe von Tasten oder Tastenkombinationen navigieren.

Weitere Informationen zur Navigation in den DB2-Tools mit Hilfe der Tastatureingabe finden Sie unter '' 'Direktaufrufe über die Tastatur: Gemeinsame GUI - Hilfe'.

Tastatureingabebereich

Unter UNIX®-Betriebssystemen ist der Bereich des aktiven Fensters, in dem die Tastatureingabe wirksam ist, hervorgehoben.

Eingabehilfen für Bildschirme

Die DB2-Tools stellen Funktionen bereit, mit denen sehbehinderten Benutzern verbesserten Eingabehilfen zur Verfügung stehen. Diese Eingabehilfen umfassen die Unterstützung individuell anpassbarer Schriftarteigenschaften.

Schriftarteinstellungen

Über das Notizbuch 'Tools - Einstellungen' können Sie die Farbe, Größe und Schriftart des Textes in Menüs und Dialogfenstern auswählen.

Weitere Informationen zur Angabe von Schriftarteinstellungen finden Sie unter '' 'Ändern der Schriftarten für Menüs und Text: Gemeinsame GUI - Hilfe'.

Unabhängigkeit von Farben

Zur Verwendung der Funktionen des vorliegenden Produkts ist es nicht erforderlich, zwischen unterschiedlichen Farben differenzieren zu können.

Kompatibilität mit Unterstützungseinrichtungen

Die Schnittstellen der DB2-Tools unterstützen die Java Accessibility API. Hierdurch wird der Einsatz von Sprachausgabeprogrammen und anderen Unterstützungseinrichtungen für Personen mit Behinderungen mit den DB2-Produkten ermöglicht.

Dokumentation im behindertengerechten Format

Die Dokumentation für DB2 steht im Format XHTML 1.0 zur Verfügung, das mit den meisten Webbrowsern geöffnet werden kann. XHTML ermöglicht das Aufrufen der Dokumentation mit den Anzeigeeinstellungen, die Sie in Ihrem Browser definiert haben. Darüber hinaus ist der Einsatz von Sprachausgabeprogrammen und anderen Unterstützungseinrichtungen möglich.

Syntaxdiagramme stehen in der Schreibweise mit Trennzeichen zur Verfügung. Dieses Format ist nur dann verfügbar, wenn Sie mit Hilfe eines Sprachausgabeprogramms auf die Onlinedokumentation zugreifen.

Zugehörige Konzepte:

• „Syntaxdiagramme in der Schreibweise mit Trennzeichen" auf Seite 435

Syntaxdiagramme in der Schreibweise mit Trennzeichen

Syntaxdiagramme stehen für Benutzer, die mit Hilfe eines Sprachausgabeprogramms auf 'DB2 Information - Unterstützung' zugreifen, in der Schreibweise mit Trennzeichen zur Verfügung.

In der Schreibweise mit Trennzeichen steht jedes Syntaxelement in einer separaten Zeile. Wenn zwei oder mehr Syntaxelemente stets gemeinsam angegeben (oder nicht angegeben) werden müssen, können sie in derselben Zeile stehen, da sie als ein zusammengesetztes Syntaxelement betrachtet werden können.

Jede Zeile beginnt mit einer Zahl in der Schreibweise mit Trennzeichen, zum Beispiel 3 oder 3.1 oder 3.1.1. Um diese Zahlen korrekt zu hören, müssen Sie sicherstellen, dass das Sprachausgabeprogramm so konfiguriert ist, dass die Interpunktion angesagt wird. Alle Syntaxelemente mit derselben Zahl in der Schreibweise mit Trennzeichen (z. B. alle Syntaxelemente mit der Zahl 3.1) stellen Alternativen dar, die sich gegenseitig ausschließen. Wenn Sie die Zeilen '3.1 USERID' und '3.1 SYSTEMID' hören, wissen Sie, dass die Syntax entweder USERID oder SYSTEMID enthalten kann, nicht jedoch beides.

