IBM DB2 Universal Database

Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz

Version 7

IBM DB2 Universal Database

Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz

Version 7

Anmerkung:
Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter „Anhang K. Bemerkungen" auf Seite 567 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 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 Univeral Database Data Recovery and High Availability Guide and Reference,
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 2001

Inhaltsverzeichnis

Zu diesem Handbuch

Dieses Buch bietet detaillierte Informationen zu den IBM DB2 Universal Database (UDB)-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, Structured Query Language (SQL) und der Betriebssystemumgebung vertraut sind, in der DB2 UDB ausgeführt wird. Basisinformationen zu DB2 UDB finden Sie im Handbuch *Systemverwaltung*. Informationen zu SQL finden Sie im Handbuch *SQL Reference*. Informationen zur Konfiguration, zum Aufruf und zur Verwendung des DB2 UDB-Befehlszeilenprozessors finden Sie im Handbuch *Command Reference*. Informationen zu den DB2 UDB-Anwendungsprogrammierschnittstellen (APIs) finden Sie im Handbuch *Administrative API Reference*. Basisinformationen zum Erstellen von Anwendungen, die DB2-Verwaltungs-APIs enthalten, finden Sie im Handbuch *Application Building Guide*. Das vorliegende Handbuch enthält keine Anleitungen zur Installation von DB2, da diese vom Betriebs-
system abhängen. Installationsanleitungen finden Sie im Handbuch *Einstieg* (Quick Beginnings) für Ihr Betriebssystem.

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 in den Datenbankwiederherstellungsprotokolldateien aufgezeichneten Transaktionen angewendet werden.

Hohe Verfügbarkeit

Kapitel 5. 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 6. Hohe Verfügbarkeit unter AIX
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.

Kapitel 7. Hohe Verfügbarkeit unter dem Windows-Betriebssystem
Behandelt die DB2-Unterstützung für die Fehlerbehebung durch Funktionsübernahme mit hoher Verfügbarkeit unter Windows NT, die zurzeit durch Microsoft Cluster Server (MSCS) implementiert ist.

Kapitel 8. Hohe Verfügbarkeit in der Solaris-Betriebsumgebung
Behandelt die DB2-Unterstützung für die Fehlerbehebung durch Funktionsübernahme mit hoher Verfügbarkeit in der Solaris-Betriebsumgebung, die zurzeit durch Sun Cluster 2.x (SC2.x), 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. Beispiele für Wiederherstellungsprogramme
Bietet eine Codeliste für Beispielprogramme, die wiederherstellungsbezogene DB2-APIs und eingebettete SQL-Aufrufe enthalten, und Informationen zu ihrer Verwendung.

Anhang F. Befehlszeilen-Script zur Wiederherstellung
Bietet eine Codeliste für ein DB2-Befehls-Script, das wiederherstellungsbezogene CLP-Befehle enthält, und Informationen zu ihrer Verwendung.

Anhang G. Tivoli Storage Manager
Bietet Informationen zum Produkt Tivoli Storage Manager (TSM, bisher ADSM), das Sie zur Verwaltung von Datenbank- oder Tabellenbereichssicherungsoperationen verwenden können.

Anhang H. Benutzer-Exit zur Datenbankwiederherstellung
Behandelt die Verwendung von Benutzer-Exit-Programmen für Datenbankprotokolldateien und beschreibt einige Beispiele für Benutzer-Exit-Programme.

Anhang I. 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

Eine Datenbank kann aufgrund von Hardware- oder Softwarefehlern (oder beidem) unbrauchbar werden. Irgendwann treten möglicherweise Speicherprobleme, Stromausfälle und Anwendungsfehler auf, und unterschiedliche Fehlerszenarios erfordern 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?

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 Befehls-Scripts, 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 Image 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 Sicherungsimage 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.

Weitere Informationen zu diesen verschiedenen Wiederherstellungsmethoden finden Sie in „Versionswiederherstellung" auf Seite 27, „Aktualisierende Wiederherstellung" auf Seite 28 oder „Wiederherstellen nach einem Systemabsturz" auf Seite 11.

Die Protokolldateien für die Wiederherstellung und die Datei des Wiederherstellungsprotokolls werden automatisch erstellt, wenn eine Datenbank erstellt wird (Abb. 1 auf Seite 5). Eine Protokolldatei für die Wiederherstellung oder die Datei des Wiederherstellungsprotokolls kann nicht direkt geändert werden.

Jedoch spielen sie eine wichtige Rolle, wenn mit Hilfe des Sicherungsimage der Datenbank Daten wiederhergestellt werden müssen, die verloren gingen oder beschädigt wurden.

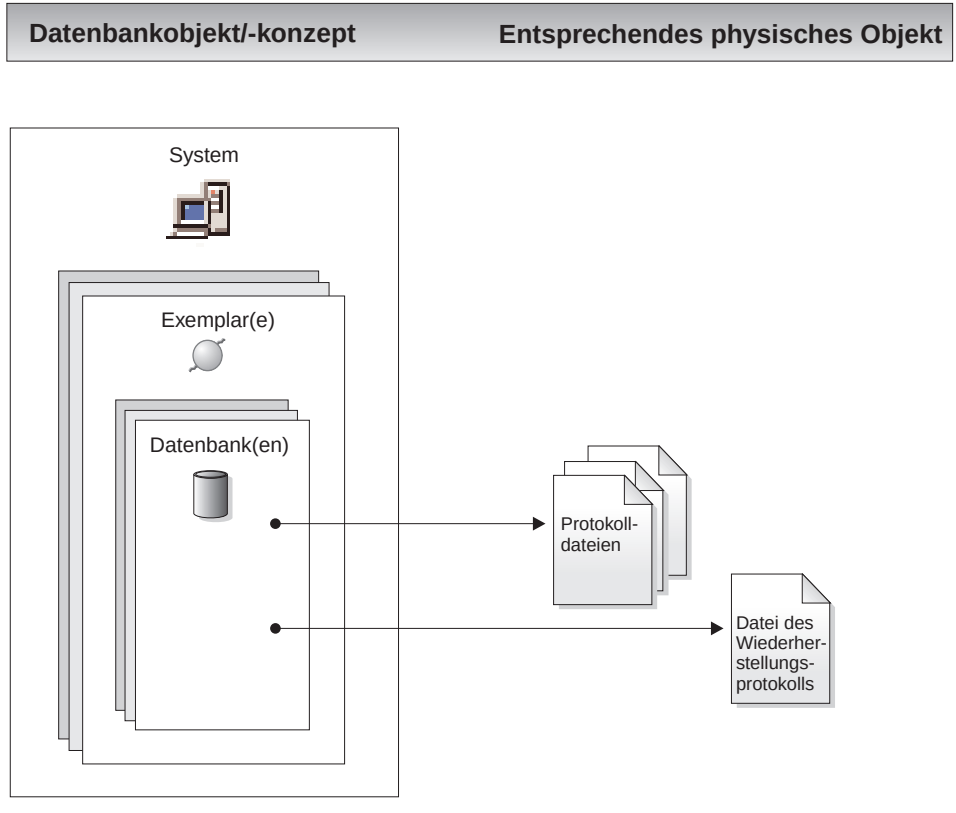


Abbildung 1. Protokolldateien für die Wiederherstellung und die Datei des Wiederherstellungsprotokolls

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 Sicherungsinformationen, die 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.

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 Protokollierungsaufwand nicht die zusätzliche Komplexität der Verwaltung

von Protokolldateien sowie die aktualisierende Wiederherstellung nach einer Wiederherstellungsoperation rechtfertigt. Bei *nicht wiederherstellbaren Datenbanken* sind die beiden Datenbankkonfigurationsparameter *logretain* und *userexit* inaktiviert. 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.

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 der Datenbankkonfigurationsparameter *logretain* auf den Wert „RECOVERY" gesetzt, und/oder der Datenbankkonfigurationsparameter *userexit* ist aktiviert. 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 zur Wiederherstellung und aktualisierenden Wiederherstellung der Datenbank müssen immer 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 aufge-

zeichnet. 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.

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 (siehe „Teilsicherung und Teilwiederherstellung" auf Seite 31).

Sie sollten in regelmäßigen Abständen Gesamtsicherungen der Datenbanken erstellen, selbst wenn Sie die Protokolle archivieren (wodurch eine aktualisierende Wiederherstellung ermöglicht wird). Es ist schwieriger, eine Datenbank aus einer Sammlung von Tabellenbereichssicherungsimages erneut zu erstellen, als sie aus einem Sicherungsimage der gesamten Datenbank wiederherzustellen. Tabellenbereichssicherungsimages sind sinnvoll für eine Wiederherstellung nach dem Ausfall einer einzelnen Platte oder nach einem Anwendungsfehler.

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 berücksichtigen. 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 durchgeführt, 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 verwenden, wenn Sie über die Protokolle verfügen, die den Zeitraum für die Sicherungsoperation umfassen.

Offlinesicherungsoperationen sind schneller als Onlinesicherungsoperationen.

Wenn eine Datenbank große Mengen von Langfeld- und LOB-Daten enthält, kann das Sichern der Datenbank sehr viel Zeit in Anspruch nehmen. Mit dem Sicherungsdienstprogramm können Sie ausgewählte Tabellenbereiche sichern. Wenn Sie DMS-Tabellenbereiche verwenden, können Sie verschiedene Arten von Daten 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 die Langfeld- und LOB-Daten in separaten Tabellenbereichen gespeichert werden, lässt sich der Zeitaufwand für eine Sicherungsoperation dadurch verringern, dass die Tabellenbereiche mit den Langfeld- und LOB-Daten von der Sicherung ausgeschlossen werden. Handelt es sich bei Ihren Langfeld- und LOB-Daten um wichtige Unternehmensdaten, sollten Sie den Zeitaufwand für das Sichern dieser Tabellenbereiche gegen den für die Wiederherstellungsoperation erforderlichen Zeitaufwand abwägen. 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: die Zeit, die zur Wiederherstellung der Sicherungskopie benötigt wird, sowie, wenn für die Datenbank die aktualisierende Wiederherstellung aktiviert ist, die Zeit, die zur Anwendung der Protokolle während der aktualisierenden Wiederherstellung benötigt wird. Bei der

Formulierung eines Wiederherstellungsplans sollten Sie diesen Wiederherstellungsaufwand und die dadurch verursachte 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.

Überlegungen zum Speicherbedarf

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 den betreffenden Spalten nicht protokolliert werden sollen. Dadurch verringert sich die Größe der Protokolldateien und somit der erforderliche Protokollarchivierungsspeicher.

Die Sicherungskopie eines SMS-Tabellenbereichs mit LOB-Daten kann größer werden als der ursprüngliche Tabellenbereich. Die Sicherungskopie kann je nach Größe der LOB-Daten im Tabellenbereich bis zu 40 % größer werden. Wenn Sie beispielsweise eine Sicherung eines 1 GB großen SMS-Tabellenbereichs (mit LOB-Daten) erstellen, benötigen Sie zur Wiederherstellung des Tabellenbereichs mehr als 1 GB Plattenspeicherplatz. Dies tritt nur bei Dateisystemen auf, die die Zuordnung von Dateien mit freien Bereichen (Sparse Allocation) unterstützen (z. B. UNIX-Betriebssysteme).

Überlegungen zum Speicherbedarf

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. (Eine Beschreibung dieses sowie weiterer Konfigurationsparameter, die für das Protokollieren relevant sind, finden Sie in „Konfigurationsparameter für die Datenbankprotokollierung" auf Seite 44.)

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:

- Stellen Sie genügend Speicher im Verzeichnispfad der Datenbankprotokolle bereit, um die Protokolldateien aufzunehmen.
- 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 Benutzer-Exit-Programms auf eine andere Speichereinheit in Ihrer Umgebung. Unter OS/2 beispielsweise unterstützt DB2 ein Benutzer-Exit-Programm zum Speichern der Sicherungsimages und Protokolle von Datenbanken auf Standard- und Nicht-Standardeinheiten. (Weitere Informationen finden Sie in „Anhang H. Benutzer-Exit zur Datenbankwiederherstellung" auf Seite 493.)

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 Mengen von Daten gemeinsam zu sichern. Solche zusammengehörigen Datenmengen können entweder auf Tabellenbereichsebene oder auf Datenbankebene eingerichtet werden. Wenn die zusammengehörigen Datenmengen 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 durchzuführen.

Verwenden verschiedener Betriebssysteme

Wenn Sie in einer Umgebung mit mehreren Betriebssystemen arbeiten, müssen Sie berücksichtigen, dass die Sicherungs- und Wiederherstellungspläne oft nicht integriert werden können. Dies bedeutet, dass Sie nicht einfach eine Datenbank unter einem Betriebssystem sichern und anschließend unter einem anderen Betriebssystem wiederherstellen können. Sie sollten daher die Wiederherstellungspläne für die einzelnen Betriebssysteme getrennt halten und unabhängig voneinander ausführen.

Das plattformübergreifende Sichern und Wiederherstellen zwischen Sun Solaris und HP wird jedoch unterstützt. Wenn Sie das Sicherungsimage von einem System auf das andere übertragen, muss dies im binären Modus erfolgen. Die Datenbank muss auf dem Zielsystem mit derselben Codepage und demselben Gebiet erstellt werden, wie die Originaldatenbank auf dem ursprünglichen System.

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. Weitere Informationen zu diesen Dienstprogrammen finden Sie im Handbuch *Versetzen von Daten Dienstprogramme und Referenz*.

Wiederherstellen 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. Der Prozess der *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 auf Seite 12).

Wenn eine Datenbank sich in einem konsistenten und verwendbaren Status befindet, hat sie einen Punkt erreicht, der als „Konsistenzzustand" bezeichnet wird.

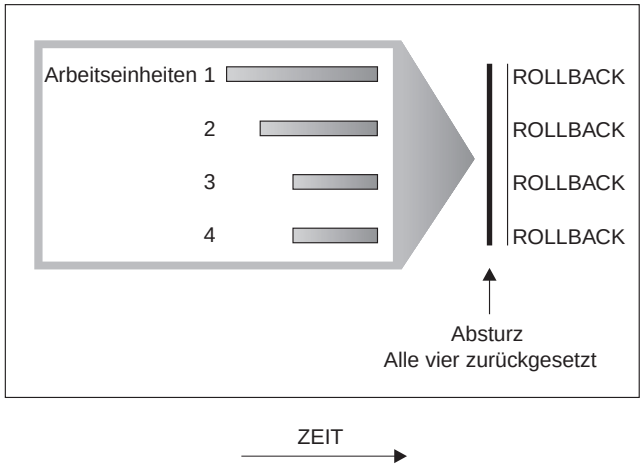


Abbildung 2. Rückgängigmachen von Arbeitseinheiten (Wiederherstellung nach einem Systemabsturz)

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 und unbrauchbar. 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 schwer wiegender Betriebssystemfehler, durch den DB2 abnormal beendet wird

Wenn das Zurücksetzen unvollständiger Arbeitseinheiten automatisch vom Datenbankmanager ausgeführt werden soll, aktivieren Sie den Datenbankkonfigurationsparameter für automatischen Wiederanlauf (*autorestart*), indem Sie ihn auf `ON` setzen. (Dies ist der Standardwert. Weitere Informationen zu diesem Parameter finden Sie im Handbuch *Systemverwaltung: Optimierung*.) Wenn Sie den automatischen Neustart nicht aktivieren, müssen Sie beim Auftreten eines Datenbankfehlers den Befehl RESTART DATABASE absetzen. Der Neustart einer Datenbank wird in der Datei `db2diag.log` aufgezeichnet.

Wenn eine Wiederherstellung nach einem Systemabsturz für eine Datenbank durchgeführt wird, für die die aktualisierende Wiederherstellung aktiviert ist (d. h., der Konfigurationsparameter *logretain* ist auf `RECOVERY` gesetzt, oder der Konfigurationsparameter *userexit* ist aktiviert) und während dieser Wiederherstellung ein Fehler auftritt, der auf einen einzelnen Tabellenbereich zurückzuführen ist, muss dieser Tabellenbereich in den Offlinestatus versetzt werden.

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 sind die anderen Tabellenbereiche in der Datenbank weiterhin verwendbar, und es können Verbindungen zur Datenbank hergestellt werden.

Wiederherstellen beschädigter Tabellenbereiche

Ein beschädigter Tabellenbereich enthält einen oder mehrere Behälter, auf den/die nicht zugegriffen werden kann. Die 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 behoben werden können, ohne dass die ursprünglichen Daten erhalten bleiben, sind nur die folgenden Optionen verfügbar:

- Wiederherstellen der Datenbank
- Wiederherstellen des Katalogtabellenbereichs (Die Wiederherstellung des Tabellenbereichs ist nur für wiederherstellbare Datenbanken zulässig, da die Datenbank aktualisierend wiedergeherstellt werden muss.)

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 Überlegungen 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.

Wiederherstellen von Tabellenbereichen in wiederherstellbaren Datenbanken

Der beschädigte Tabellenbereich wird in den Status *offline*, *nicht verfügbar* und *Aktualisierende Wiederherstellung anstehend* versetzt, da eine Wiederherstellung nach Systemabsturz erforderlich ist. Die Neustartoperation ist erfolgreich, falls kein weiterer Fehler vorliegt. Der beschädigte Tabellenbereich kann wieder verwendet werden, wenn Folgendes geschehen ist:

- Die Fehler an den beschädigten Behältern wurden ohne Verlust der ursprünglichen Daten behoben, und anschließend wurde der Tabellenbereich aktualisierend wiederhergestellt. (Die Operation zur aktualisierenden Wiederherstellung versucht dabei zunächst, den Tabellenbereich wieder vom Offlinestatus in den normalen Status zurückzuversetzen.)
- Nach der Reparatur der beschädigten Behälter (mit oder ohne Verlust der Originaldaten) wurden eine RESTORE-Operation für den Tabellenbereich und anschließend eine Operation zur aktualisierenden Wiederherstellung durchgeführt.

Wiederherstellen von Tabellenbereichen in nicht wiederherstellbaren Datenbanken

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

Dies können Sie erreichen, indem Sie eine Operation RESTART DATABASE ohne Qualifikationsmerkmale durchführen. Wenn keine beschädigten Tabellenbereiche vorhanden sind, ist diese Operation erfolgreich. Wenn sie fehlschlägt (SQL0290N), können Sie die vollständige Liste der Tabellenbereiche, die momentan beschädigt sind, in der Datei `db2diag.log` überprüfen.

- Wenn Sie bereit sind, alle diese Tabellenbereiche zu löschen, sobald die Operation RESTART DATABASE beendet ist, können Sie eine weitere Operation RESTART DATABASE einleiten, wobei Sie alle beschädigten Tabellenbereiche unter der Option DROP PENDING TABLESPACES auflisten. Wenn ein beschädigter Tabellenbereich in der Liste DROP PENDING TABLESPACES enthalten ist, wird der Tabellenbereich in den Status *Löschen anstehend* versetzt, und die einzige Option nach der Wiederherstellung besteht darin, den Tabellenbereich zu löschen. Die Neustartoperation wird ohne Wiederherstellung dieses Tabellenbereichs 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.

- Wenn Sie diese Tabellenbereiche nicht löschen (d. h. die darin enthaltenen Daten nicht verlieren) wollen, haben Sie folgende Möglichkeiten:
  - Sie können abwarten und die beschädigten Behälter reparieren (ohne die Originaldaten zu verlieren), und anschließend die Operation RESTART DATABASE wiederholen.
  - Sie können eine Operation RESTORE DATABASE durchführen.

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. Dies wäre nur der Fall, wenn der Tabellenbereich während des Neustarts beschädigt ist. Nach der erfolgreichen Ausführung der Neustartoperation müssen Sie jeden der Tabellenbereiche, die sich im Status *Löschen anstehend* befinden (mit Hilfe des Befehls LIST TABLESPACES können Sie die Tabellenbereiche in diesem Status feststellen), durch Ausführen von Anweisungen DROP TABLESPACE löschen. Dadurch kann der Speicherbereich wieder verfügbar gemacht oder können die Tabellenbereiche erneut erstellt werden.

Begrenzen der Auswirkungen von Datenträgerfehlern

Zur Verringerung der Wahrscheinlichkeit eines Datenträgerfehlers und zur Vereinfachung der Wiederherstellung der Daten nach einem solchen Fehler sollten Sie folgende Maßnahmen in Betracht ziehen:

- 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 Level 5. Weitere Informationen zu RAID finden Sie in „Schützen vor Plattenfehlern".
- 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 Ver-

wendung einer *Platteneinheit* (Disk Array) an. Eine Platteneinheit besteht aus einem Verbund von Plattenlaufwerken, die einer Anwendung gegenüber wie ein einziges großes Plattenlaufwerk erscheinen.

Solche Platteneinheiten bieten *Striping*-Funktionen, d. h. Funktionen zur Verteilung einer Datei über mehrere Platten, zur Spiegelung von Platten und zur Paritätsprüfung.

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 sämtliche, zur Verwaltung der zu dieser Einheit gehörenden Platten erforderliche Logik im Plattencontroller enthalten ist, ist diese Implementierung vom Betriebssystem unabhängig.

Es gibt verschiedene Arten der RAID-Architektur, die sich in Funktion und Leistung unterscheiden. Die derzeit gängigsten Arten sind jedoch RAID Level 1 und Level 5.

RAID Level 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 Level 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 Level 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 (RAID Level 1) oder eine Softwareplatteneinheit in Betracht ziehen (siehe „Softwareplatteneinheiten"), da diese Möglichkeiten eine Wiederherstellbarkeit bis zu dem Punkt des Ausfalls bieten und eine gute Schreibleistung zeigen, was für Protokolle wichtig ist. 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 ebenso 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 Levels 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 (siehe „Hardwareplatteneinheiten" auf Seite 16), jedoch wird die Verwaltung des Plattenverkehrs entweder durch das Betriebssystem oder durch ein auf dem Server aktives Anwendungsprogramm erledigt. 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 (siehe „Hardwareplatteneinheiten" auf Seite 16). Obwohl redundante

Platten erforderlich sind, ist eine Softwareplatteneinheit relativ preiswert zu implementieren, da kostenintensive Plattencontroller nicht benötigt werden.

Achtung:
Wenn sich das Boot-Laufwerk 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 Boot-Laufwerk 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
- Ausreichender Plattenspeicherplatz für Datenbankprotokolle
- Zuverlässige Kommunikationsverbindungen zwischen den Datenbankpartitionsservern in einer Umgebung mit partitionierten Datenbanken
- Synchronisation der Systemuhren in einer Umgebung mit partitionierten Datenbanken (siehe „Synchronisation der Systemuhren in einem System mit partitionierten Datenbanken" auf Seite 169)

Wiederherstellen nach Transaktionsfehlern in einer partitionierten Datenbankumgebung

Tritt ein Transaktionsfehler in einer Umgebung mit partitionierten Datenbanken auf, ist in der Regel eine Datenbankwiederherstellung sowohl auf dem ausgefallenen 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 erfolgte keine Datenänderung.
YES	Auf diesem Server erfolgte 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.

Weitere Informationen zum zweiphasigen Festschreiben finden Sie im Handbuch *Systemverwaltung: Konzept*.

Wiederherstellung auf einem aktiven Datenbankpartitionsserver nach Transaktionsfehler

Wenn ein Datenbankpartitionsserver feststellt, dass ein anderer Server abgestürzt 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-Fehlernachricht SQL0279N und verliert die Datenbankverbindung. Ansonsten sendet der Koordinatoragent eine ROLLBACK-Anforderung an alle an der Transaktion beteiligten Server, und die Anwendung empfängt die Nachricht SQL1229N.
- Wenn der ausgefallene Datenbankpartitionsserver der Koordinatorknoten für die Anwendung war, werden Agenten, die noch für die Anwendung auf den aktiven Servern arbeiten, unterbrochen, um Wiederherstellungsmaßnahmen durchzuführen. 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).

Weitere Informationen zur Auflösung unbestätigter Transaktionen finden Sie im Handbuch *Systemverwaltung: Konzept*.

- 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 ROLL- BACK-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 des Prozessors erneut zu starten. Falls der Neustart des Prozessors nicht möglich ist, können Sie den Befehl **db2start** auch auf einem anderen Prozessor zum Starten des Datenbankmanagers verwenden. Weitere Informationen finden Sie im Handbuch *Command Reference*.

Die abnormale Beendigung des Datenbankmanagers kann zur Inkonsistenz einiger Datenbankpartitionen auf dem Server führen. Um diese Datenbankpartionen 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 angewandt, um sicherzustellen, dass die Ergebnisse aller vollständigen Transaktionen in der Datenbank vorhanden sind. Nachdem alle Änderungen erneut angewandt wurden, werden alle nicht festgeschriebenen Transaktionen *außer* unbestätigten Transaktionen lokal rückgängig gemacht. In einer Umgebung mit partitionierten Datenbanken gibt es zwei Arten 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 durchgeführte 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 (siehe *Command Reference*) 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 Arten unbestätigter Transaktionen enthält das Feld für die Quelle (*Originator*) in der Ausgabe des Befehls LIST INDOUBT TRANSACTIONS eine der folgenden Angaben:

- DB2 Universal Database Enterprise - Extended 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.

Weitere Informationen zu verteilten Umgebungen finden Sie im Handbuch *Systemverwaltung: Konzept*.

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 die Datei `db2diag.log` 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.

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.

Wiederherstellen von unbestätigten Transaktionen auf dem Host

Wenn Ihre Anwendung während einer Transaktion auf einen Host- oder AS/400-Datenbankserver zugegriffen hat, gibt es einige Unterschiede darin, wie unbestätigte Transaktionen wiederhergestellt werden.

DB2 Connect wird für den Zugriff auf Host- oder AS/400-Datenbankserver verwendet. Die Schritte zur Wiederherstellung unterscheiden sich, wenn bei DB2 Connect der DB2-Synchronisationspunktmanager konfiguriert ist.

Wiederherstellung, wenn bei DB2 Connect der DB2-Synchronisationspunktmanager konfiguriert ist

Die Wiederherstellung unbestätigter Transaktionen auf Host- oder AS/400-Servern wird normalerweise automatisch vom Transaktionsmanager (TM) und dem DB2-Synchronisationspunktmanager (SPM) durchgeführt. Eine unbestätigte Transaktion auf einem Host- oder AS/400-Server belegt keine Ressourcen auf der lokalen DB2-Station, belegt jedoch Ressourcen auf dem Host- oder AS/400-Server, solange die Transaktion auf dem betreffenden Server unbestätigt bleibt. Wenn der Administrator des Host- oder AS/400-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 AS/400-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 AS/400-Server. Die Partner-LU (partner_lu) ist der vollständig qualifizierte LU-Name des Host- oder AS/400-Servers. Dies stellt die beste Identifikation des Host- oder AS/400-Servers zur Verfügung und sollte vom Anrufer vom Standort des Host- oder AS/400-Servers eingeholt werden. Die Angabe luwid ist eine eindeutige Kennung für eine Transaktion und steht auf allen Host- und AS/400-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. Weitere Informationen finden Sie im Handbuch *Command Reference*.

   ```
   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 dann DB2 Connect die Resynchronisation mit dem Host- oder AS/400-Server auf der Basis der heuristischen Entscheidung.

Wiederherstellung, wenn der DB2-Synchronisationspunktmanager bei DB2 Connect nicht verwendet wird

Verwenden Sie die Informationen in diesem Abschnitt, wenn die TCP/IP-Konnektivität verwendet wird, um DB2 für OS/390 in einer Aktualisierung auf mehreren Systemen von DB2 Connect Personal Edition oder DB2 Connect Enterprise Edition aus 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. Weitere Informationen zu Alerts finden Sie im Handbuch *Troubleshooting Guide*.

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 OS/390-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 OS/390 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 finden Sie im Handbuch *DB2 for OS/390 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:

- Ein 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 getrenntes 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. Weitere Informationen finden Sie in „Erneutes Definieren von Tabellenbereichsbehältern während einer Wiederherstellungsoperation (Umgeleitete Wiederherstellung)” auf Seite 130.

Sowohl Tabellenbereichssicherungen als auch vollständige Datenbanksicherungen können in Ihrer Planung zur Wiederherstellung nach einem Katastrophenfall eingesetzt werden. Die zur Sicherung, Wiederherstellung und aktualisierenden Wiederherstellung von Daten verfügbaren DB2-Einrichtungen bieten die Grundlage für einen Plan zur Wiederherstellung nach einem Katastrophenfall. Sie sollten die eingerichteten Wiederherstellungsprozeduren getestet haben, um Ihr Unternehmen zu schützen.

Versionswiederherstellung

Eine *Versionswiederherstellung* ist die Wiederherstellung einer früheren Version der Datenbank mit Hilfe eines Image 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 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).

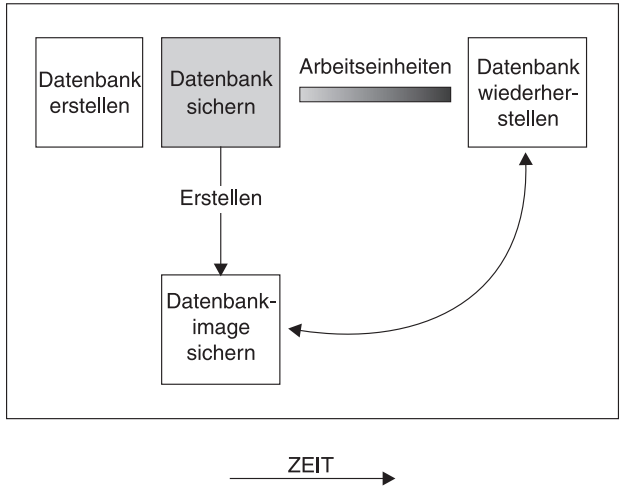


Abbildung 3. Versionswiederherstellung. Die Datenbank wird aus dem zuletzt erstellten Sicherungsimage wiederhergestellt, es gehen jedoch alle Arbeitseinheiten verloren, die vom Zeitpunkt der Sicherung ausgehend bis zum Eintreten des Fehlers ausgeführt wurden.

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 wiederher-

gestellt.) 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 (durch Aktivieren des Datenbankkonfigurationsparameters *logretain* und/oder *userexit*. Informationen zu den Entscheidungen hinsichtlich der zu verwendenden Protokollierungsprozedur finden Sie in „Wiederherstellungsprotokolle" auf Seite 36.) 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 für die Operation RESTORE DATABASE 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.

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

- *Aktualisierende Wiederherstellung der Datenbank*. Bei dieser Art der aktualisierenden Wiederherstellung werden im Anschluss an die Wiederherstellung der Datenbank Transaktionen angewendet, die in den Datenbankprotokollen aufgezeichnet sind (siehe Abb. 4 auf Seite 29). 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. 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.

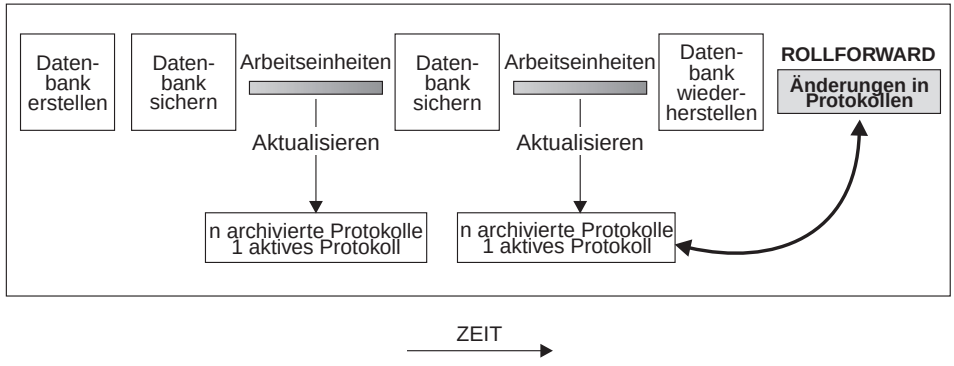


Abbildung 4. Aktualisierende Wiederherstellung einer Datenbank. Bei einer zeitintensiven Transaktion kann mehr als eine Protokolldatei aktiv sein.

- *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 30). 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 (siehe „ROLLFORWARD DATABASE, Befehl" auf Seite 171) auf, um die Protokolle auf die Tabellenbereiche entweder bis zu einem Zeitpunkt oder bis zum Ende der Protokolle 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 Span-

nungsverlust 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 die aktualisierende Wiederherstellung bis zum Ende der Protokolle ausführen.

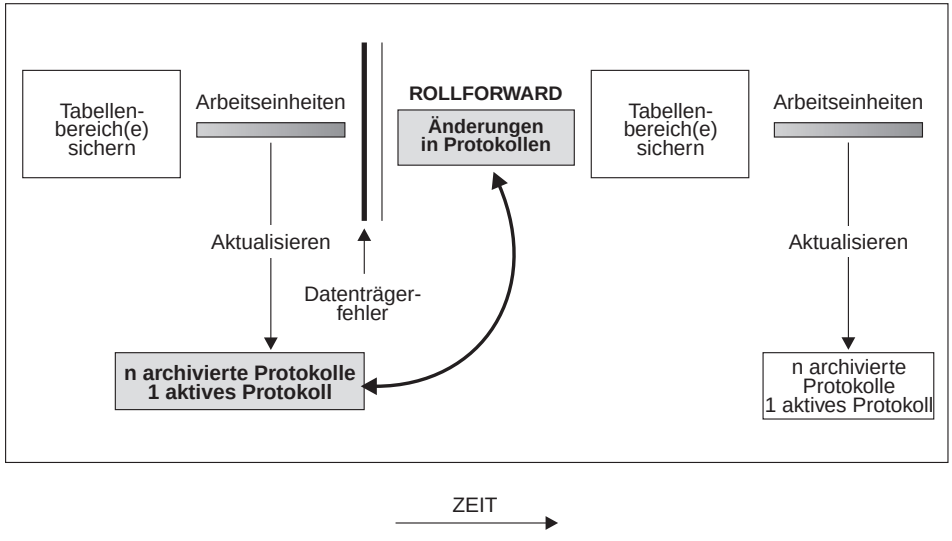


Abbildung 5. Aktualisierende Wiederherstellung von Tabellenbereichen. Bei einer zeitintensiven Transaktion kann mehr als eine Protokolldatei aktiv sein.

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.

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 andern, bei der Sicherung nicht berücksichtigten Tabellenbereichen vorgenommen wurden.

DB2 bietet nun Unterstützung für Teilsicherungen und Teilwiederherstellungen (jedoch nicht für Langfeld- oder LOB-Daten). 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.

Es werden zwei Arten von Teilsicherungen 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 Teilsicherungsimage enthalten. Der Vorgänger eines Teilsicherungsimage 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 Deltasicherungsimage 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 Sicherung geänderten oder neu hinzugekommenen Daten. Deltasicherungsimages enthalten hingegen nur die seit der letzten Sicherung 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 Datenbanksicherungsimage (oder eines Image 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 zutreffenden Teilsicherungen in der nachfolgend beschriebenen Reihenfolge anwenden (siehe „Wiederherstellen von Teilsicherungsimages” auf Seite 33).

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.
- YES. Teilsicherungen sind bei dieser Konfiguration zulässig. Wird die Aktualisierungsüberwachung aktiviert, wird diese Änderung bei der nächsten erfolgreichen Verbindungsherstellung zu einer beliebigen Datenbank im Exemplar wirksam. Bevor eine Teilsicherung erstellt werden kann, muss zuerst eine Gesamtsicherung der Datenbank vorgenommen werden.

Die Standardeinstellung von *trackmod* lautet für vorhandene Datenbanken `NO`, für neue Datenbanken `YES`.

Bei SMS-Tabellenbereichen erfolgt die Unterteilung dieser Überwachung auf Tabellenbereichsebene. Bei DMS-Tabellenbereichen erfolgt die Unterteilung für Daten und Indexseiten auf Speicherbereichsebene, und für andere Seitenarten auf Tabellenbereichsebene.

Die Überwachung von Datenbankaktualisierungen kann sich, wenn auch nur minimal, auf die Laufzeitleistung von Transaktionen auswirken, die Daten aktualisieren oder einfügen.

Wiederherstellen von Teilsicherungsimages

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

1. Identifizieren des Zielimage der Teilwiederherstellung.

   Der Datenbankadministrator muss zuerst das endgültige wiederherzustellende Image ermitteln und eine Teilwiederherstellungsoperation vom DB2-Wiederherstellungsdienstprogramm anfordern. Dieses Image wird als Zielimage der Teilwiederherstellung bezeichnet, da es das letzte Image ist, das wiederhergestellt wird. Wenn für dieses Image ein Befehl zur Teilwiederherstellung ausgeführt wird, kann dies das Erstellen einer neuen Datenbank einleiten, die die Konfigurations- und Tabellenbereichsdefinitionen aus diesem Zielimage erhält. 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 Gesamtsicherungsimage der Datenbank gelöscht wurde, werden die Tabellenbereichsdaten für dieses Image zwar aus den Sicherungsimages gelesen, aber bei der Verarbeitung der Teilwiederherstellung ignoriert.

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

```
db2 restore db sample incremental automatic taken at 20001228152133
```

Dieser Befehl veranlasst, dass das DB2-Wiederherstellungsdienstprogramm automatisch die oben 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 Wiederherstellung nicht möglich, so dass Sie die Wiederherstellungsprozedur manuell fortsetzen müssen.

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 Gesamtsicherungsimage der Datenbank erforderlich sind. Mit der Option FORCE ist es jedoch möglich, Einträge zu löschen, die für eine automatische Wiederherstellungsoperation benötigt werden.

Wenn das Datenbankprotokoll nicht verfügbar ist, können Sie manuell eine Teilwiederherstellung ausführen. Führen Sie dazu die am Anfang dieses Abschnitts beschriebenen Schritte aus. Zum Beispiel:

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

   Dabei gilt Folgendes:
     <zm> zeigt auf das zuletzt wiederherzustellende Teilsicherungsimage.

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.
```

In Fällen, in denen eine Wiederherstellungsoperation für eine Datenbank ausgeführt werden soll und zuvor Teilsicherungsimages von Tabellenbereichen erstellt wurden, müssen die Tabellenbereichsimages in der chronologischen Reihenfolge der Zeitmarken ihrer Sicherung wiederhergestellt werden.

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.