Die Nummerierung bei der Schreibweise mit Trennzeichen gibt den Grad der Ausgliederung an. Beispiel: Wenn auf das Syntaxelement mit der Zahl 3 in der Schreibweise mit Trennzeichen eine Reihe von Syntaxelementen mit der Zahl 3.1 folgt, sind alle Syntaxelemente mit der Zahl 3.1 dem Syntaxelement mit der Zahl 3 untergeordnet.

Bestimmte Wörter und Symbole werden zusätzlich zu den Zahlen in der Schreibweise mit Trennzeichen verwendet, um weitere Informationen zu den Syntaxelementen anzugeben. In manchen Fällen können diese Wörter und Symbole am Anfang des Elements selbst stehen. Zur einfacheren Identifizierung wird dem Wort oder Symbol ein umgekehrter Schrägstrich (\) vorangestellt, wenn es Teil des Syntaxelements ist. Das Symbol * (Stern) kann zusätzlich zu einer Zahl in der Schreibweise mit Trennzeichen verwendet werden, um anzugeben, dass das Syntaxelement wiederholt wird. Beispiel: Das Syntaxelement *FILE mit der Zahl 3 in der Schreibweise mit Trennzeichen erhält das Format 3 * FILE. Format 3* FILE gibt an, dass das Syntaxelement FILE wiederholt wird. Format 3* * FILE gibt an, dass das Syntaxelement * FILE wiederholt wird.

Zeichen wie beispielsweise Kommas, die bei einer Folge von Syntaxelementen als Trennzeichen verwendet werden, werden in der Syntax unmittelbar vor den Elementen dargestellt, die sie trennen. Diese Zeichen können in derselben Zeile stehen wie das jeweilige Element oder in einer separaten Zeile mit derselben Zahl in der Schreibweise mit Trennzeichen, die auch dem betreffenden Element zugeordnet ist. Die Zeile kann auch ein weiteres Symbol enthalten, das Informationen zu den Syntaxelementen angibt. So bedeuten z. B. die Zeilen 5.1*, 5.1 LASTRUN und 5.1 DELETE, dass, wenn Sie mehr als eines der Elemente LASTRUN und DELETE verwenden, diese Elemente durch Kommas voneinander getrennt werden müssen.

Wenn kein Trennzeichen angegeben wird, verwendet das System zum Trennen der einzelnen Syntaxelemente ein Leerzeichen.

Wenn einem Syntaxelement das Symbol % vorangestellt ist, gibt dies einen Verweis an, der an anderer Stelle definiert ist. Die Zeichenfolge, die auf das Symbol % folgt, ist der Name eines Syntaxfragments und kein Literal. So gibt die Zeile 2.1 %OP1 beispielsweise einen Verweis auf das separate Syntaxfragment OP1 an.

Die nachfolgend aufgeführten Wörter und Symbole werden zusätzlich zu den Zahlen in der Schreibweise mit Trennzeichen verwendet:

- ? stellt ein optionales Syntaxelement dar. Eine Zahl in der Schreibweise mit Trennzeichen, auf die das Symbol ? (Fragezeichen) folgt, gibt an, dass alle Syntaxelemente mit einer entsprechenden Zahl in der Schreibweise mit Trennzeichen sowie alle untergeordneten Syntaxelemente optional sind. Ist nur ein Syntaxelement mit einer Zahl in der Schreibweise mit Trennzeichen vorhanden, wird das Symbol ? in derselben Zeile angezeigt wie das Syntaxelement (zum Beispiel 5? NOTIFY). Sind mehrere Syntaxelemente mit einer Zahl in der Schreibweise mit Trennzeichen vorhanden, wird das Symbol ? in einer separaten Zeile angezeigt, gefolgt von den optionalen Syntaxelementen. Wenn Sie beispielsweise die Zeilen 5 ?, 5 NOTIFY und 5 UPDATE hören, wissen Sie, dass die Syntaxelemente NOTIFY und UPDATE optional sind; das bedeutet, Sie können eines oder keines dieser Elemente auswählen. Das Symbol ? entspricht einer Umgehungslinie in einem Pfeildiagramm.
- ! stellt ein Standardsyntaxelement dar. Eine Zahl in der Schreibweise mit Trennzeichen, auf die das Symbol ! (Ausrufezeichen) und ein Syntaxelement folgen, gibt an, dass es sich bei diesem Syntaxelement um die Standardoption für alle Syntaxelemente handelt, denen dieselbe Zahl in der Schreibweise mit Trennzeichen zugeordnet ist. Nur für eines der Syntaxelemente, denen dieselbe Zahl in der Schreibweise mit Trennzeichen zugeordnet ist, darf das Symbol ! angegeben werden. Wenn Sie beispielsweise die Zeilen 2? FILE, 2.1! (KEEP) und 2.1 (DELETE) hören, wissen Sie, dass (KEEP) die Standardoption für das Schlüsselwort FILE ist. Wenn Sie in diesem Beispiel das Schlüsselwort FILE verwenden, jedoch keine Option angeben, wird die Standardoption KEEP verwendet. Eine Standardoption ist auch für die nächsthöhere Zahl in der Schreibweise mit Trennzeichen gültig. In diesem Beispiel bedeutet das: Wenn das Schlüsselwort FILE weggelassen wird, wird der Standardwert FILE(KEEP) verwendet. Wenn Sie jedoch die Zeilen 2? FILE, 2.1, 2.1.1! (KEEP) und 2.1.1 (DELETE) hören, gilt die Standardoption KEEP nur für die nächsthöhere Zahl in der Schreibweise mit Trennzeichen, 2.1 (der kein Schlüsselwort zugeordnet ist), nicht jedoch für 2? FILE. Wird das Schlüsselwort FILE weggelassen, wird kein Wert verwendet.
- * stellt ein Syntaxelement dar, das keinmal, einmal oder mehrmals wiederholt werden kann. Eine Zahl in der Schreibweise mit Trennzeichen, auf die das Symbol * (Stern) folgt, gibt an, dass dieses Syntaxelement keinmal, einmal oder mehrmals verwendet werden kann, d. h., es ist optional und kann wiederholt werden. Wenn Sie beispielsweise die Zeile 5.1* Datenbereich hören, wissen Sie, dass Sie einen, mehrere oder keinen Datenbereich angeben können. Hören Sie die Zeilen 3*, 3 HOST und 3 STATE, wissen Sie, dass Sie HOST, STATE, beide oder keines der Elemente angeben können.

Anmerkungen:

1. Wenn neben einer Zahl in der Schreibweise mit Trennzeichen ein Stern (*) angezeigt wird und nur ein Element mit dieser Zahl vorhanden ist, können Sie dieses Element mehrmals wiederholen.
2. Wenn neben einer Zahl in der Schreibweise mit Trennzeichen ein Stern angezeigt wird und diese Zahl mehreren Elementen zugeordnet ist, können Sie mehrere Elemente aus der Liste verwenden, jedes davon jedoch nur einmal. Im vorhergehenden Beispiel könnten Sie HOST STATE angeben, nicht jedoch HOST HOST.
3. Das Symbol * entspricht einer zum Ausgangspunkt zurück führenden Linie in einem Pfeildiagramm.

- + stellt ein Syntaxelement dar, das mindestens einmal angegeben werden muss. Eine Zahl in der Schreibweise mit Trennzeichen, auf die das Symbol + (Pluszeichen) folgt, gibt an, dass dieses Syntaxelement mindestens einmal angegeben werden muss und wiederholt werden kann. Wenn Sie beispielsweise die Zeile 6.1+ Datenbereich hören, müssen sie mindestens einen Datenbereich angeben. Wenn Sie die Zeilen 2+, 2 HOST und 2 STATE hören, wissen Sie, dass Sie HOST, STATE oder beides angeben müssen. Wie auch für das Symbol * gilt hier, dass mit dem Pluszeichen ein bestimmtes Element nur dann wiederholt werden kann, wenn es sich um das einzige Element mit dieser Zahl in der Schreibweise mit Trennzeichen handelt. Das Symbol + entspricht wie das Symbol * einer zum Ausgangspunkt zurück führenden Linie in einem Pfeildiagramm.