   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>
   ```

   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 Sicherungsimage hatte. Alle nach dieser Zeit 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 nicht korrekte Sicherungsimagekette zu erstellen, so dass es eine entsprechende Fehlermeldung (″in falscher Reihenfolge″) ausgibt.

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.

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 drei Arten von Protokollierung zur Verfügung: *Umlaufprotokollierung*, *Capture-Protokollierung* und *Archivprotokollierung*. Jede Art stellt eine andere Stufe der Wiederherstellbarkeit bereit:

- *Umlaufprotokollierung* ist die Standardprotokollierung, wenn eine neue Datenbank erstellt wird. (Der Datenbankkonfigurationsparameter *logretain* muss hierbei die Einstellung `NO` haben.) Bei dieser Art der Protokollierung sind nur vollständige Offlinesicherungen der Datenbank gültig. 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. Die Datenbank muss offline sein (d. h. für Benutzer nicht zugänglich), wenn eine Gesamtsicherung erstellt wird. Da diese Art der Wiederherstellung die Daten auf dem Stand des Zeitpunkts der Gesamtsicherung wiederherstellt, wird sie als *Versionswiederherstellung* bezeichnet.

  Abb. 6 auf Seite 37 zeigt, dass die aktive Protokolldatei mit einem Ring aus Protokolldateien arbeitet, wenn die Umlaufprotokollierung aktiv ist.

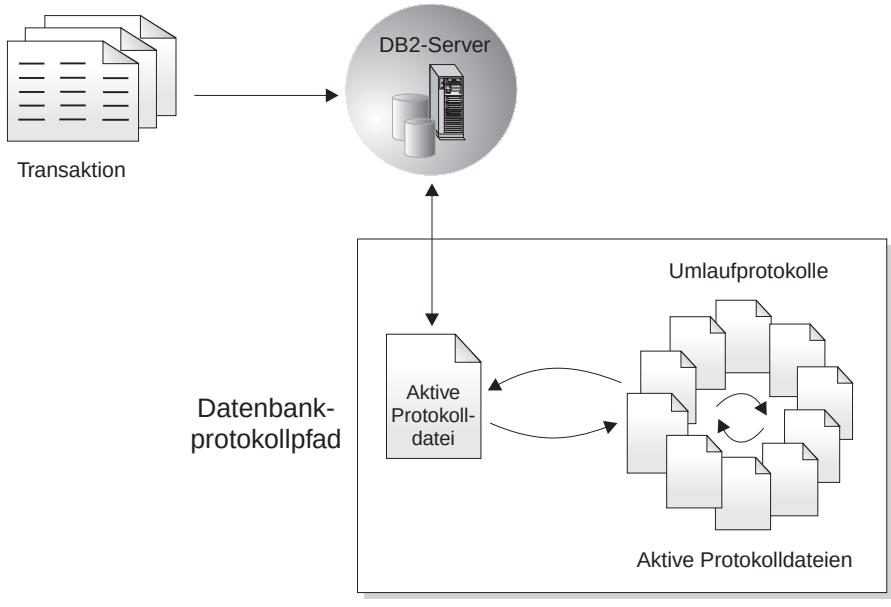


Abbildung 6. Umlaufprotokollierung

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. Der Befehl RESTART DATABASE verwendet die aktiven Protokolle bei Bedarf, um die Datenbank in einen konsistenten und brauchbaren Status zu versetzen. Während einer Wiederherstellung nach einem Systemabsturz werden in diesen Protokollen aufgezeichnete, noch nicht festgeschriebene Änderungen rückgängig gemacht. Festgeschriebene, aber noch nicht aus dem Hauptspeicher (Pufferpool) auf Platte (Datenbankbehälter) geschriebene Änderungen werden wiederholt. Durch diese Maßnahmen wird die Integrität der Datenbank gewährleistet. Aktive Protokolldateien befinden sich im Verzeichnispfad der Datenbankprotokolle.

- *Capture-Protokollierung* wird konfiguriert, indem Sie den Datenbankkonfigurationsparameter *logretain* auf `CAPTURE` setzen. Capture-Protokollierung wird zur Replikationsverarbeitung verwendet. Die Protokolldateien werden bis zur Beendigung der Replikationsverarbeitung beibehalten und danach automatisch gelöscht. Alle DB2-Dienstprogramme handhaben diesen Protokollierungsmodus auf die gleiche Weise wie die Umlaufprotokollierung, d. h., es sind keine Onlinesicherungen, Sicherungen und Wiederherstellungen von Tabellenbereichen oder aktualisierende Wiederherstellungen zulässig, und Ladeoperationen, bei denen die Option RECOVERY NO angegeben ist, versetzen Tabellenbereiche nicht in den Status *Sicherung anstehend*.

- *Archivprotokollierung* wird besonders für die aktualisierende Wiederherstellung verwendet. Diese Protokollierungsart wird konfiguriert, indem Sie den Datenbankkonfigurationsparameter *logretain* auf `RECOVERY` setzen. Archivierte Protokolldateien können Folgendes sein:

  Onlinearchivprotokolldateien
  Wenn die Änderungen in der aktiven Protokolldatei für die normale Verarbeitung nicht mehr benötigt werden, wird die Protokolldatei geschlossen und wird somit zu einer Archivprotokolldatei. Eine Archivprotokolldatei wird als *online* bezeichnet, wenn sie im Verzeichnispfad der Datenbankprotokolle gespeichert ist (siehe Abb. 7).

  Offlinearchivprotokolldateien
  Eine Archivprotokolldatei wird als *offline* bezeichnet, wenn sie sich nicht mehr im Verzeichnispfad der Datenbankprotokolle befindet ist (siehe Abb. 8 auf Seite 39). Sie haben auch die Möglichkeit, mit Hilfe eines Benutzer-Exit-Programms Archivprotokolldateien an einer anderen Speicherposition als im Verzeichnispfad der Datenbankprotokolle zu speichern. (Zusätzliche Informationen finden Sie in „Anhang H. Benutzer-Exit zur Datenbankwiederherstellung" auf Seite 493.)

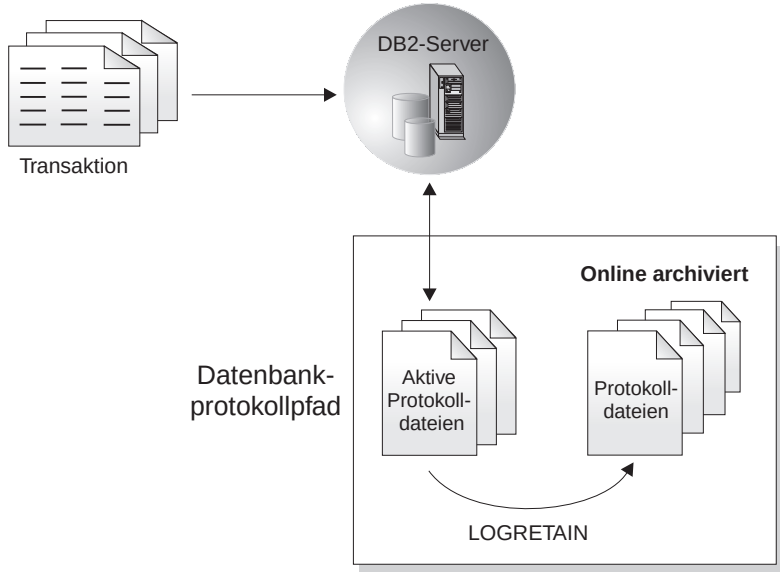


Abbildung 7. Onlinearchivierung von Protokolldateien

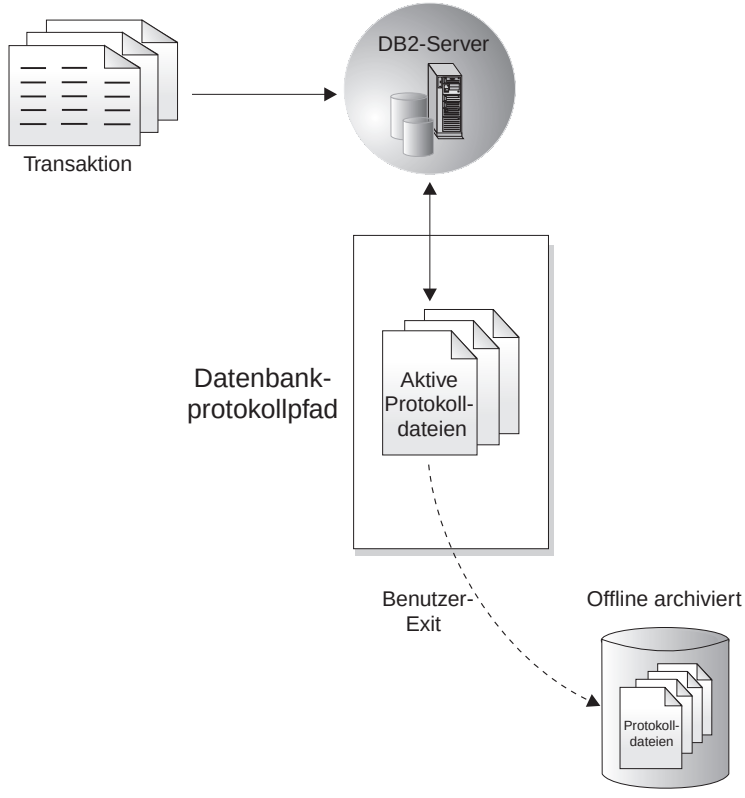


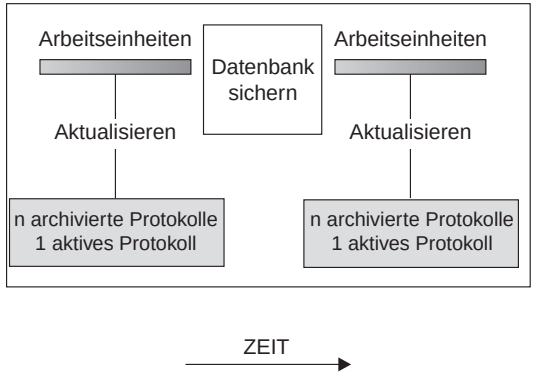
Abbildung 8. Offlinearchivierung von Protokolldateien

Von der aktualisierenden Wiederherstellung können sowohl archivierte als auch aktive Protokolldateien verwendet werden, um eine Datenbank entweder bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt wiederherzustellen. Das Dienstprogramm zur aktualisierenden Wiederherstellung erreicht dies, indem es die in den archivierten und aktiven Protokolldateien gefundenen festgeschriebenen Änderungen erneut auf die wiederhergestellte Datenbank anwendet.

Die aktualisierende Wiederherstellung kann mit Hilfe von Protokollen auch einen Tabellenbereich wiederherstellen, indem festgeschriebene Aktualisierungen in den archivierten und aktiven Protokolldateien erneut angewendet werden. Sie können einen Tabellenbereich bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt wiederherstellen.

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

Mit Hilfe zweier Datenbankkonfigurationsparameter können Sie die Position ändern, an der die archivierten Protokolldateien gespeichert werden sollen: *newlogpath* und *userexit*. Änderungen am Parameter *newlogpath* wirken sich außerdem auf die Speicherposition aus, an der die aktiven Protokolle gespeichert werden. Weitere Informationen zu diesen Konfigurationsparametern finden Sie im Handbuch *Systemverwaltung: Optimierung*.

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*.

Anmerkungen:

1. Wenn Sie ein aktives Protokoll löschen, wird die Datenbank unbrauchbar. Eine Weiterverwendung der Datenbank ist dann nur möglich, wenn Sie die Datenbank wiederherstellen. Außerdem können Sie eine aktualisierende Wiederherstellung nur bis zum ersten gelöschten Protokoll durchführen.
2. Wenn Sie befürchten, dass Ihre aktiven Protokolle beschädigt werden könnten (aufgrund eines Plattenfehlers), sollten Sie ein Spiegeln in Betracht ziehen, entweder auf Betriebssystemebene durch Spiegeln der Platte, auf der die Protokolle gespeichert sind, oder auf DB2-Ebene durch Verwenden der Registrierungsvariablen NEWLOGPATH.

Spiegeln von Protokollen

DB2 bietet nun 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 beschädigt werden könnten (aufgrund eines Plattenfehlers), sollten Sie in Betracht ziehen, mit der neuen DB2-Registrierungsvariablen NEWLOGPATH2 einen sekundären Pfad für die Datenbank zur Verwaltung von Kopien des aktiven Protokolls anzugeben, so dass die Datenträger gespiegelt werden, auf denen die Protokolle gespeichert sind.

Die Registrierungsvariable NEWLOGPATH2 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 einzigen Fehlerpunkt darstellt.

Anmerkung: Da unter Windows NT das „Anhängen" einer Einheit unter einem willkürlichen Pfadnamen nicht zulässt, kann auf dieser Plattform kein sekundärer Pfad auf einer separaten Einheit angegeben werden.

NEWLOGPATH2 kann aktiviert (auf `1` gesetzt) oder inaktiviert (auf `0` gesetzt) werden. Der Standardwert ist Null. Wenn diese Variable auf `1` gesetzt ist, entspricht der Name des sekundären Pfads dem aktuellen Wert der Variablen LOGPATH, mit einer am Ende des Werts angehängten Ziffer `2`. Beispiel: Wenn in einer SMP-Umgebung LOGPATH den Wert `/u/dbuser/sqllogdir/logpath` hat, lautet der sekundäre Protokollpfad `/u/dbuser/sqllogdir/logpath2`. Wenn

in einer MPP-Umgebung LOGPATH den Wert /u/dbuser/sqllogdir/logpath hat, fügt DB2 dem Pfad den Knotenbezugswert hinzu und verwendet als sekundären Protokollpfad /u/dbuser/sqllogdir/logpath/NODE0000. In diesem Fall lautet der sekundäre Protokollpfad /u/dbuser/sqllogdir/logpath2/NODE0000.

Wird NEWLOGPATH2 zum ersten Mal aktiviert, wird die Variable erst beim Beenden der aktuellen Protokolldatei beim nächsten Start der Datenbank verwendet. Dies entspricht der Weise, auf die LOGPATH und NEWLOGPATH derzeit verwendet werden.

Wenn beim Schreiben in Pfad 1 oder in Pfad 2 ein Fehler auftritt, markiert die Datenbank den entsprechenden Pfad als „unbrauchbar", schreibt eine Nachricht in die Datei db2diag.log 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 beendet wird. 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.

Verringern der Protokollierung für Arbeitstabellen

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. (Dies funktioniert jedoch nur, wenn die Tabelle mit der Option NOT LOGGED INITIALLY erstellt wurde.)

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, den Parameter NOT LOGGED INITIALLY zu verwenden, Folgendes beachten:

- *Alle* Änderungen an der Tabelle müssen zum COMMIT-Zeitpunkt auf Platte geschrieben werden. Das bedeutet, dass die COMMIT-Operation länger dauern kann.
- Ein Fehler, der für irgendeine Operation in der Arbeitseinheit, in der die Tabelle erstellt wird, zurückgegeben wird, führt dazu, dass die gesamte Arbeitseinheit rückgängig gemacht wird (SQLCODE-Wert -1476, SQLSTATE-Wert 40506).
- Diese Tabellen können Sie bei einer aktualisierenden Wiederherstellung nicht wiederherstellen. Wenn die aktualisierende Wiederherstellung eine Tabelle antrifft, 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. Weitere Informationen finden Sie im Handbuch *Systemverwaltung: Optimierung* im Abschnitt über den gemeinsamen Zugriff.

Weitere Informationen zum Erstellen von Tabellen finden Sie im Handbuch *SQL Reference*.

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 Zeile 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.

Weitere Informationen zu deklarierten temporären Tabellen und ihren Einschränkungen finden Sie im Abschnitt zur Anweisung DECLARE GLOBAL TEMPORARY TABLE des Handbuchs *SQL Reference*.

Konfigurationsparameter für die Datenbankprotokollierung

Die Konfigurationsdatei der Datenbank enthält Parameter für die aktualisierende Wiederherstellung. Diese Art der Wiederherstellung wird von den Standardwerten der Parameter nicht unterstützt. Wenn Sie beabsichtigen, diese Methode zu verwenden, müssen Sie daher einige der Standardwerte ändern. Weitere Informationen zum Konfigurieren von DB2 Universal Database finden Sie im Handbuch *Systemverwaltung: Optimierung*.

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

Ein primäres Protokoll belegt in leerem sowie in vollem 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 Anwendung volle Protokolle behandeln kann.

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.

Die Protokolldatei kann maximal 32 GB groß sein. Dies bedeutet, dass die Anzahl der Protokolldateien (*logprimary* + *logsecond*) multipliziert mit der Größe der einzelnen Protokolldateien in Byte (*logfilsiz* * 4096) kleiner als 32 GB sein muss.

Weitere Informationen zu diesem Konfigurationsparameter finden Sie im Handbuch *Systemverwaltung: Optimierung*.

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 eine größere Anzahl sekundärer Protokolldateien als die konfigurierte Anzahl benötigt wird, wird ein Fehler zurückgegeben, und die Aktivitäten für diese Datenbank werden gestoppt.

Weitere Informationen zu diesem Konfigurationsparameter finden Sie im Handbuch *Systemverwaltung: Optimierung*.

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

Es besteht eine logische Begrenzung von 32 GB für den konfigurierbaren Gesamtspeicherplatz des aktiven Protokolls. Diese Begrenzung ergibt sich aus dem oberen Grenzwert für *logfilsiz*, der bei 65535 liegt, und dem oberen Grenzwert für (*logprimary* + *logsecond*), der bei 128 liegt. Daraus ergibt sich: ((*logprimary* + *logsecond*) * *logfilsiz*) < (32 GB / 4096).

Die Größe der Protokolldatei hat eine direkte Auswirkung auf die Leistung. Wenn die Datenbank für das Beibehalten von Protokollen definiert wurde, wird jedesmal, wenn ein Protokoll voll ist, eine Anforderung für die Zuordnung und Initialisierung eines neuen Protokolls abgesetzt. Durch Erhöhen der Protokollgröße kann die Anzahl der Anforderungen, die für die Zuordnung und Initialisierung neuer Protokolle erforderlich sind, verringert werden. (Beachten Sie dabei jedoch, dass bei einer Erhöhung der Protokollgröße mehr Zeit für das Formatieren jedes neuen Protokolls benötigt wird.) Das Formatieren neuer Protokolle ist für die Anwendungen mit einer Verbindung zur Datenbank transparent, d. h., die Datenbankleistung wird durch die Formatierung nicht beeinträchtigt.

Angenommen, Sie haben eine Anwendung, die die Datenbank geöffnet hält, um die Verarbeitungszeit beim Öffnen von Datenbanken zu minimieren (siehe „Verbessern der Wiederherstellungsleistung" auf Seite 66). In diesem Fall sollte der Wert für die Protokollgröße durch den Zeitraum bestimmt werden, der erforderlich ist, um Kopien von Offlinearchivprotokolldateien anzulegen.

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. Entscheidend ist dabei die 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. (Informationen zur Verwendung unterschiedlicher Datenträger für die Speicherung von Protokollen finden Sie in „Anhang H. Benutzer-Exit zur Datenbankwiederherstellung" auf Seite 493.) 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.

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.

Protokollpuffer (logbufsz)

Mit diesem Parameter kann die Menge des gemeinsamen Speichers 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.

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.

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. Archivprotokolldateien, 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 aktuali-

sierenden 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. Weitere Informationen zu diesem Konfigurationsparameter finden Sie im Handbuch *Systemverwaltung: Optimierung*. Informationen zur Rolle der Datenbankprotokolle bei Inkonsistenz einer Datenbank finden Sie in „Verwalten von Protokolldateien".

Beibehalten von Protokollen (logretain)

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.

Wenn *logretain* auf `CAPTURE` gesetzt ist, ruft das Capture-Programm für Datenreplikation den Befehl PRUNE LOGFILE auf, um die Protokolldateien bei Beendigung des Capture-Programms zu löschen. Sie sollten *logretain* nicht auf `CAPTURE` setzen, wenn Sie die Möglichkeit haben wollen, für die Datenbank eine aktualisierende Wiederherstellung durchzuführen.

Benutzer-Exit (userexit)

Der Konfigurationsparameter *userexit* weist den Datenbankmanager an, ein Benutzer-Exit-Programm 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. Informationen zu Benutzer-Exit-Programmen finden Sie in „Anhang H. Benutzer-Exit zur Datenbankwiederherstellung" auf Seite 493.

Verwalten von Protokolldateien

Bei der Verwaltung von Datenbankprotokollen sind folgende Faktoren zu beachten:

- Das Numerierungsschema 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 *inaktivieren*.
  - S9999999.LOG wurde verwendet

DB2 verwendet die Protokollnamen 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. Er ist jedoch nicht in der Lage, die Speicherposition des erforderlichen Protokolls zu ermitteln. Sie müssen sicherstellen, dass die korrekten 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 Benutzer-Exit-Programm aufrufen, um die Protokolle an eine andere Position zu kopieren. Informationen zu Benutzer-Exit-Programmen finden Sie in „Anhang H. Benutzer-Exit zur Datenbankwiederherstellung" auf Seite 493.)

- 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 wiederverwendet werden (siehe Abb. 10 auf Seite 51). 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 verfügen, 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. Das Wiederherstellungsdienstprogramm kann 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 eines Tabellenbereichs wird jedoch fehlschlagen, und der Tabellenbereich befindet sich weiter im Status *Aktualisierende 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. 10 auf Seite 51). Wenn Sie unter Verwendung des Sicherungsimage 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 Sicherungsimage 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 Sicherungsimage auf Tabellenbereichsebene, Sicherung 3, können Sie eine aktualisierende Wiederherstellung bis zum Ende der Protokolle ausführen.

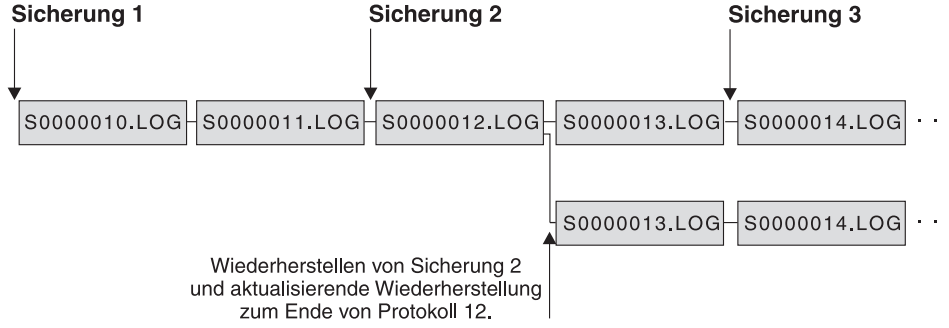


Abbildung 10. Erneutes Verwenden der Namen von Protokolldateien

Verwalten von Protokolldateien mit einem Benutzer-Exit-Programm

Folgende Hinweise und Informationen sind für das Aufrufen eines Benutzer-Exit-Programms zum Archivieren und Abrufen von Protokolldateien zu beachten:

- Der Parameter *userexit* der Datenbankkonfigurationsdatei gibt an, ob der Datenbankmanager während der aktualisierenden Wiederherstellung von Datenbanken ein Benutzer-Exit-Programm zum Archivieren von Dateien oder zum Abrufen von Protokolldateien aufruft. Es erfolgt eine Anforderung zum Abrufen einer Protokolldatei, wenn das Dienstprogramm zur aktualisierenden Wiederherstellung für die aktualisierende Wiederherstellung eine Protokolldatei benötigt, die nicht im Verzeichnis für den Protokollpfad zu finden ist.

  Anmerkung: Unter Windows NT können Sie keinen REXX-Benutzer-Exit zum Archivieren von Protokollen verwenden.

- Beim Archivieren wird eine Protokolldatei an den Benutzer-Exit 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 Benutzer-Exit 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 Benutzer-Exit-Programm zum Archivieren einer Protokolldatei startet. Auf einigen Plattformen nimmt dies dem Benutzer-Exit-Programm die Fähigkeit, die Protokolldatei zu löschen. Andere Plattformen wie AIX ermöglichen es Prozessen, einschließlich des Benutzer-Exit-Programms, Protokolldateien zu löschen. Ein Benutzer-Exit-Programm sollte eine Protokolldatei nie nach ihrer Archivierung löschen können, da die Datei weiterhin aktiv sein könnte und für

die Wiederherstellung nach Systemabsturz benötigt werden kann. DB2 verwaltet bei der Archivierung vn Protokolldateien die erneute Verwendung von Plattenspeicherplatz.

- Wenn eine Protokolldatei archiviert wurde und inaktiv ist, 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 günstiger, die erforderlichen Seiten auf der Platte wiederzuverwenden, als sie freizugeben und neu zuzuordnen.
- DB2 ruft das Benutzer-Exit-Programm weder bei einer Wiederherstellung nach Systemabsturz noch nach einer ROLLBACK-Operation zum Abrufen der Protokolldatei auf.
- Ein Benutzer-Exit-Programm garantiert nicht die aktualisierende Wiederherstellung bis zum Fehlerpunkt, sondern versucht lediglich, das Fehlerfenster kleiner zu machen. Wenn die Protokolldateien voll werden, werden sie für die Benutzer-Exit-Routine 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 jeder einzelnen Protokolldatei hat direkte Auswirkungen auf den Benutzer-Exit. 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 Benutzer-Exit-Routine 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. Wenn die Warteschlange voll wird, werden Anforderungen zum Archivieren und Abrufen nicht verarbeitet. Die Verarbeitung wird erst wieder aufgenommen, wenn wieder Platz in der Warteschlange ist. Nicht verarbeitete Anforderungen werden automatisch erneut in die Warteschlange gestellt.
- Eine Anforderung zum Archivieren erfolgt nur dann an das Benutzer-Exit-Programm, wenn der Parameter *userexit* konfiguriert ist und eine aktive Protokolldatei voll wird. Es ist möglich, dass eine aktive Protokolldatei nicht voll ist, wenn die letzte Verbindung von der Datenbank getrennt wird, und das Benutzer-Exit-Programm 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 die Datenbank geschlossen wird, bevor eine positive Antwort für eine Archivierungsanforderung von einem Benutzer-Exit-Programm 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 Benutzer-Exit-Programm 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 Benutzer-Exit-Programm 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 (siehe „Verwalten von Protokolldateien" auf Seite 48).
- Wenn zwei oder mehr Datenbanken eine Einheit gleichzeitig verwenden und eine der Operationen eine aktualisierende Wiederherstellung beinhaltet, ist es möglich, dass eine für die aktualisierende Wiederherstellung benötigte Protokolldatei auf dem momentan im Laufwerk eingelegten Datenträger nicht existiert. Es können zwei Bedingungen eintreten:
  - Wenn das Benutzer-Exit-Programm einen Rückkehrcode null (erfolgreiche Ausführung) an den Datenbankmanager übergibt und die angeforderte Protokolldatei nicht abgerufen wurde, nimmt der Datenbankmanager an, dass die aktualisierende Wiederherstellung bis zum Ende der Protokolle ausgeführt wurde, und die aktualisierende Wiederherstellung wird gestoppt. Die aktualisierende Wiederherstellung ging jedoch möglicherweise nicht bis zum Ende der Protokolle.
  - Wenn ein Rückkehrcode ungleich null übergeben wird, verbleibt die Datenbank im Status *Aktualisierende Wiederherstellung anstehend*, und die aktualisierende Wiederherstellung muss entweder wiederaufgenommen oder beendet werden.

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

Blockieren von Transaktionen bei vollem Protokollverzeichnis

Die neue DB2-Registrierungsvariable DB2_BLOCK_ON_LOG_DISK_FULL kann so eingestellt werden, dass keine Fehlermeldungen aufgrund von mangelndem Plattenplatz 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 in der Datenbankkonfiguration der Parameter *userexit* auf `ON` gesetzt ist, überprüft DB2 außerdem den Beendigungsstatus der Protokolldateiarchivierung. Wenn eine inaktive Protokolldatei erfolgreich archiviert wurde, kann DB2 die inaktive Protokolldatei in den neuen Protokolldateinamen umbenennen und fortfahren. Nach jedem Versuch schreibt DB2 eine Nachricht in die Datei `db2diag.log`. Sie können nur durch Überwachen der Datei `db2diag.log` 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.

Bedarfsgesteuerte Protokollarchivierung

DB2 bietet nun Unterstützung für das Schließen (bei aktivierter Benutzer-Exit-Option auch für das Archivieren) des aktiven Protokolls 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 (API) **db2ArchiveLog** aufrufen. Beschreibungen hierzu finden Sie in „ARCHIVE LOG" auf Seite 362 bzw. in „db2ArchiveLog - Aktive Protokolldatei archivieren (API)" auf Seite 376.

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:
  - Die Einheiten können nicht von anderen Anwendungen mitverwendet werden. Die gesamte Einheit *muss* DB2 zugeordnet sein.
  - 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. Ein beispiel für die Syntax bei der Angabe einer unformatierten Einheit finden Sie im Handbuch *Systemverwaltung: Implementierung* im Abschnitt über unformatierte Ein-/Ausgabe. 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 plattengreifende 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 von Plattenspeicherplatz, auf die 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 jeder eine Protokolldateigröße (*logfilsiz*) von 4-KB-Seiten hat. Protokollspeicherbereiche werden nacheinander in die unformatierte Einheit platziert. Jeder Speicherbereich besteht zudem aus zwei zusätzlichen Seiten für den Speicherbereichs-Header. 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 Protokolle unterstützen zu können. Das heißt, dass die *Anzahl verfügbarer Protokollspeicherbereiche* größer als der (oder gleich dem) Wert sein muss, der für den Konfigurationsparameter *logprimary* angegebenen ist.
- 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.
- Bei Verwendung von Protokollspeicherung (*logretain*) ohne Benutzer-Exit-Programm empfangen alle Operationen, die zu einer Aktualisierung führen, nach Ausschöpfen der *Anzahl verfügbarer Protokollspeicherbereiche* 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.
- Bei Verwendung von Protokollspeicherung (*logretain*) mit Benutzer-Exit-Programm wird das Benutzer-Exit-Programm für jeden Protokollspeicherbereich aufgerufen, wenn er mit Protokollsätzen gefüllt ist. Das Benutzer-Exit-Programm muss die Einheit lesen und das Archivprotokoll als Datei speichern können. DB2 ruft kein Benutzer-Exit-Programm auf, um Protokolldateien für eine unformatierte Einheit abzurufen. Stattdessen liest DB2 während der aktualisierenden Wiederherstellung den Speicherbereichs-Header, 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 das Benutzer-Exit-Programm auf, um die Protokolldatei in den Überlauf-

protokollpfad abzurufen. Wenn Sie für die aktualisierende Wiederherstellung keinen Überlaufprotokollpfad angeben, ruft DB2 während der Operationen zur aktualisierenden Wiederherstellung das Benutzer-Exit-Programm nicht zum Abruf der Protokolldatei auf. Zusätzliche Informationen zum Aufrufen eines Benutzer-Exit-Programms finden Sie in „Aufrufformat" auf Seite 496.

- Wenn Sie DataPropagator Relational (DPROP) verwenden und Protokolle auf eine unformatierte Einheit schreiben, ruft die API zum Lesen des Protokolls das Benutzer-Exit-Programm nicht zum Abrufen der Protokolldateien auf. Angeforderte Protokollsätze werden jedoch weiterhin zurückgegeben, sofern sie auf der Einheit verfügbar sind. Wenn Sie Protokolle anfordern, die den ältesten auf der Einheit zeitlich vorangehen, werden sie nicht zurückgegeben (diese Arbeitsweise ähnelt der von DB2, wenn die Protokolldatei nicht auffindbar ist, die die angeforderten Protokollsätze enthält).

  Anmerkungen:
  1. Es wird empfohlen, DataPropagator Relational (DPROP) bei Verwendung einer unformatierten Einheit zum Protokollieren nicht einzusetzen.
  2. Wenn Sie die API **sqlurlog** verwenden, sollten Sie zum Protokollieren keine unformatierte Einheit einsetzen.

Verlust von Protokollen

Beim Löschen einer Datenbank werden alle Protokolle im aktuellen Verzeichnis des Protokollpfads der Datenbank gelöscht. Vor dem Löschen einer Datenbank sollten Sie eventuell Kopien der Protokolle anlegen.

Wenn Sie eine Datenbank bis zu einem bestimmten Zeitpunkt aktualisierend wiederherstellen, werden das letzte Protokoll, das dabei verwendet wurde, und alle vorhandenen nachfolgenden Protokolle erneut verwendet. Aus diesem Grund sollten Sie alle Protokolle im aktuellen Verzeichnis des Protokollpfads der Datenbank kopieren, *bevor* Sie mit einer Wiederherstellung bis zu einem bestimmten Zeitpunkt beginnen.

Bei Beendigung der aktualisierenden Wiederherstellung wird die Protokolldatei mit der zuletzt festgeschriebenen Transaktion abgeschnitten, und die Protokollierung beginnt mit dem nächsten sequenziellen Protokoll. Wenn Sie weder über eine Kopie des Protokolls, die vor dem Abschneiden erstellt wurde, noch über Protokolle mit höheren Folgenummern verfügen, können Sie keine Wiederherstellung der Datenbank über den angegebenen Zeitpunkt hinaus ausführen. (Sobald nach einer aktualisierenden Wiederherstellung die normalen Datenbankaktivitäten wieder aufgenommen werden, werden neue Protokolle erstellt, die dann für nachfolgende Wiederherstellungen verwendet werden können.)

Wenn Sie das Protokollpfadverzeichnis ändern und anschließend das Unterverzeichnis entfernen oder in dem im Protokollpfad angegebenen Unterverzeichnis befindliche Protokolle löschen, sucht der Datenbankmanager beim Öffnen der Datenbank im Standardprotokollpfad SQLOGDIR nach den Protokollen. Falls die Protokolle nicht auffindbar sind, wird die Datenbank in den Status *Sicherung anstehend* versetzt. In diesem Fall müssen Sie eine Datenbanksicherung durchführen, damit die Datenbank wieder verwendbar ist. Diese Sicherung muss auch dann ausgeführt werden, wenn das Unterverzeichnis leere Protokolle enthielt.

Wenn Sie das Protokoll verlieren, das den Endzeitpunkt der Onlinesicherung enthält, und eine aktualisierende Wiederherstellung des entsprechenden wiederhergestellten Image ausführen, ist die Datenbank nicht mehr verwendbar. Um die Datenbank wieder verwendbar zu machen, müssen Sie sie mit einer anderen Sicherung und allen zugehörigen Protokollen wiederherstellen.

Stellen Sie sich beispielsweise folgende Situation vor: Sie möchten eine Wiederherstellung einer vollständigen Datenbank bis zu einem bestimmten Zeitpunkt ausführen, befürchten jedoch, dass während des Wiederherstellungsprozesses ein Protokoll verlorengehen könnte. (Eine solche Situation könnte eintreten, wenn es eine große Anzahl archivierter Protokolle zwischen dem Zeitpunkt der Erstellung des letzten Sicherungsimage der Datenbank und dem Zeitpunkt gibt, bis zu dem die Wiederherstellung durchgeführt werden soll.)

Als erstes sollten Sie alle gültigen Protokolle an eine „sichere" Speicherposition kopieren. Anschließend können Sie den Befehl RESTORE ausführen und die Datenbank bis zum gewünschten Zeitpunkt aktualisierend wiederherstellen. Falls bei diesem Prozess Protokolle beschädigt werden oder verlorengehen, können Sie auf die an anderer Stelle gespeicherten Sicherungskopien aller Protokolle zurückgreifen.

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
- Reorganisieren einer Tabelle
- Aktualisieren von Tabellenstatistiken

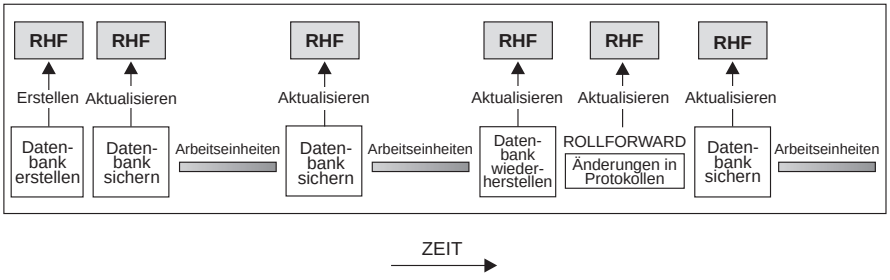


Abbildung 11. Erstellen und Aktualisieren der Datei des Wiederherstellungsprotokolls

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

Datei des Wiederherstellungsprotokolls

Verwenden Sie den Befehl LIST HISTORY, wenn Sie die Einträge in der Datei des Wiederherstellungsprotokolls anzeigen wollen. Weitere Informationen zu diesem Befehl finden Sie in „LIST HISTORY" auf Seite 366.

Anmerkung: Wenn eine Wiederherstellung und anschließend eine aktualisierende Wiederherstellung bis zum Ende aller Protokolle ausgeführt wird, steht die nach Aufruf des Befehls LIST HISTORY angezeigte Sicherungs-ID für das Ende der Verarbeitungszeit, d. h., die Sicherungs-ID hat den Wert 99991231235959. Die Sicherungs-ID wird in dieser Weise nur dann umgesetzt, wenn eine aktualisierende Wiederherstellung ausgeführt wird.

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 mit der Datenbank verknüpft. 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 nicht überschrieben.

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* definiert, 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 aktuellsten 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. Weitere Informationen zu diesem Konfigurationsparameter finden Sie im Handbuch *Systemverwaltung: Optimierung*.

Garbage Collection

Auch wenn Sie den Befehl PRUNE HISTORY (siehe „PRUNE HISTORY/LOGFILE" auf Seite 369) 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 nach erfolgreicher Beendigung einer Datenbanksicherung oder einer Datenbankwiederherstellung automatisch aufgerufen. Mit dem Konfigurationsparameter *num_db_backups* wird definiert, wie viele Images von Gesamtsicherungen (keine Teilsicherungen) aufbewahrt werden. Der Wert dieses Parameters wird verwendet, um die Protokolldatei ab dem letzten Eintrag zu durchsuchen.

Nach jeder vollständigen Datenbanksicherung 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.

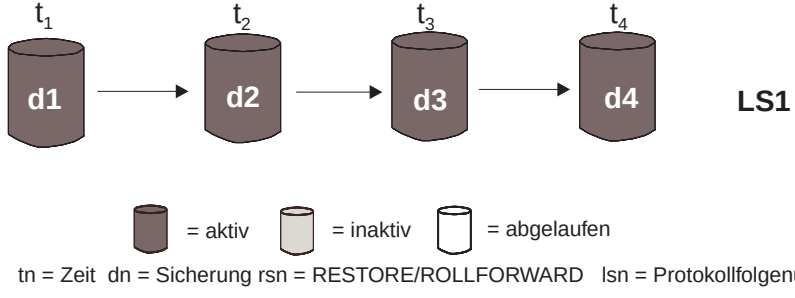


Abbildung 12. Aktive Datenbanksicherungen. Der Wert von *num_db_backups* wurde auf Vier gesetzt.

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.

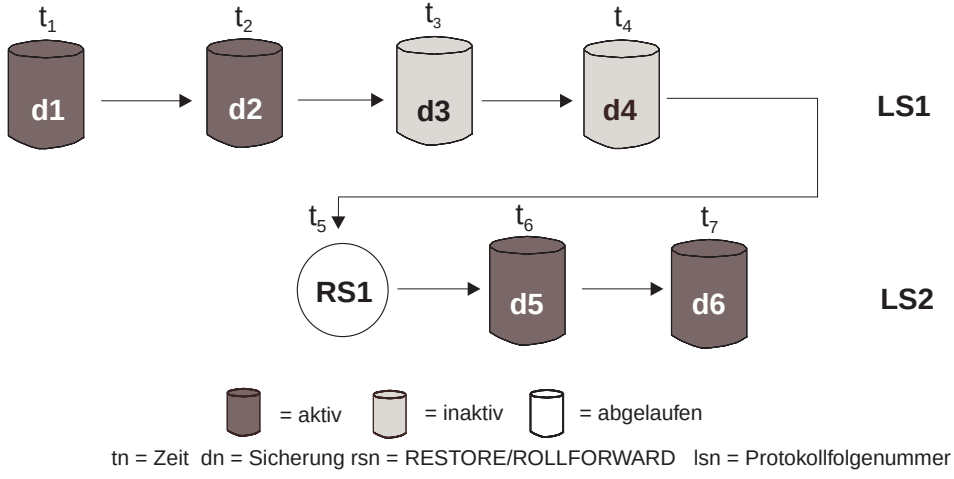


Abbildung 13. Inaktive Datenbanksicherungen

Wenn ein aktives Datenbanksicherungsimage wiederhergestellt wird, es aber nicht die aktuellste 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.

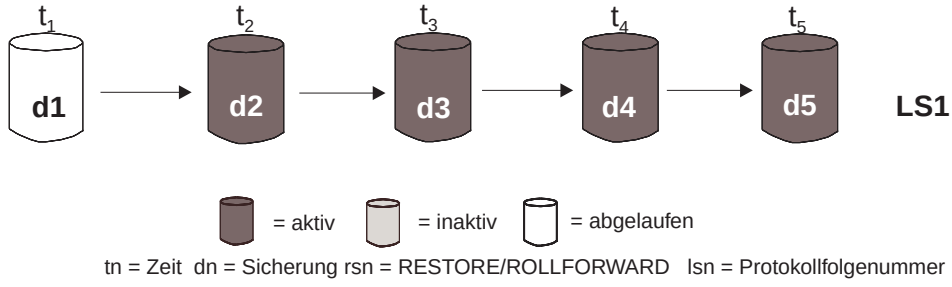


Abbildung 14. Abgelaufene Datenbanksicherungen

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 Datenbanksicherungsimage 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.

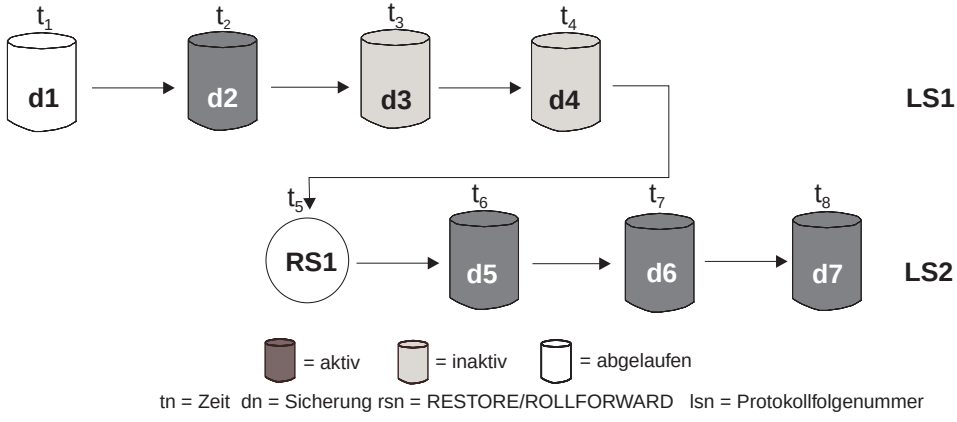


Abbildung 15. Gemischte aktive, inaktive und abgelaufene Datenbanksicherungen

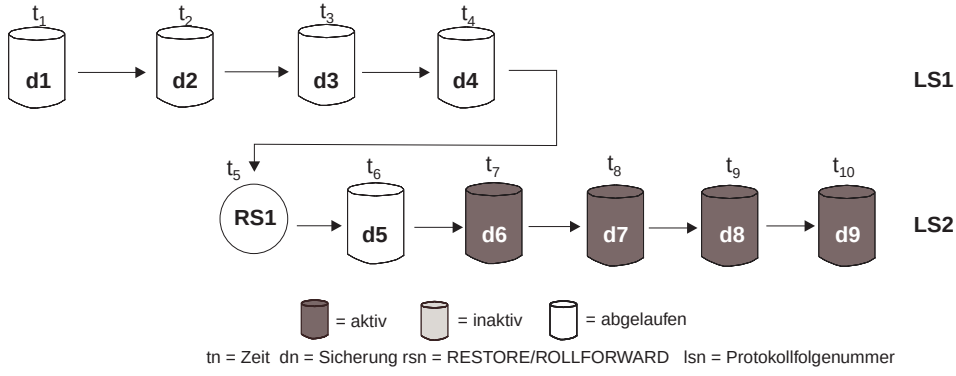


Abbildung 16. Abgelaufene Protokollfolge

Tabellenbereichsstatus

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

- *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.
- *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.
- *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.
- *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.)

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.
- 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 E/A-Kanäle verwenden, sollten Sie zumindest doppelt so viele Puffer als Kanäle verwenden, um sicherzustellen, dass die Kanäle 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 bzw. Behälter 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 zweimal 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.

Parallele Wiederherstellung

DB2 verwendet nun zum parallelen Wiederherstellen nach Systemabsturz und 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.

Verbessern der Wiederherstellungsleistung

Der neue, mit dieser Verbesserung implementierte Agententyp ist db2agnsc. DB2 wählt die bei der Datenbankwiederherstellung zu verwendende Anzahl Agenten auf Basis der vorhandenen Anzahl CPUs auf der Maschine aus. Für SMP-Maschinen liegt die Anzahl verwendeter Agenten bei (Anzahl CPUs + 1). Auf Maschinen mit einer CPU werden mehrere Agenten eingesetzt für effizienteres Lesen von Protokollen, Verarbeiten von Protokollsätzen und Bereitstellen von Datenseiten.

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.

Überlegungen zu DB2 Data Links Manager

Die folgenden Abschnitte enthalten Informationen zu Tabellen, die DATALINK-Spalten enthalten. Eine vollständige Beschreibung der DATALINK-Spalten finden Sie im Handbuch *SQL Reference* im Abschnitt zur Anweisung CREATE TABLE.

Überlegungen zur Wiederherstellung nach einem Systemabsturz

Wenn eine Anwendung SQL-Anforderungen an Data Links-Server sendet, auf denen DB2 Data Links Manager ausgeführt wird (über DATALINK-Spalten mit dem Attribut FILE LINK CONTROL), verteilt der Datenbankmanager die Arbeit auf die Data Links-Server. Außerdem verfolgt er, welche Data Links-Server an der Transaktion beteiligt sind. Wenn die Anwendung für eine Transaktion den Befehl COMMIT absetzt, schreibt der Datenbankmanager die Transaktion mit Hilfe des Protokolls zur zweiphasigen Festschreibung fest. In der ersten Phase schreibt der Datenbankmanager einen PREPARE-Protokollsatz und sendet an alle Data Links-Server eine PREPARE-Anforderung. Die einzelnen Data Links-Server antworten daraufhin mit einer der folgenden Rückmeldungen:

- YES bedeutet, dass der Data Links-Server zur COMMIT-Operation bereit ist.
- NO bedeutet, dass ein Fehler aufgetreten ist und der Data Links-Server nicht zur COMMIT-Operation bereit ist.

Die erste Phase gilt als erfolgreich, wenn alle Data Links-Server mit „YES" antworten.

Die Verarbeitung in der zweiten Phase hängt vom Ergebnis der ersten Phase ab. Wenn mindestens ein Data Links-Server mit „NO" antwortet, sendet der Datenbankmanager an alle beteiligten Data Links-Server eine ABORT-Anforderung. Die Transaktion wird rückgängig gemacht, und an die Anwendung wird die Fehlernachricht SQL0903N mit dem Ursachencode „03" zurückgegeben. Andernfalls setzt der Datenbankmanager die COMMIT-Operation der Transaktion fort, wie er es ohne beteiligte Data Links-Server normalerweise tut. Am Ende der Verarbeitung sendet er an alle an der Transaktion beteiligten Data Links-Server eine COMMIT-Anforderung.

Wenn auf einem Data Links-Server ein Fehler auftritt, bei dem einige Transaktionen im Status PREPARED bleiben, werden diese Transaktionen als *unbestätigte Transaktionen* bezeichnet. Der Datenbankmanager ist für die Überwachung des Ergebnisses dieser Transaktionen zuständig und löst diese schließlich auf dem Data Links-Server auf. Immer wenn der Datenbankmanager erkennt, dass ein Fehler auf einem Data Links-Server möglicherweise unbestätigte Transaktionen verursacht hat, versetzt er den Data Links-Server in den Status *Wiederherstellung nach einem Systemabsturz erforderlich*. In diesem Status werden alle SQL-Anforderungen, bei denen der Data Links-Server beteiligt ist, zurückgewiesen. Die Fehlernachricht SQL0357N mit dem Ursachencode „03" wird an die Anwendung zurückgegeben, von der die SQL-Anforderung ausging.

Während der Verarbeitung der Befehle RESTART und ACTIVATE DATABASE sowie des ersten Befehls CONNECT versucht der Datenbankmanager, zu jedem konfigurierten Data Links-Server eine Verbindung herzustellen und dessen unbestätigte Transaktionen aufzulösen, indem er diese abbricht oder festschreibt. Ein Data Links-Server wird als verfügbar markiert, wenn alle seine unbestätigten Transaktionen aufgelöst werden, mit Ausnahme der Transaktionen, die auch auf dem Datenbankmanager unbestätigt sind. In diesem Status werden SQL-Anforderungen, an denen Data Links-Server beteiligt sind, zugelassen. Wenn der Datenbankmanager nach dem Ende dieses Versuchs, unbestätigte Transaktionen aufzulösen, feststellt, dass sich möglicherweise auf einem Data Links-Server weiterhin aufzulösende unbestätigte Transaktionen befinden, versetzt er diesen in den Status *Wiederherstellung nach einem Systemabsturz erforderlich*. Dieser Fall tritt z. B. ein, wenn ein Data Links-Server während der Verarbeitung von RESTART, ACTIVATE DATABASE oder dem ersten CONNECT nicht verfügbar ist oder wenn der Data Links-Server während der Verarbeitung dieser Befehle einen Fehler erkennt.

Wenn ein Data Links-Server, der für eine Datenbank konfiguriert ist, sich im Status *Wiederherstellung nach einem Systemabsturz erforderlich* befindet, lässt der Datenbankmanager keine SQL-Anforderungen zu, an denen dieser Data Links-Server beteiligt ist. SQL-Anforderungen, die andere Datenbankdaten betreffen, sind weiterhin zulässig. Der Datenbankmanager startet einen Prozess, bei dem versucht wird, die Wiederherstellung nach einem Systemabsturz

auf den einzelnen Data Links-Servern, auf denen eine Wiederherstellung erforderlich ist, asynchron abzuschließen. Wenn der Prozess der Wiederherstellung nach Systemabsturz erfolgreich ist, wird der Status des Data Links-Servers auf verfügbar gesetzt, wodurch weitere SQL-Anforderungen, an denen er beteiligt ist, zugelassen werden.

Überlegungen zum Sicherungsdienstprogramm

DB2 sorgt dafür, dass bis zu dem Zeitpunkt, zu dem das Sicherungsdienstprogramm beendet wird, verbundene Dateien auf Data Links-Servern, auf denen DB2 Data Links Manager ausgeführt wird, ebenfalls gesichert werden. (Das Sicherungsdienstprogramm kann online oder offline ausgeführt werden, und das Sicherungsimage kann sowohl von der Datenbank als auch vom Tabellenbereich erstellt werden.) Die folgende Beschreibung trifft nur auf Dateien zu, die über DATALINK-Spalten verbunden sind, deren Parameter RECOVERY auf den Wert YES gesetzt ist. (Dateien, auf die über DATALINK-Spalten mit RECOVERY=NO verwiesen wird, werden nicht gesichert.)

Wenn Dateien verbunden werden, werden sie von den Data Links-Servern zum asynchronen Kopieren auf einen Archivierungsserver wie TSM (Tivoli Storage Manager) oder auf Platte geplant. Wenn das Sicherungsprogramm ausgeführt wird, stellt DB2 sicher, dass alle Dateien, die zum Kopieren geplant sind, kopiert wurden. Zu Beginn des Sicherungsvorgangs kontaktiert DB2 alle Data Links-Server, die in der DB2-Konfigurationsdatei angegeben sind. Wenn mindestens einer der Data Links-Server für die Datenbank konfiguriert ist, wird die Sicherung Erfolg haben, selbst wenn einer der Data Links-Server nicht verfügbar ist. Bei einem Neustart des Data Links-Servers wird das Sicherungsverfahren auf diesem Data Links-Server vollständig beendet, bevor er wieder für die Datenbank verfügbar gemacht wird. Wenn ein Data Links-Server mindestens eine verbundene Datei hat und nicht aktiv ist oder während der Sicherungsoperation inaktiv wird, enthält das Sicherungsimage keine vollständigen DATALINK-Informationen. Die Sicherungsoperation wird erfolgreich ausgeführt. In der Protokolldatei für diese Sicherungsoperation wird im Kommentarfeld der ausgefallene DLM-Server angeben, oder die Anzahl der ausgefallen DLM-Server, falls mehr als vier DLM-Server betroffen waren. Nach der erfolgreichen Sicherung der DLMs erhält das Kommentarfeld wieder seinen vorherigen Wert.

Bevor der Data Links-Server als wieder verfügbar für die Datenbank markiert werden kann, müssen alle offenen Sicherungen erfolgreich beendet worden sein. (Dies geschieht automatisch über einen asynchronen Prozess.) Wenn eine Sicherungsoperation eingeleitet wird und bereits die doppelte Anzahl der in *num_db_backups* (siehe weiter unten) angegebenen Sicherungen auf Beendigung auf dem Data Links-Server wartet, schlägt die Sicherungsoperation fehl. Es sind erst dann wieder weitere Sicherungen möglich, nachdem dieser Data Links-Server erneut gestartet wurde und die offenen Sicherungen abgeschlossen wurden.

Wenn die Verbindung einer Datei aufgehoben wird, wird die Datei entweder gelöscht oder auf ihre vorigen Berechtigungen zurückgesetzt, je nachdem, welcher Wert für den Parameter ON UNLINK angegeben ist. Eine erfolgreiche Sicherungsoperation kann den Data Links-Server dazu veranlassen, die archivierten Versionen von Dateien auf dem Archivierungsserver (entweder Platte oder TSM; siehe „Garbage Collection" auf Seite 61) zu bereinigen. Der Datenbankkonfigurationsparameter *num_db_backups* gibt die Anzahl der DB2-Datenbanksicherungen an, bevor archivierte Versionen der Dateien, deren Verbindung aufgehoben wurde, entfernt werden. Weitere Informationen zu diesem Konfigurationsparameter finden Sie im Handbuch *Systemverwaltung: Optimierung*.

Wenn Dateien entfernt werden, deren Verbindung aufgehoben wurde, werden die Informationen zu den nicht mehr verbundenen Dateien aus den Registriertabellen des Data Links-Servers ebenfalls entfernt.

Auswählen einer Sicherungsmethode für DB2 Data Links Manager unter AIX

Zur Sicherung von Dateien, die sich auf einem Data Links-Server befinden, können Sie außer Disk Copy (Dienstprogramm zum Erstellen von Plattenkopien) und XBSA auch Tivoli Storage Manager (TSM) verwenden.

Gehen Sie wie folgt vor, um Tivoli Storage Manager als Archivierungsserver zu verwenden:

1. Installieren Sie Tivoli Storage Manager auf dem Data Links-Server. Weitere Informationen hierzu finden Sie in der Produktdokumentation von Tivoli Storage Manager.
2. Registrieren Sie die Client-Anwendung des Data Links-Servers mit dem TSM-Server. Weitere Informationen hierzu finden Sie in der Produktdokumentation von Tivoli Storage Manager.
3. Fügen Sie der Script-Datei `db2profile` oder `db2cshrc` von Data Links Manager Administrator die folgenden Umgebungsvariablen hinzu:

```
(für Bash-, Bourne- oder Korn-Shell)
export DSMI_DIR=/usr/tivoli/tsm/client/api/bin
export DSMI_CONFIG=$HOME/tsm/dsm.opt
export DSMI_LOG=$HOME/dldump
export PATH=$PATH:$DSMI_DIR

(für C-Shell)
setenv DSMI_DIR /usr/tivoli/tsm/client/api/bin
setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
setenv DSMI_LOG ${HOME}/dldump
setenv PATH=${PATH}:$DSMI_DIR
```

4. Stellen Sie sicher, dass die TSM-Systemoptionsdatei `dsm.sys` sich im Verzeichnis `$DSMI_DIR` befindet.

5. Stellen Sie sicher, dass die TSM-Benutzeroptionsdatei `dsm.opt` sich im Verzeichnis *INSTHOME*`/tsm` befindet, wobei *INSTHOME* das Ausgangsverzeichnis von Data Links Manager Administrator ist.
6. Setzen Sie die Option *PASSWORDACCESS* in der TSM-Systemoptionsdatei `/usr/tivoli/tsm/client/api/bin/dsm.sys` auf `generate`.
7. Registrieren Sie *vor* dem ersten Aufruf von Data Links File Manager das TSM-Kennwort mit der Option `generate`. Auf diese Weise müssen Sie kein Kennwort angeben, wenn Data Links File Manager eine Verbindung zum TSM-Server herstellt. Weitere Informationen finden Sie in Ihrer TSM-Produktdokumentation.
8. Setzen Sie die Registrierungsvariable DLFM_BACKUP_TARGET auf `TSM`. Der Wert der Registrierungsvariablen DLFM_BACKUP_DIR_NAME wird in diesem Fall ignoriert. Dies aktiviert die TSM-Sicherungsoption.

   Anmerkungen:

   a. Wenn Sie die Einstellung der Registrierungsvariablen DLFM_BACKUP_TARGET zur Laufzeit von TSM in Platte oder umgekehrt ändern, beachten Sie, dass die archivierten Dateien nicht an die neu angegebene Archivposition verschoben werden. Wenn Sie z. B. Data Links File Manager mit der auf den Wert TSM gesetzten Registrierungsvariablen DLFM_BACKUP_TARGET starten und den Wert danach in eine Plattenposition ändern, werden alle neu archivierten Dateien an dieser Position auf Platte gespeichert. Die zuvor auf TSM archivierten Dateien werden jedoch nicht an die neue Plattenposition verschoben.
   b. Zur Überschreibung der Standard-TSM-Verwaltungsklasse wird eine neue Registrierungsvariable bereitgestellt, DLFM_TSM_MGMTCLASS. Wenn diese Registrierungsvariable keinen Wert erhält, wird die Standard-TSM-Verwaltungsklasse verwendet.
9. Stoppen Sie Data Links File Manager, indem Sie den Befehl **dlfm stop** eingeben.
10. Starten Sie Data Links File Manager, indem Sie den Befehl **dlfm start** eingeben.

Auswählen einer Sicherungsmethode für DB2 Data Links Manager in der Solaris-Betriebsumgebung

Zur Sicherung von Dateien, die sich auf einem Data Links-Server befinden, können Sie außer Disk Copy (Dienstprogramm zum Erstellen von Plattenkopien) und XBSA auch Tivoli Storage Manager (TSM) verwenden.

Gehen Sie wie folgt vor, um Tivoli Storage Manager als Archivierungsserver zu verwenden:

1. Installieren Sie Tivoli Storage Manager auf dem Data Links-Server. Weitere Informationen hierzu finden Sie in der Produktdokumentation von Tivoli Storage Manager.

2. Registrieren Sie die Client-Anwendung des Data Links-Servers mit dem TSM-Server. Weitere Informationen hierzu finden Sie in der Produktdokumentation von Tivoli Storage Manager.
3. Fügen Sie der Script-Datei `db2profile` oder `db2cshrc` von Data Links Manager Administrator die folgenden Umgebungsvariablen hinzu:

```
(für Bash-, Bourne- oder Korn-Shell)
export DSMI_DIR=/opt/tivoli/tsm/client/api/bin
export DSMI_CONFIG=$HOME/tsm/dsm.opt
export DSMI_LOG=$HOME/dldump
export PATH=$PATH:/opt/tivoli/tsm/client/api/bin

(für C-Shell)
setenv DSMI_DIR /opt/tivoli/tsm/client/api/bin
setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
setenv DSMI_LOG ${HOME}/dldump
setenv PATH=${PATH}:/opt/tivoli/tsm/client/api/bin
```

4. Stellen Sie sicher, dass die TSM-Systemoptionsdatei `dsm.sys` sich im Verzeichnis `/opt/tivoli/tsm/client/api/bin` befindet.
5. Stellen Sie sicher, dass die TSM-Benutzeroptionsdatei `dsm.opt` sich im Verzeichnis *INSTHOME*`/tsm` befindet, wobei *INSTHOME* das Ausgangsverzeichnis von Data Links Manager Administrator ist.
6. Setzen Sie die Option *PASSWORDACCESS* in der TSM-Systemoptionsdatei `/usr/tivoli/tsm/client/api/bin/dsm.sys` auf `generate`.
7. Registrieren Sie *vor* dem ersten Aufruf von Data Links File Manager das TSM-Kennwort mit der Option `generate`. Auf diese Weise müssen Sie kein Kennwort angeben, wenn Data Links File Manager eine Verbindung zum TSM-Server herstellt. Weitere Informationen finden Sie in Ihrer TSM-Produktdokumentation.
8. Setzen Sie die Registrierungsvariable DLFM_BACKUP_TARGET auf `TSM`. Der Wert der Registrierungsvariablen DLFM_BACKUP_DIR_NAME wird in diesem Fall ignoriert. Dies aktiviert die TSM-Sicherungsoption.

   Anmerkungen:

   a. Wenn Sie die Einstellung der Registrierungsvariablen DLFM_BACKUP_TARGET zur Laufzeit von TSM in Platte oder umgekehrt ändern, beachten Sie, dass die archivierten Dateien nicht an die neu angegebene Archivposition verschoben werden. Wenn Sie z. B. Data Links File Manager mit der auf den Wert TSM gesetzten Registrierungsvariablen DLFM_BACKUP_TARGET starten und den Wert danach in eine Plattenposition ändern, werden alle neu archivierten Dateien an dieser Position auf Platte gespeichert. Die zuvor auf TSM archivierten Dateien werden jedoch nicht an die neue Plattenposition verschoben.

   b. Zur Überschreibung der Standard-TSM-Verwaltungsklasse wird eine neue Registrierungsvariable bereitgestellt, DLFM_TSM_MGMTCLASS. Wenn diese Registrierungsvariable keinen Wert erhält, wird die Standard-TSM-Verwaltungsklasse verwendet.

9. Stoppen Sie Data Links File Manager, indem Sie den Befehl **dlfm stop** eingeben.
10. Starten Sie Data Links File Manager, indem Sie den Befehl **dlfm start** eingeben.

Auswählen einer Sicherungsmethode für DB2 Data Links Manager unter Windows NT

Wenn ein DATALINK-Wert in eine Tabelle mit einer DATALINK-Spalte eingefügt wird, die für Wiederherstellung definiert ist, werden die zugehörigen DATALINK-Dateien auf dem Data Links-Server zu einem geplanten Zeitpunkt auf einem Archivierungsserver gesichert. Derzeit werden zum Sichern von Dateien auf einem Archivierungsserver die beiden Optionen Disk Copy (Standardmethode) und Tivoli Storage Manager unterstützt. Zukünftige Releases von DB2 Data Links Manager für Windows NT werden außerdem Unterstützung für entsprechende Datenträger und Software anderer Hersteller bieten.

Disk Copy (Standardmethode)

Wenn das Sicherungsdienstprogramm auf dem DB2-Server aufgerufen wird, stellt es sicher, dass die verbundenen Dateien in der Datenbank auf dem Data Links-Server in dem Verzeichnis gesichert werden, das von der Umgebungsvariable DLFM_BACKUP_DIR_NAME angegeben wird. Der Standardwert für diese Variable lautet `c:\dlfmbackup`, wobei `c:\` für das Sicherungsinstallationslaufwerk von Data Links Manager steht.

Um diese Variable auf den Wert c:\dlfmbackup zu setzen, geben Sie den folgenden Befehl ein:

```
db2set -g DLFM_BACKUP_DIR_NAME=c:\dlfmbackup
```

Die von der Umgebungsvariablen DLFM_BACKUP_DIR_NAME angegebene Position darf sich *nicht* auf einem Dateisystem befinden, das Data Links Filesystem Filter verwendet. Stellen Sie sicher, das in dem für die Sicherungsdateien angegebenen Verzeichnis genügend Platz vorhanden ist.

Stellen Sie ebenfalls sicher, dass die Variable DLFM_BACKUP_TARGET auf LOCAL gesetzt ist. Geben Sie dazu den folgenden Befehl ein:

```
db2set -g DLFM_BACKUP_TARGET=LOCAL
```

Nach Einstellung oder Änderung dieser Variablen stoppen Sie Data Links File Manager mit dem Befehl **dlfm stop**, und führen Sie mit dem Befehl **dlfm start** einen Neustart von DLFM aus.

Tivoli Storage Manager

Gehen Sie wie folgt vor, um Tivoli Storage Manager als Archivierungsserver zu verwenden:

1. Installieren Sie Tivoli Storage Manager auf dem Data Links-Server. Weitere Informationen hierzu finden Sie in der Produktdokumentation von Tivoli Storage Manager.
2. Registrieren Sie die Client-Anwendung des Data Links-Servers mit dem TSM-Server. Weitere Informationen hierzu finden Sie in der Produktdokumentation von Tivoli Storage Manager.
3. Klicken Sie auf **Start**, und wählen Sie **Einstellungen —> Systemsteuerung —> System** aus. Das Fenster **Systemsteuerung** wird geöffnet. Wählen Sie die Registerkarte **Umgebung** aus, und geben Sie die folgenden Umgebungsvariablen und entsprechenden Werte ein:

Variable	Wert
DSMI_DIR	c:\tsm\baclient
DSMI_CONFIG	c:\tsm\baclient\dsm.opt
DSMI_LOG	c:\tsm\dldump

4. Stellen Sie sicher, dass die TSM-Systemoptionsdatei `dsm.sys` sich im Verzeichnis `c:\tsm\baclient` befindet.
5. Stellen Sie sicher, dass die TSM-Benutzeroptionsdatei `dsm.opt` sich im Verzeichnis `c:\tsm\baclient` befindet.
6. Setzen Sie die Option *PASSWORDACCESS* in der TSM-Systemoptionsdatei `/usr/tivoli/tsm/client/api/bin/dsm.sys` auf `generate`.
7. Registrieren Sie *vor* dem ersten Aufruf von Data Links File Manager das TSM-Kennwort mit der Option `generate`. Auf diese Weise müssen Sie kein Kennwort angeben, wenn Data Links File Manager eine Verbindung zum TSM-Server herstellt. Weitere Informationen finden Sie in Ihrer TSM-Produktdokumentation.
8. Setzen Sie die Umgebungsvariable DLFM_BACKUP_TARGET mit dem folgenden Befehl auf `TSM`:

   ```
   db2set -g DLFM_BACKUP_TARGET=TSM
   ```

   Der Wert der Umgebungsvariablen DLFM_BACKUP_DIR_NAME wird in diesem Fall ignoriert. Dies aktiviert die TSM-Sicherungsoption.

   Anmerkungen:

   a. Wenn Sie die Einstellung der Umgebungsvariablen DLFM_BACKUP_TARGET zur Laufzeit von TSM in LOCAL oder umgekehrt ändern, beachten Sie, dass die archivierten Dateien nicht an die neu angegebene Archivposition verschoben wer-

den. Wenn Sie z. B. Data Links File Manager mit der auf den Wert TSM gesetzten Umgebungsvariablen DLFM_BACKUP-_TARGET starten und den Wert danach in LOCAL ändern, werden alle neu archivierten Dateien an dieser Position auf der Platte gespeichert. Die zuvor auf TSM archivierten Dateien werden jedoch nicht an die neue Plattenposition verschoben.

b. Zur Überschreibung der Standard-TSM-Verwaltungsklasse wird eine neue Umgebungsvariable bereitgestellt, DLFM_TSM_MGMTCLASS. Wenn diese Variable keinen Wert erhält, wird die Standard-TSM-Verwaltungsklasse verwendet.

9. Stoppen Sie Data Links File Manager, indem Sie den Befehl **dlfm stop** eingeben.
10. Starten Sie Data Links File Manager, indem Sie den Befehl **dlfm start** eingeben.

Sichern eines Journaled File System unter AIX

Sie können eine Offlinesicherung eines Journaled File System unter AIX ausführen, nachdem Data Links Manager gestoppt wurde. Die nachstehend beschriebene Vorgehensweise, bei der Data Links Manager aktiv bleiben kann, wird für Benutzer empfohlen, die eine höhere Verfügbarkeit benötigen.

1. Greifen Sie auf die CLI-Quellendatei `quiesce.c` und das Shell-Script `online.sh` zu. Diese Dateien befinden sich im Verzeichnis `/samples/dlfm`.
2. Kompilieren Sie die Datei `quiesce.c`:

```
xlC -o quiesce  -L$HOME/sqllib/lib -I$HOME/sqllib/include -c quiesce.c
```

3. Führen Sie als Root das Script für den Knoten aus, auf dem sich das DLFS-Dateisystem befindet.

Das Shell-Script `online.sh` nimmt an, dass auf dem Data Link Manager-Knoten für jede bei Data Link Manager registrierte Datenbank ein Katalogeintrag vorhanden ist. Es nimmt weiter an, dass sich unter `/etc/filesystems` der vollständige Eintrag für das DLFS-Dateisystem befindet. Das Shell-Script führt nun folgende Aktionen aus:

- Versetzen aller, bei Data Links Manager registrierten Datenbanktabellen in den Wartemodus. Dies stoppt alle neuen Aktivitäten.
- Abhängen des Dateisystems und erneutes Anhängen als schreibgeschütztes Dateisystem.
- Erstellen einer Dateisystemsicherung.
- Abhängen des Dateisystems und erneutes Anhängen als Dateisystem mit Lese-/Schreibzugriff.
- Zurücksetzen der DB2-Tabellen, d. h. Aufheben des internen Wartestatus.

Das Script muss für Ihre Umgebung anhand folgender Schritte angepasst werden:

1. Wählen Sie den Befehl BACKUP aus, und fügen Sie ihn in die Funktion do_backup des Scripts ein.
2. Legen Sie innerhalb des Scripts die Umgebungsvariable fest:
   - DLFM_INST: Setzen Sie diese Variable auf den Namen des DLFM-Exemplars.
   - PATH_OF_EXEC: Setzen Sie diese Variable auf den Pfad, in dem sich die ausführbare Datei für "quiesce" (Wartemodus) befindet.

Rufen Sie das Script wie folgt auf:

```
online.sh <dateisystemname>
```

Überlegungen zu den Dienstprogrammen für Wiederherstellung und aktualisierende Wiederherstellung

Die folgenden Informationen beziehen sich auf den Fall, dass Sie eine DATALINK-Spalte (bzw. Spalten) haben, die mit der Option RECOVERY=YES für eine Tabelle definiert ist (bzw. sind). Wenn eine Tabelle eine DATALINK-Spalte enthält, für die die Option RECOVERY=NO definiert ist, wird die Tabelle am Ende der Wiederherstellungsoperation in den Status *DRP (Datalink Reconcile Pending)* versetzt. Weitere Informationen finden Sie in „Abstimmen von Data Links" auf Seite 92.

Während der Wiederherstellungsoperation können Tabellen mit DATALINK-Spalten in einen der beiden folgenden Status versetzt werden:

- *DRNP (Datalink Reconcile Not Possible)*

  Wenn eine Tabelle in den Status *DRNP (Datalink Reconcile Not Possible)* versetzt wurde, ist sie für uneingeschränkte Operationen an allen Spalten außer den DATALINK-Spalten verfügbar. Wenn eine DATALINK-Spalte in einer SELECT-Anweisung enthalten ist, wird eine Warnung ausgegeben. Sie können UPDATE-Aufrufe für DATALINK-Spalten absetzen (mit einigen Einschränkungen: Einzelheiten finden Sie in „Entfernen einer Tabelle aus dem Status "DRNP (Datalink Reconcile Not Possible)"" auf Seite 91). Sie können keine INSERT- und DELETE-Anweisungen absetzen, da sie die DATALINK-Spalte betreffen.

- *DRP (Datalink Reconcile Pending)*

  Wenn eine Tabelle in den Status *DRP (Datalink Reconcile Pending)* versetzt wurde, ist sie für uneingeschränkte Operationen an allen Spalten außer den DATALINK-Spalten verfügbar. Wenn eine DATALINK-Spalte in einer SELECT-Anweisung enthalten ist, wird eine Warnung ausgegeben. Sie können jedoch keine DML-Anweisungen wie UPDATE, INSERT oder DELETE absetzen.

Diese Status werden in der Datei `db2diag.log` aufgelistet, wenn die Dienstprogramme zur Wiederherstellung und aktualisierenden Wiederherstellung ausgeführt werden. Sie können diese Informationen auch über den Befehl **db2dart** abrufen.

Wenn Sie eine Datenbank bzw. einen Tabellenbereich wiederherstellen wollen, müssen für eine erfolgreiche Wiederherstellungsoperation die folgenden Bedingungen erfüllt sein:

- Wenn einer der in der Sicherungsdatei angegebenen Data Links-Server inaktiv ist, wird die Wiederherstellungsoperation dennoch erfolgreich beendet.

  Tabellen mit DATALINK-Spalteninformationen, auf die sich das Fehlen eines Data Links-Servers auswirkt, werden nach der (ggf. aktualisierenden) Wiederherstellungsoperation in den Status *DRP (Datalink Reconcile Pending)* versetzt. Bevor die Data Links-Server als wieder verfügbar für die Datenbank markiert werden können, muss dieser Wiederherstellungsprozess erfolgreich beendet werden. Dies übernimmt der asynchrone Prozess, der bei Verfügbarkeit des DLM das Sicherungsverfahren abschließt (siehe „Überlegungen zum Sicherungsdienstprogramm" auf Seite 70).
- Wenn einer der in der Sicherungsdatei angegebenen Data Links-Server während der Wiederherstellungsoperation ausfällt bzw. stoppt, schlägt die Wiederherstellungsoperation fehl. Die Wiederherstellung kann mit dem inaktiven Data Links-Server erneut gestartet werden (siehe oben).
- Wenn eine vorherige Operation zur Datenbankwiederherstellung auf einem der Data Links-Server noch nicht abgeschlossen ist, schlagen nachfolgende Wiederherstellungsoperationen für Datenbanken oder Tabellenbereiche fehl, bis die entsprechenden Data Links-Server erneut gestartet und die unvollständigen Wiederherstellungen abgeschlossen werden.
- Informationen zu allen DATALINK-Spalten, die in der Sicherungsdatei eingetragen sind, müssen in den entsprechenden Registriertabellen des Data Links-Servers vorhanden sein.

  Wenn nicht alle Informationen zu DATALINK-Spalten in den Registriertabellen eingetragen sind, wird die Tabelle mit den fehlenden Informationen zu einer DATALINK-Spalte nach dem Ende der Wiederherstellungsoperation (bzw. der Operation zur aktualisierenden Wiederherstellung, falls verwendet) in den Status *DRNP (Datalink Reconcile Not Possible)* versetzt.

  Wenn die Sicherung nicht in den Registriertabellen aufgezeichnet ist, ist die bereitgestellte Sicherungsdatei möglicherweise eine frühere Kopie als durch den Wert für *num_db_backups* angegeben und wurde somit bereits mit Garbage Collection bearbeitet. Dies bedeutet, dass die archivierten Dateien aus dieser früheren Sicherung entfernt wurden und nicht wiederhergestellt werden können. Alle Tabellen mit DATALINK-Spalten werden in den Status *DRP (Datalink Reconcile Pending)* versetzt.

  Wenn die Sicherung nicht in den Registriertabellen aufgezeichnet ist, kann dies bedeuten, dass das Sicherungsverfahren noch nicht abgeschlossen

wurde, weil der Data Links-Server nicht aktiv ist. Alle Tabellen mit DATALINK-Spalten werden in den Status *DRP (Datalink Reconcile Pending)* versetzt. Bei einem Neustart des Data Links-Servers wird das Sicherungsverfahren auf diesem Data Links-Server vor der Wiederherstellung vollständig beendet.

Die Tabelle bleibt für Benutzer verfügbar, jedoch geben die Werte in den DATALINK-Spalten eventuell die Dateien nicht korrekt an (z. B. wird vielleicht keine Datei gefunden, die dem Wert für die DATALINK-Spalte entspricht). Wenn Sie dies nicht wünschen, können Sie die Tabelle in den Status *Überprüfung anstehend* versetzen, indem Sie die Anweisung "SET CONSTRAINTS FOR tabellenname TO DATALINK RECONCILE PENDING" absetzen.

Wenn eine Tabelle sich nach einer Wiederherstellungsoperation im Status *DRNP (Datalink Reconcile Not Possible)* befindet, können Sie die Daten der DATALINK-Spalten mit einer der in „Entfernen einer Tabelle aus dem Status "DRNP (Datalink Reconcile Not Possible)"" auf Seite 91 beschriebenen Methoden berichtigen.

Anmerkung: Beim Markieren einer Datei vom unverbundenen Status in den verbundenen Status muss diese Datei eventuell von einem Archivierungsserver auf das Dateisystem abgerufen werden. Wenn während dieses Prozesses ein Fehler auftritt (wenn z. B., eine Datei wegen eines Dateinamenskonflikts nicht in das Dateisystem kopiert werden kann), wird die entsprechende Tabelle in den Status *DRP (Datalink Reconcile Pending)* versetzt.

Es wird dringend empfohlen, die Datei `datalink.cfg` zu archivieren, um auf ungewöhnliche Wiederherstellungssituationen vorbereitet zu sein, da die im Sicherungsimage der Datenbank enthaltene Datei `datalink.cfg` nur eine Fassung der Datei zum Zeitpunkt der Sicherung darstellt. Um alle Wiederherstellungssituationen abzudecken, müssen Sie immer über eine Kopie der aktuellsten Fassung von `datalink.cfg` verfügen. Aus diesem Grund müssen Sie nach jedem Aufruf des Befehls ADD DATALINKS MANAGER oder DROP DATALINKS MANAGER eine neue Sicherungskopie der Datei `datalink.cfg` erstellen. So könnten Sie jederzeit die neueste Fassung von `datalink.cfg` abrufen, wenn die Datei `datalink.cfg` nicht auf Platte verfügbar ist.

Wenn die aktuellste Fassung von `datalink.cfg` auf Platte nicht zur Verfügung steht, ersetzen Sie die (aus einem Sicherungsimage wiederhergestellte) Datei `datalink.cfg` durch die Datei `datalink.cfg`, die vor einer aktualisierenden Wiederherstellung archiviert wurde. Führen Sie dies nach der Wiederherstellung der Datenbank aus.

Wiederherstellen von Datenbanken von einer Offlinesicherung ohne aktualisierendes Wiederherstellen

Sie können nur auf der Datenbankebene, nicht aber auf der Tabellenbereichsebene eine Wiederherstellung ohne aktualisierende Wiederherstellung ausführen. Zur Wiederherstellung einer Datenbank ohne aktualisierende Wiederherstellung können Sie entweder eine nicht wiederherstellbare Datenbank (d. h. eine Datenbank, für die die Umlaufprotokollierung aktiv ist) wiederherstellen oder für den Befehl RESTORE DATABASE den Parameter WITHOUT ROLLING FORWARD angeben.

Wenn Sie das Dienstprogramm RESTORE mit der Option WITHOUT DATALINK verwenden, werden alle Tabellen mit DATALINK-Spalten in den Status *DRP (Datalink Reconcile Pending)* versetzt, und es wird während der Wiederherstellung keine Abstimmung mit den Data Links-Servern durchgeführt.

Wenn Sie die Option WITHOUT DATALINK nicht verwenden und einer der in der Sicherungsdatei aufgeführten Data Links-Server nicht mehr für die Datenbank definiert ist (d. h., er wurde mit dem Befehl DROP DATALINKS MANAGER gelöscht), werden Tabellen, die DATALINK-Daten mit Verweisen auf den gelöschten Data Links-Server enthalten, vom Wiederherstellungsdienstprogramm in den Status *DRP (Datalink Reconcile Pending)* versetzt.

Wenn Sie die Option WITHOUT DATALINK nicht verwenden, sämtliche Data Links-Server verfügbar sind und sämtliche Informationen zu den DATALINK-Spalten vollständig in den Registriertabellen aufgezeichnet sind, wird für jeden in der Sicherungsdatei erfassten Data Links-Server Folgendes ausgeführt:

- Alle Dateien, die nach Erstellung des Sicherungsimage verbunden wurden, das für die Wiederherstellung der Datenbank verwendet wurde, werden als nicht verbunden markiert (weil sie im Sicherungsimage nicht als verbunden aufgezeichnet sind).
- Alle Dateien, deren Verbindung nach Erstellung des Sicherungsimage aufgehoben wurde, die jedoch vor der Erstellung des Sicherungsimage verbunden waren, werden als verbunden markiert (weil sie im Sicherungsimage als verbunden aufgezeichnet sind). Wenn die Datei nachfolgend mit einer anderen Tabelle in einer anderen Datenbank verbunden wurde, wird die wiederhergestellte Tabelle in den Status *DRP (Datalink Reconcile Pending)* versetzt.

Anmerkung: Die oben aufgeführten Aktionen können nicht ausgeführt werden, wenn bei Erstellung des für die Datenbankwiederherstellung verwendeten Sicherungsimage mindestens ein Data Links-Server inaktiv war, da die in der Sicherung enthaltenen DATALINK-Informationen unvollständig sind. Die aufgeführten Aktionen können ebenfalls nicht ausgeführt werden, wenn das

bei der Wiederherstellungsoperation für die Datenbank verwendete Sicherungsimage nach einer Datenbankwiederherstellung (mit oder ohne aktualisierende Wiederherstellung) erstellt wurde. In beiden Fällen werden alle Tabellen mit DATALINK-Spalten in den Status *DRP (Datalink Reconcile Pending)* versetzt, und es wird während der Wiederherstellungsoperation keine Abstimmung mit den Data Links-Servern durchgeführt.

Wiederherstellen von Datenbanken und Tabellenbereichen und aktualisierendes Wiederherstellen bis zum Ende der Protokolle

Wenn Sie die Datenbank bzw. den Tabellenbereich wiederherstellen und anschließend bis zum Ende der Protokolle aktualisierend wiederherstellen (d. h., alle Protokolle stehen zur Verfügung), ist nur dann eine Abstimmungsüberprüfung erforderlich, wenn mindestens einer der in der Sicherungsdatei aufgeführten Data Links-Server während der Wiederherstellungsoperation inaktiv ist. Wenn Sie nicht sicher sind, ob alle Protokolle für die aktualisierende Wiederherstellung verfügbar sind, oder annehmen, dass Sie eventuell DATALINK-Werte abstimmen müssen, gehen Sie wie folgt vor:

1. Setzen Sie folgende SQL-Anweisung für die betroffenen Tabellen ab:

   ```
   SET CONSTRAINTS FOR tabellenname TO DATALINK RECONCILE PENDING
   ```

   Dadurch erhalten die Tabellen gleichzeitig die beiden Status *DRP (Datalink Reconcile Pending)* und *Überprüfung anstehend*.
2. Wenn Sie für eine der Tabellen den Status *Überprüfung anstehend* wieder aufheben wollen, setzen Sie die folgende SQL-Anweisung ab:

   ```
   SET CONSTRAINTS FOR tabellenname IMMEDIATE CHECKED
   ```

   Dadurch wird die Tabelle aus dem Status *Überprüfung anstehend* herausgenommen, aber im Status *DRP (Datalink Reconcile Pending)* belassen. Sie müssen das Dienstprogramm RECONCILE zum Abstimmen verwenden, um die Tabelle aus diesem Status herauszunehmen.

Es kann vorkommen, dass die Sicherungsdatei DATALINK-Daten enthält, die auf ein Exemplar von DB2 Data Links Manager verweisen (d. h., DB2 Data Links Manager wurde für die Datenbank registriert, als die Sicherung erstellt wurde), das aus der Datenbank gelöscht wurde. Wenn in aktualisierend wiederhergestellten Tabellenbereichen mindestens eine Tabelle DATALINK-Daten enthält, die auf den gelöschten Data Links-Server verweisen, versetzt das Dienstprogramm zur aktualisierenden Wiederherstellung alle Tabellen mit DATALINK-Daten in den den Status *DRP (Datalink Reconcile Pending)*.

Wiederherstellen von Datenbanken und Tabellenbereichen und aktualisierendes Wiederherstellen bis zu einem bestimmten Zeitpunkt

Bei Verwendung von Data Links-Tabellen können Sie bis zum Ende der Protokolle oder bis zu einem angegebenen Zeitpunkt aktualisierend wiederherstellen.

Tabellen in Tabellenbereichen, die bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt werden, werden am Ende der aktualisierenden Wiederherstellung in den Status *DRP (Datalink Reconcile Pending)* versetzt. Sie sollten das Dienstprogramm zur Abstimmung verwenden, um sie aus diesem Status herauszunehmen. Weitere Informationen finden Sie in „Abstimmen von Data Links" auf Seite 92.

Beispiel für eine aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt

Im Folgenden wird ein einfaches Szenario beschrieben, das Dateien zeigt, die behalten werden müssen, um Sicherung und Wiederherstellung durchführen zu können. Das Beispiel zeigt Änderungen am Wert einer einzelnen Zeile in einer Spalte des Typs DATALINK zusammen mit den Dateien, die DB2 Data Links Manager zur Unterstützung der Wiederherstellung behalten muss. Für dieses Beispiel wird angenommen, dass es keine Unterstützung für eine aktualisierende Wiederherstellung dieser Dateien bis zu einem Zeitpunkt gibt, der vor der letzten Sicherung liegt. Data Links-Server, die DB2 Data Links Manager ausführen, unterliegen einer solchen Einschränkung nicht. Beachten Sie, dass DateiA bis Zeit 3 existiert, zu der sie gelöscht wird, weil die Verbindung zur Zeit 2 aufgehoben wurde, und für die Datenbank in diesem Beispiel gilt, dass eine nicht mehr verbundene Datei bis zur Durchführung der nächsten Sicherung behalten wird (d. h., der Konfigurationsparameter *num_db_backups* der Datenbank hat den Wert 1).

Zeit	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Aktivität	Erstellen	Aktualisieren	Sichern	Aktualisieren	Aktualisieren	Löschen	Wiederherstellen bis 5
Spaltenwert	WertA	WertB	WertB	WertC	WertD	-	WertD
Verbundene Datei	DateiA	DateiB	DateiB	DateiC	DateiD	-	DateiD

Zeit	1	2	3	4	5	6	7
Von Data Links File Manager behaltene Extra-dateien		DateiA		DateiB	DateiB, DateiC	DateiB, DateiC, DateiD	DateiB, DateiC

Anmerkung: Die Wiederherstellung verbundener Dateien erfolgt immer in Verbindung mit dem Rest der Datenbank.

DB2 Data Links Manager und Interaktionen bei Wiederherstellung

Die folgende Tabelle zeigt die verschiedenen Arten von Wiederherstellung, die Sie durchführen können, die Verarbeitung in DB2 Data Links Manager, die bei Wiederherstellungsoperationen und Operationen zur aktualisierenden Wiederherstellung auftritt, und ob Sie nach Beendigung der Wiederherstellung das Dienstprogramm RECONCILE zur Abstimmung ausführen müssen:

Art der Wiederherstellung	Verarbeitung in DB2 Data Links Manager während Wiederherstellung	Verarbeitung in DB2 Data Links Manager während aktualisierender Wiederherstellung	Abstimmung mit RECONCILE
Nicht wiederherstellbare Datenbank			
Datenbank-wiederherstellung aus Gesamt-sicherung, alle Data Links-Server aktiv	Schnelle Abstim-mung wird durch-geführt	N/V	Kann optional aus-geführt werden, wenn ein Problem mit Datei-verbindungen (Links) angenom-men wird.
Datenbank-wiederherstellung mit Option WITHOUT DATALINK	Tabellen erhalten den Status *DRP (Datalink Reconcile Pending)*	N/V	Erforderlich

Art der Wiederherstellung	Verarbeitung in DB2 Data Links Manager während Wiederherstellung	Verarbeitung in DB2 Data Links Manager während aktualisierender Wiederherstellung	Abstimmung mit RECONCILE
Datenbankwiederherstellung aus Gesamtsicherung, mindestens ein Data Links-Server inaktiv	Schnelle Abstimmung wird nur für Tabellen in Tabellenbereichen ausgeführt, die keine Verbindung zu dem inaktiven Data Links-Server haben, die übrigen Tabellen erhalten den Status *DRP (Datalink Reconcile Pending)*	N/V	Erforderlich für Tabellen in Tabellenbereichen, die Verbindung zum inaktiven Data Links-Server haben
Datenbankwiederherstellung aus unvollständiger Sicherung, alle Data Links-Server aktiv	Keine schnelle Abstimmung, alle Tabellen mit DATALINK-Spalten erhalten den Status *DRP (Datalink Reconcile Pending)*	N/V	Erforderlich
Wiederherstellbare Datenbank			
Datenbankwiederherstellung mit Option WITHOUT ROLLING FORWARD, aus Gesamtsicherung, alle Data Links-Server aktiv	Schnelle Abstimmung wird durchgeführt	N/V	Optional

Art der Wiederherstellung	Verarbeitung in DB2 Data Links Manager während Wiederherstellung	Verarbeitung in DB2 Data Links Manager während aktualisierender Wiederherstellung	Abstimmung mit RECONCILE
Datenbankwiederherstellung mit Optionen WITHOUT ROLLING FORWARD und WITHOUT DATALINK, aus Gesamtsicherung oder unvollständiger Sicherung, Data Links-Server aktiv oder inaktiv	Tabellen erhalten den Status *DRP (Datalink Reconcile Pending)*	N/V	Erforderlich
Datenbankwiederherstellung mit Option WITHOUT ROLLING FORWARD, aus Gesamtsicherung, mindestens ein Data Links-Server inaktiv	Schnelle Abstimmung wird nur für Tabellen in Tabellenbereichen ausgeführt, die keine Verbindung zu den inaktiven Data Links-Servern haben, die übrigen Tabellen erhalten den Status *DRP (Datalink Reconcile Pending)*	N/V	Erforderlich für Tabellen in Tabellenbereichen, die Verbindungen zu den inaktiven Data Links-Servern haben
Datenbankwiederherstellung mit Option WITHOUT ROLLING FORWARD, aus unvollständiger Sicherung, Data Links-Server aktiv oder inaktiv	Keine schnelle Abstimmung, alle Tabellen mit DATALINK-Spalten erhalten den Status *DRP (Datalink Reconcile Pending)*	N/V	Erforderlich

Art der Wiederherstellung	Verarbeitung in DB2 Data Links Manager während Wiederherstellung	Verarbeitung in DB2 Data Links Manager während aktualisierender Wiederherstellung	Abstimmung mit RECONCILE
Datenbankwiederherstellung und aktualisierende Wiederherstellung bis zum Ende der Protokolle, aus Gesamtsicherung, alle Data Links-Server aktiv	Keine Aktion	Keine Aktion	Optional
Datenbankwiederherstellung und aktualisierende Wiederherstellung bis zum Ende der Protokolle, aus Gesamtsicherung, mindestens ein Data Links-Server inaktiv während aktualisierender Wiederherstellung	Keine Aktion	Keine Aktion	Optional
Datenbankwiederherstellung und aktualisierende Wiederherstellung bis zum Ende der Protokolle, aus Gesamtsicherung oder unvollständiger Sicherung, beliebige Data Links-Server inaktiv während Wiederherstellung	Keine Aktion	Alle Tabellen mit DATALINK-Spalten erhalten den Status *DRP (Datalink Reconcile Pending)*	Erforderlich für alle Tabellen mit DATALINK-Spalten

Art der Wiederherstellung	Verarbeitung in DB2 Data Links Manager während Wiederherstellung	Verarbeitung in DB2 Data Links Manager während aktualisierender Wiederherstellung	Abstimmung mit RECONCILE
Datenbankwiederherstellung und aktualisierende Wiederherstellung bis zum Ende der Protokolle, aus unvollständiger Sicherung, alle Data Links-Server aktiv während Wiederherstellung	Keine Aktion	Keine Aktion	Optional
Datenbankwiederherstellung und aktualisierende Wiederherstellung bis zum Ende der Protokolle, aus Gesamtsicherung oder unvollständiger Sicherung, alle Data Links-Server aktiv während Wiederherstellung, Sicherungsimage auf allen Data Links-Servern unbekannt	Keine Aktion	Alle Tabellen in Tabellenbereichen mit Verbindungen zu einem Data Links-Server, auf dem die Sicherung unbekannt ist, erhalten den Status *DRP (Datalink Reconcile Pending)*	Erforderlich
Tabellenbereichswiederherstellung und aktualisierende Wiederherstellung bis zum Ende der Protokolle, aus Gesamtsicherung, alle Data Links-Server aktiv	Keine Aktion	Keine Aktion	Optional

Art der Wiederherstellung	Verarbeitung in DB2 Data Links Manager während Wiederherstellung	Verarbeitung in DB2 Data Links Manager während aktualisierender Wiederherstellung	Abstimmung mit RECONCILE
Tabellenbereichswiederherstellung und aktualisierende Wiederherstellung bis zum Ende der Protokolle, aus Gesamtsicherung, mindestens ein Data Links-Server inaktiv während aktualisierender Wiederherstellung	Keine Aktion	Keine Aktion	Optional
Tabellenbereichswiederherstellung und aktualisierende Wiederherstellung bis zum Ende der Protokolle, aus Gesamtsicherung oder unvollständiger Sicherung, beliebige Data Links-Server inaktiv während Wiederherstellung	Keine Aktion	Alle Tabellen in Tabellenbereichen mit Verbindungen zu einem inaktiven Server erhalten den Status *DRP (Datalink Reconcile Pending)*	Erforderlich für Tabellen in Tabellenbereichen, die Verbindung zu inaktiven Data Links-Servern haben
Tabellenbereichswiederherstellung und aktualisierende Wiederherstellung bis zum Ende der Protokolle, aus unvollständiger Sicherung, alle Data Links-Server aktiv	Keine Aktion	Keine Aktion	Optional

Art der Wiederherstellung	Verarbeitung in DB2 Data Links Manager während Wiederherstellung	Verarbeitung in DB2 Data Links Manager während aktualisierender Wiederherstellung	Abstimmung mit RECONCILE
Datenbankwiederherstellung und aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt, aus Gesamtsicherung oder unvollständiger Sicherung, Data Links-Server aktiv oder inaktiv während Wiederherstellung und/oder aktualisierender Wiederherstellung	Keine Aktion	Tabellen erhalten den Status *DRP (Datalink Reconcile Pending)*	Erforderlich
Tabellenbereichswiederherstellung und aktualisierende Wiederherstellung bis zu einem bestimmten Zeitpunkt, aus Gesamtsicherung oder unvollständiger Sicherung, Data Links-Server aktiv oder inaktiv während Wiederherstellung und/oder aktualisierender Wiederherstellung	Keine Aktion	Tabellen erhalten den Status *DRP (Datalink Reconcile Pending)*	Erforderlich
Datenbankwiederherstellung mit anderem Datenbanknamen, Aliasnamen, Hostnamen oder Exemplar ohne aktualisierende Wiederherstellung (siehe Anmerkung 1 auf Seite 91)	Tabellen erhalten den Status *DRNP (Datalink Reconcile Not Possible)*	N/V	Optional, aber Tabellen mit Status *DRNP (Datalink Reconcile Not Possible)* müssen manuell repariert werden

Art der Wiederherstellung	Verarbeitung in DB2 Data Links Manager während Wiederherstellung	Verarbeitung in DB2 Data Links Manager während aktualisierender Wiederherstellung	Abstimmung mit RECONCILE
Datenbankwiederherstellung mit anderem Datenbanknamen, Aliasnamen, Hostnamen oder Exemplar und aktualisierende Wiederherstellung	Keine Aktion	Tabellen erhalten den Status *DRNP (Datalink Reconcile Not Possible)*	Optional, aber Tabellen mit Status *DRNP (Datalink Reconcile Not Possible)* müssen manuell repariert werden
Datenbankwiederherstellung aus unbrauchbarer Sicherung (Garbage Collection wurde auf Data Links-Server für Image ausgeführt), ohne aktualisierende Wiederherstellung (siehe Anmerkung 1 auf Seite 91), mit oder ohne Option WITHOUT DATALINK	Tabellen erhalten den Status *DRP (Datalink Reconcile Pending)*	Keine Aktion	Erforderlich
Datenbankwiederherstellung aus unbrauchbarer Sicherung (Garbage Collection wurde auf Data Links-Server für Image ausgeführt), mit aktualisierender Wiederherstellung, mit oder ohne Option WITHOUT DATALINK	Keine Aktion	Tabellen erhalten den Status *DRP (Datalink Reconcile Pending)*	Erforderlich

Art der Wiederherstellung	Verarbeitung in DB2 Data Links Manager während Wiederherstellung	Verarbeitung in DB2 Data Links Manager während aktualisierender Wiederherstellung	Abstimmung mit RECONCILE
Tabellenbereichswiederherstellung aus unbrauchbarer Sicherung (Garbage Collection wurde auf Data Links-Server für Image ausgeführt), mit aktualisierender Wiederherstellung	Keine Aktion	Tabellen erhalten den Status *DRP (Datalink Reconcile Pending)*	Erforderlich

Anmerkungen:

1. Eine Wiederherstellungsoperation, bei der ein Onlinesicherungsimage und die Option WITHOUT ROLLING FORWARD verwendet werden, oder eine Wiederherstellungsoperation, bei der ein Offlinesicherungsimage verwendet wird.
2. Eine *Gesamtsicherung* ist eine Sicherung, bei deren Erstellung alle erforderlichen Data Links-Server aktiv waren. Eine *unvollständige* Sicherung ist eine Sicherung, bei deren Erstellung mindestens ein erforderlicher Data Links-Server inaktiv war.
3. Schnelle Abstimmung wird nicht ausgeführt, wenn das bei der Wiederherstellung der Datenbank verwendete Sicherungsimage nach einer Datenbankwiederherstellung (mit oder ohne aktualisierende Wiederherstellung) erstellt wurde. In diesem Fall erhalten alle Tabellen mit DATALINK-Spalten den Status *DRP (Datalink Reconcile Pending)*.

Entfernen einer Tabelle aus dem Status "DRNP (Datalink Reconcile Not Possible)"

Wiederhergestellte Tabellen, die DATALINK-Spalten enthalten, werden in den Status *DRNP (Datalink Reconcile Not Possible)* versetzt, wenn ein Tabellenbereich von einer Sicherung wiederhergestellt wird, die zeitlich vor dem Wert liegt, der für den Datenbankkonfigurationsparameter *num_db_backups* angegeben ist. Weitere Informationen zu diesem Konfigurationsparameter finden Sie im Handbuch *Systemverwaltung: Optimierung*.

DB2 ermöglicht den Zugriff auf die Tabelle, auch wenn die Werte der DATALINK-Spalte vielleicht ungültig sind. Wenn Sie den Zugriff auf eine Tabelle mit möglicherweise inkonsistenten Werten in der DATALINK-Spalte verhindern wollen, setzen Sie den Befehl SET CONSTRAINTS FOR *tabellenname* TO DATALINK RECONCILE PENDING ab. Gehen Sie wie folgt vor, um die DATALINK-Werte zu aktualisieren:

- Setzen Sie mit Hilfe der SQL-Anweisung UPDATE den Teil für die Datenposition eines DATALINK-Spaltenwerts auf eine URL-Adresse der Länge null, wenn die Spalte keine Nullwerte enthalten darf, bzw. auf NULL, wenn die Spalte Nullwerte enthalten darf.
- Stellen Sie die Datei auf den entsprechenden Data Links-Servern wieder her. Führen Sie anschließend eine Anwendung aus, die SELECT-Anweisungen zum Lesen der DATALINK-Spaltenwerte und UPDATE-Anweisungen zum Aktualisieren der DATALINK-Spalten mit denselben Werten absetzt. Beachten Sie, dass beim Aktualisieren der DATALINK-Spaltenwerte der Status *DRNP (Datalink Reconcile Not Possible)* aktiv sein muss. Nach Beendigung der UPDATE-Operationen werden die Dateien auf den entsprechenden Data Links-Servern als verbunden markiert.

Anschließen setzen Sie den Status *DRNP (Datalink Reconcile Not Possible)* zurück, indem Sie den folgenden Befehl absetzen:

```
SET CONSTRAINTS FOR tabellenname DATALINK RECONCILE PENDING IMMEDIATE UNCHECKED
```

Abstimmen von Data Links

Zur Abstimmung von Data Links wird das Dienstprogramm RECONCILE verwendet. Dieses Dienstprogramm wird von DB2 aus eingeleitet und betrifft alle Data Links-Server, die DB2 Data Links Manager ausführen und auf die durch DATALINK-Spaltenwerte verwiesen wird. Es überprüft entweder, ob die angegebenen Dateien auf dem Data Links-Server vorhanden sind oder ob die Verbindungen (Links) wiederhergestellt werden können. In den folgenden Abschnitten wird beschrieben, wie DB2 erkennt, ob Sie Data Links abstimmen müssen und wie Sie sie abstimmen.

Wenn eine Data Links-Serverdateireferenz nicht vorhanden ist oder nicht erneut hergestellt werden kann, stellt das Dienstprogramm RECONCILE eine Kopie der fehlerhaften Zeilen zusammen mit einer Ursache für den jeweiligen Fehler in eine Ausnahmetabelle (falls angegeben) und ändert anschließend diese Zeilen. Wenn die Ausnahmetabelle nicht angegeben ist, werden die DATALINK-Spaltenwerte, für die keine Dateireferenz erneut hergestellt werden konnte, mit einer Spaltenkennung und einer Ursachenangabe in eine Ausnahmeberichtsdatei kopiert. Sie können die Daten der Ausnahmetabelle (falls angegeben) oder des Berichts zur Aktualisierung der Zeilen mit den erforderlichen Korrekturen verwenden. Die Ausnahmetabelle, die mit dem Dienstprogramm RECONCILE verwendet wird, ist mit der Ausnahmetabelle des Dienstprogramms LOAD identisch. Weitere Informationen zum Dienstprogramm LOAD finden Sie im Handbuch *Versetzen von Daten Dienstprogramme und Referenz*. Der Bericht verwendet die Namenskonvention *report*.exp (die Erweiterung .exp wird vom Dienstprogramm RECONCILE bereitgestellt). Sie können zum Beispiel mit der folgenden Anweisung das Dienstprogramm RECONCILE aufrufen:

```
db2 RECONCILE dept DLREPORT  /u/scottba/report FOR EXCEPTION excptab
```

Mit diesem Befehl wird die Tabelle mit dem Namen `dept` abgestimmt und Ausnahmen werden in die Ausnahmetabelle `excptab` geschrieben, die vom Benutzer erstellt wurde. Informationen zu Dateien, deren Verbindung beim Abstimmen aufgehoben wurde, werden in die Datei `report.ulk` geschrieben, die im Verzeichnis `/u/scottba` erstellt wird. Wenn *FOR EXCEPTION excptab* nicht angegeben wird, werden die Ausnahmedaten in die Datei `report.exp` geschrieben, die im Verzeichnis `/u/scottba` erstellt wird. Weitere Informationen zum Dienstprogramm RECONCILE finden Sie im Handbuch *Command Reference*.

Erkennen von Situationen, die Abstimmung erfordern

Im Folgenden werden einige Situationen beschrieben, in denen Sie eventuell das Dienstprogramm RECONCILE ausführen müssen:

- Die gesamte Datenbank wird wiederhergestellt und bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt. Da die gesamte Datenbank bis zu einer festgeschriebenen Transaktion aktualisierend wiederhergestellt wird, werden keine Tabellen in den Status *Überprüfung anstehend* (aufgrund referenzieller Integritätsbedingungen oder Prüfungen auf Integritätsbedingungen) versetzt. Sämtliche Daten in der Datenbank werden in einen konsistenten Status gebracht. Die DATALINK-Spalten werden eventuell jedoch nicht mit den Metadaten in DB2 Data Links Manager synchronisiert, so dass eine Abstimmung erforderlich ist.

  In diesem Fall befinden sich Tabellen mit DATALINK-Spaltendaten bereits im Status *DRP (Datalink Reconcile Pending)*. Sie sollten das Dienstprogramm RECONCILE für jede dieser Tabellen ausführen.
- Ein bestimmter Data Links-Server, auf dem DB2 Data Links Manager ausgeführt wird, findet seine Metadaten nicht mehr. Dies kann aus verschiedenen Gründen eintreten. Zum Beispiel:
  - Für den Data Links-Server wurde ein Kaltstart ausgeführt.
  - Die Metadaten des Data Links-Servers wurden in einem früheren Status wiederhergestellt.

  In einigen Fällen, zum Beispiel bei den SQL-Anweisungen UPDATE und DELETE, kann DB2 möglicherweise ein Problem mit den Metadaten auf einem Data Links-Server erkennen. In solchen Fällen würde die SQL-Anweisung fehlschlagen. Als Maßnahme würden Sie die Tabelle mit der Anweisung SET CONSTRAINTS in den Status *DRP (Datalink Reconcile Pending)* versetzen und anschließend das Dienstprogramm RECONCILE für diese Tabelle ausführen.

- Ein Dateisystem ist nicht verfügbar (zum Beispiel aufgrund eines Plattenausfalls) und wird nicht auf den aktuellen Stand wiederhergestellt. In diesem Fall fehlen möglicherweise Dateien.
- Ein Exemplar von DB2 Data Links Manager wird aus einer Datenbank gelöscht, und es sind Werte für DATALINK FILE LINK CONTROL vorhanden, die auf dieses Exemplar verweisen. Führen Sie für solche Tabellen das Dienstprogramm RECONCILE aus.

Zusammenfassung der Abstimmungsprozedur

Gehen Sie wie folgt vor, wenn Sie Data Links nach einer Wiederherstellung bis zu einem bestimmten Zeitpunkt oder aufgrund nicht übereinstimmender Informationen der DB2-Steuerung und der Data Links-Server, die DB2 Data Links Manager ausführen, abstimmen müssen:

1. Versetzen Sie die Tabelle in den Status *DRP (Datalink Reconcile Pending)*, indem Sie die Anweisung SET CONSTRAINTS absetzen. (In einigen Fällen tut DB2 dies von selbst.)
2. Lösen Sie Verbindungen mit dem Dienstprogramm RECONCILE auf, und führen Sie die entsprechenden Maßnahmen für die Ausnahmen in der Ausnahmetabelle oder im Ausnahmebericht aus.

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 Befehls DB2 BACKUP DATABASE müssen Sie lediglich den Aliasnamen der Datenbank angeben, die Sie sichern wollen. Zum 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.

Unter Windows NT/2000 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
```

Sie können beim Aufrufen des Sicherungsdienstprogramms optional eine Zielposition angeben, an der das Sicherungsimage erstellt werden soll. 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; siehe „Anhang G. Tivoli Storage Manager" auf Seite 485)
- Den Server eines anderen Lieferanten
- Unter OS/2 kann die Zielposition auch durch ein Benutzer-Exit-Programm angegeben werden.