Zugehörige Konzepte:

- „Eingabehilfen" auf Seite 433

Zugehörige Tasks:

- „Inhaltsverzeichnis"

Zugehörige Referenzen:

- „How to read the syntax diagrams" im Handbuch *SQL Reference, Volume 2*

Common Criteria-Zertifizierung von DB2 Universal Database-Produkten

Für DB2 Universal Database läuft momentan der Bewertungsprozess für die Zertifizierung entsprechend den Richtlinien von Common Criteria Evaluation Assurance Level 4 (EAL4). Weitere Informationen zu Common Criteria finden Sie auf der Common Criteria-Website unter: http://niap.nist.gov/cc-scheme/.

Anhang J. Bemerkungen

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 IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. An Stelle der IBM Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Dienstleistungen in Verbindung mit Fremdprodukten und Fremddienstleistungen liegt beim Kunden, soweit nicht ausdrücklich solche Verbindungen erwähnt sind.

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 Europe, Director of Licensing, 92066 Paris La Defense Cedex, France.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter dienen lediglich als Benutzerinformationen 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 unabhängigen, 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
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
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 Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der Allgemeinen Geschäftsbedingungen der IBM, der Internationalen Nutzungsbedingungen der IBM für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer gesteuerten 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 Garantie, 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 abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Informationen über Produkte anderer Hersteller als IBM wurden von den Herstellern dieser Produkte zur Verfügung gestellt, bzw. aus von ihnen veröffentlichten Ankündigungen oder anderen öffentlich zugänglichen Quellen entnommen. IBM hat diese Produkte nicht getestet und übernimmt im Hinblick auf Produkte anderer Hersteller keine Verantwortung für einwandfreie Funktion, Kompatibilität oder andere Ansprüche. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten der IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele der IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogrammes 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. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle konform sind, für die diese Beispielprogramme geschrieben werden. Die in diesem Handbuch aufgeführten Beispiele sollen lediglich der Veranschaulichung und zu keinem anderen Zweck dienen. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet.

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

Folgende Namen sind in gewissen Ländern Marken der International Business Machines Corporation und wurden in mindestens einem der Dokumente in der DB2 UDB-Dokumentationsbibliothek verwendet:

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
IBM System AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
IBM System /390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

Folgende Namen sind in gewissen Ländern Marken oder eingetragene Marken anderer Unternehmen und wurden in mindestens einem der Dokumente in der DB2 UDB-Dokumentationsbibliothek verwendet.

Microsoft, Windows, Windows NT und das Windows-Logo sind in gewissen Ländern Marken der Microsoft Corporation.

Intel und Pentium sind in gewissen Ländern Marken der Intel Corporation.

Java und alle auf Java basierenden Marken sind in gewissen Ländern Marken von Sun Microsystems, Inc.

UNIX ist in gewissen Ländern eine eingetragene Marke von The Open Group.

Andere Namen von Unternehmen, Produkten oder Dienstleistungen können Marken anderer Unternehmen sein.

Index

E

F

G

H

I

S

T

U

V

W

X

Z

Kontaktaufnahme mit IBM

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3 313233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0190 7 72243 erreichen Sie die DB2 Helpline, wo Sie Antworten zu DB2-spezifischen Problemen erhalten.

Informationen zur nächsten IBM Niederlassung in Ihrem Land oder Ihrer Region finden Sie im IBM Verzeichnis für weltweite Kontakte, das Sie im Web unter http://www.ibm.com/planetwide abrufen können.

Produktinformationen

Informationen zu DB2 Universal Database-Produkten erhalten Sie telefonisch oder im World Wide Web unter http://www.ibm.com/software/data/db2/udb.

Diese Site enthält die neuesten Informationen zur technischen Bibliothek, zum Bestellen von Büchern, zu Produktdownloads, Newsgroups, FixPaks, Neuerungen und Links auf verfügbare Webressourcen.

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3 313233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0180 5 5090 können Sie Handbücher telefonisch bestellen.

Informationen dazu, wie Sie sich mit IBM in Verbindung setzen können, finden Sie auf der globalen IBM Internet-Seite unter folgender Adresse:
www.ibm.com/planetwide

SC12-3054-01