Die Datei des Wiederherstellungsprotokolls wird jedesmal automatisch mit den Ergebnisinformationen aktualisiert, wenn Sie eine Gesamtsicherung einer Datenbank starten. Diese Datei wird im selben Verzeichnis wie die Datenbankkonfigurationsdatei erstellt. Weitere Informationen zur Datei des Wiederherstellungsprotokolls finden Sie in „Datei des Wiederherstellungsprotokolls" auf Seite 58.

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

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

Zum Beispiel:

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

Auf anderen Plattformen wird eine Unterverzeichnisbaumstruktur mit vier Ebenen verwendet:

```
DB_alias.art\exemplar.NODEnnnn\CATNnnnn\yyyymmdd\hhmmss.folgenr
```

Beispiel (Windows NT/2000):

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

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

art
Die Art 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 Sicherungs-Header 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. Weitere Informationen zu TSM finden Sie in „Anhang G. Tivoli Storage Manager" auf Seite 485.
- In einem partitionierten Datenbanksystem können Sie darüber hinaus Produkte verwenden, die virtuelle Bandeinheitenfunktionen 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.

Für die Verwendung des Sicherungsdienstprogramms erforderliche Zugriffsrechte und Berechtigungen

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf sie zuzugreifen. Berechtigungsstufen stellen eine Methode dar, Berechtigungen sowie höhere 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

Vor der Verwendung des Sicherungsdienstprogramms

Es sollte noch keine Verbindung zu der zu sichernden Datenbank bestehen: das Sicherungsdienstprogramm stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der Sicherungsoperation beendet wird.

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 Sicherungsdienstprogramm 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. Weitere Informationen zum Befehl LIST NODES finden Sie im Handbuch *Command Reference*.) Wenn Sie auf diese Weise vorgehen, müssen Sie zuerst den Katalogknoten und anschließend die anderen Datenbankpartitionen sichern. Zur Sicherung von Datenbankpartitionen können Sie auch die Befehlszentrale 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 (Oberflächen, 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. Informationen zur Migration von Datenbanken finden Sie im Handbuch *Systemverwaltung: Konzept*.

Verwenden des Sicherungsdienstprogramms

Aufrufen des Sicherungsdienstprogramms

Das Sicherungsdienstprogramm kann wie folgt aufgerufen werden:

- Über den Befehlszeilenprozessor (CLP).

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

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

- Über das Notizbuch bzw. den Assistenten **Datenbank sichern** in der Steuerzentrale. 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.

  Basisinformationen zur Steuerzentrale finden Sie im Handbuch *Systemverwaltung*. Zusatzinformationen bietet die Onlinehilfefunktion der Steuerzentrale.

- Über die Anwendungsprogrammierschnittstelle (API) **sqlubkp**. Informationen zu dieser API finden Sie in „API zur Datenbanksicherung" auf Seite 109. Basisinformationen zum Erstellen von Anwendungen, die DB2-Verwaltungs-APIs enthalten, finden Sie im Handbuch *Application Building Guide*.

Anzeigen von Sicherungsinformationen

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

- Testen der Integrität des Sicherungsimage und Klärung der Frage, ob ein Wiederherstellen möglich ist.
- Anzeigen der im Sicherungs-Header gespeicherten Informationen.

Weitere Informationen zu diesem Dienstprogramm finden Sie in „db2ckbkp - Sicherung überprüfen" auf Seite 352.

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, benötigen Sie bei der Verwendung einer festen Blockgröße für die Sicherung sehr viel Zeit. 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.

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:

```
tcl -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 Sicherungsimage erstellt. Der Befehl **dd** schreibt einen Speicherauszug des Image 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 speichern und diesen anschließend von dieser Einheit 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 Image 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, wieviel Speicherplatz auf der unformatierten Einheit zum Speichern des Image 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. 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. Das folgende Beispiel gilt unter AIX:

1. Erstellen Sie eine benannte Pipe:

   ```
   mkfifo /u/dmcinnis/meinepipe
   ```

2. Verwenden Sie diese Pipe als Ziel für eine Datenbanksicherungsoperation:

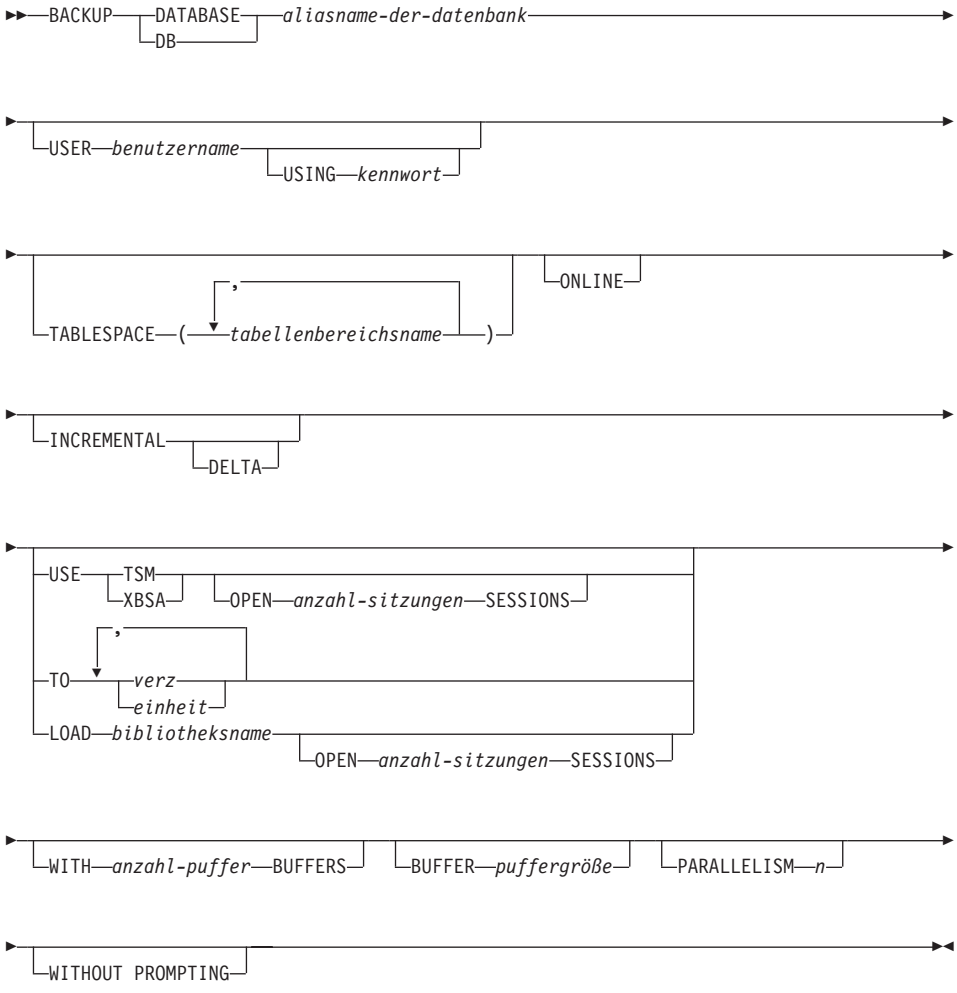
   ```
   db2 backup db sample to /u/dmcinnis/meinepipe
   ```

3. 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
   ```

BACKUP DATABASE, Befehl

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 für Datenbanken verfügbar, die mit den aktivierten Optionen *logretain* oder *userexit* konfiguriert wurden.

Anmerkung: Eine Onlinesicherungsoperation könnte ein Zeitlimit überschreiten, wenn eine IX-Sperre für `sysibm.systables` vorhanden ist, weil das DB2-Sicherungsdienstprogramm eine S-Sperre für Objekte aktiviert, die LOBs enthalten.

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 Delta-Sicherungsimage ist eine Kopie aller Datenbankdaten, die seit der letzten erfolgreichen Sicherungsoperation einer beliebigen Art geändert wurden.

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

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

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

USE XBSA

Gibt an, dass die XBSA-Schnittstelle verwendet werden soll. Backup Services APIs (XBSA) sind offene Anwendungsprogrammierschnittstellen für Anwendungen oder Einsatzmittel, die eine Datenspeicherverwaltung für Sicherungs- oder Archivierungszwecke benötigen. Legato NetWorker ist ein Speichermanager, der zurzeit die XBSA-Schnittstelle unterstützt.

TO verz/einheit

Eine Liste mit Verzeichnissen oder Bandeinheitennamen. Der vollständige Pfad, auf dem sich das Verzeichnis befindet, muss angegeben werden. Das Ziel muss sich auf dem Datenbankserver befinden. Dieser Parameter kann wiederholt werden, um die Zielverzeichnisse und Einheiten anzugeben, die das Sicherungsimage umfasst. Wenn mehr als ein Ziel angegeben wird (z. B. ziel1, ziel2 und ziel3), wird ziel1 zuerst geöffnet. Der Datenträger-Header 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 OS/2 oder dem Windows-Betriebssystem keine allgemeine Bandunterstützung gibt, erfordert jede Art Bandeinheit einen eindeutigen Einheitentreiber. Benutzer müssen sich an die Benennungseinschränkung (8+3) halten, um eine Sicherung auf ein FAT-Dateisystem unter OS/2 oder dem Windows-Betriebssystem 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. Brechen Sie die Sicherungsoperation ab.

Ein Band wird unter OS/2 nicht unterstützt. Unter OS/2 kann `0` oder `0:` angegeben werden, damit eine Sicherungsoperation ein Benutzer-Exit-Programm aufruft (siehe „Anhang H. Benutzer-Exit zur Datenbankwiederherstellung" auf Seite 493). Die Datenbank wird in den Wartemodus versetzt, bevor eine Onlinedatenbanksicherungsoperation mit einem Benutzer-Exit-Programm startet. Das Sicherungsdienstprogramm wartet, bis alle Transaktionen festgeschrieben oder rückgängig gemacht wurden. Während das Dienstprogramm ausgeführt wird, befinden sich alle neuen Transaktionen im Wartestatus, bis die Sicherungsoperation beendet ist.

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 OS/2 oder dem Windows-Betriebssystem), die die Sicherungs- und Wiederherstellungs-E/A-Funktionen des Lieferanten enthält, die verwendet werden sollen. Die Bibliothek kann den vollständigen Pfad enthalten. Wenn kein vollständiger Pfad angegeben wird, wird standardmäßig der Pfad verwendet, auf dem sich das Benutzer-Exit-Programm befindet.

WITH anzahl-puffer BUFFERS
Die Anzahl zu verwendender Puffer. Der Standardwert ist 2. Wenn Sie allerdings eine Sicherung auf mehreren Speicherpositionen erstellen, könnte eine größere Anzahl Puffer verwendet werden, um die Leistung zu verbessern.

BUFFER puffergröße
Die Größe des bei der Erstellung des Sicherungsimage verwendeten Puffers in 4-KB-Seiten. Der Mindestwert für diesen Parameter ist 8 Seiten. Der Standardwert ist 1024 Seiten. Wenn die Puffergröße Null angegeben wird, wird der Wert des Konfigurationsparameters *backbufsz* des Datenbankmanagers als Pufferzuordnungsgröße verwendet.

Wenn Sie ein Band mit variablen Blockgrößen verwenden, reduzieren Sie die Puffergröße auf einen 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 für *backbufsz* oder *restbufsz*, 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. Der Standardwert ist 1.

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

API zur Datenbanksicherung

C-API-Syntax

```
/* File: sqlutil.h */
/* API: Backup Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlubkp (
    char * pDbAlias,
    sqluint32     BufferSize,
    sqluint32     BackupMode,
    sqluint32     BackupType,
    sqluint32     CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    sqluint32     NumBuffers,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaTargetList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    sqluint32 * pBackupSize,
    void * pReserved4,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

Generische API-Syntax

```
/* File: sqlutil.h */
/* API: Backup Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgbkp (
    unsigned short DbAliasLen,
    unsigned short UserNameLen,
    unsigned short PasswordLen,
    unsigned short * pReserved1,
    char * pDbAlias,
    sqluint32 BufferSize,
    sqluint32 BackupMode,
    sqluint32 BackupType,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    sqluint32 NumBuffers,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaTargetList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    sqluint32 * pBackupSize,
    void * pReserved4,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

API-Parameter

DbAliasLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Aliasnamens der Datenbank in Byte darstellt.

UserNameLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Benutzernamens in Byte darstellt. Wird auf null gesetzt, wenn kein Benutzername angegeben wird.

PasswordLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Kennworts in Byte darstellt. Wird auf null gesetzt, wenn kein Kennwort angegeben wird.

pReserved1.
Zur zukünftigen Verwendung reserviert.

pDbAlias
Eingabe. Eine Zeichenfolge, die den (im Systemdatenbankverzeichnis katalogisierten) Aliasnamen der zu sichernden Datenbank enthält.

BufferSize
Eingabe. Sicherungspuffergröße in 4 KB großen Zuordnungseinheiten (Seiten). Die Mindestanzahl ist 8 Einheiten. Der Standardwert ist 1024 Einheiten.

BackupMode
Eingabe. Gibt den Sicherungsmodus an. Gültige Werte (definiert in `sqlutil`) sind:

SQLUB_OFFLINE
Stellt eine exklusive Verbindung zur Datenbank her.

SQLUB_ONLINE
Ermöglicht den Datenbankzugriff durch andere Anwendungen, während die Sicherungsoperation ausgeführt wird.

Anmerkung: Eine Onlinesicherungsoperation könnte ein Zeitlimit überschreiten, wenn eine IX-Sperre für `sysibm.systables` vorhanden ist, weil das DB2-Sicherungsdienstprogramm S-Sperren für SMS-LOB-Objekte und IN-Sperren für alle anderen Objekte aktiviert.

BackupType
Eingabe. Gibt die Art der auszuführenden Sicherung an. Gültige Werte (definiert in `sqlutil`) sind:

SQLUB_FULL
Gibt eine vollständige (nicht inkrementelle) Datenbanksicherungsoperation an. Dieser Wert wird in Zukunft nicht mehr unterstützt. Verwenden Sie `SQLUB_DB`, um eine vollständige Datenbanksicherungsoperation anzugeben.

SQLUB_DB
Gibt eine Sicherung aller Tabellenbereiche in der Datenbank an.

SQLUB_TABLESPACE
Gibt eine Sicherungsoperation auf Tabellenbereichsebene an. Geben Sie im Parameter *pTablespaceList* eine Liste der zu sichernden Tabellenbereiche an.

SQLUB_INCREMENTAL
Gibt ein kumulatives (inkrementelles) Sicherungsimage an. Ein inkrementelles Sicherungsimage ist eine Kopie aller

Datenbankdaten, die seit der letzten erfolgreichen vollständigen Sicherungsoperation geändert wurden.

SQLUB_DELTA
Gibt ein nicht kumulatives (Delta-)Sicherungsimage an. Ein Delta-Sicherungsimage ist eine Kopie aller Datenbankdaten, die seit der letzten erfolgreichen Sicherungsoperation einer beliebigen Art geändert wurden.

CallerAction
Eingabe. Gibt die auszuführende Aktion an. Gültige Werte (definiert in `sqlutil`) sind:

SQLUB_BACKUP
Die Sicherungsoperation starten.

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

SQLUB_CONTINUE
Die Sicherungsoperation fortsetzen, nachdem der Benutzer die vom Dienstprogramm angeforderten Aktionen ausgeführt (z. B. ein neues Band angehängt) hat.

SQLUB_TERMINATE
Die Sicherungsoperation beenden, nachdem der Benutzer eine vom Dienstprogramm angeforderte Aktion nicht ausführen konnte.

SQLUB_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.

SQLUB_PARM_CHECK
Parameter auswerten, ohne eine Sicherungsoperation 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.

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

pApplicationId
Ausgabe. Geben Sie einen Puffer der Länge SQLU_APPLID_LEN+1 (definiert in `sqlutil`) 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.

pTimestamp
Ausgabe. Geben Sie einen Puffer der Länge SQLU_TIME_STAMP-_LEN+1 (definiert in `sqlutil`) an. Die API gibt die Zeitmarke des Sicherungsimage zurück.

NumBuffers
Eingabe. Gibt die Anzahl der zu verwendenden Sicherungspuffer an.

pTablespaceList
Eingabe. Eine Liste der zu sichernden Tabellenbereiche. Nur für Sicherungsoperationen auf Tabellenbereichsebene erforderlich. Siehe „Datenstruktur: SQLU-TABLESPACE-BKRST-LIST" auf Seite 122.

pMediaTargetList
Eingabe. Diese Struktur ermöglicht dem aufrufenden Programm die Angabe einer Zieladresse für die Sicherungsoperation. Die bereitgestellten Daten hängen vom Wert des Feldes *media_type* ab. Die gültigen Werte für *media_type* (definiert in `sqlutil`) sind:

SQLU_LOCAL_MEDIA
Lokale Einheiten (eine Kombination von Bändern, Platten oder Disketten). Geben Sie eine Liste mit *sqlu_media_entry*-Strukturen an. Unter OS/2 oder dem Windows-Betriebssystem können die Einträge nur Verzeichnispfade und keine Bandeinheitennamen sein.

SQLU_TSM_MEDIA
TSM. Wenn keine *sqlu_media_entry*-Struktur verwendet wird, um einen Pfad für das Sicherungsimage anzugeben, initialisieren Sie den Zeiger *media* in der Struktur *sqlu_media_list_targets* mit NULL. Die mit DB2 gelieferte gemeinsam benutzte Bibliothek von TSM wird 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 in der Struktur *sqlu_vendor* an.

SQLU_USER_EXIT
Benutzer-Exit. Es ist keine zusätzliche Eingabe erforderlich (nur unter OS/2 verfügbar).

Siehe „Datenstruktur: SQLU-MEDIA-LIST" auf Seite 118.

pUserName
Eingabe. Eine Zeichenfolge, die den Benutzernamen enthält, der beim Versuch einer Verbindung verwendet wird.

pPassword
Eingabe. Eine Zeichenfolge, die das Kennwort enthält, das mit dem Benutzernamen verwendet werden soll.

pReserved2
Zur zukünftigen Verwendung reserviert.

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

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

Parallelism
Eingabe. Grad der Parallelität (Anzahl Puffermanipulatoren).

pBackupSize
Ausgabe. Größe des Sicherungsimage (in MB). Kann auf NULL gesetzt werden.

pReserved4
Zur zukünftigen Verwendung reserviert.

pReserved3
Zur zukünftigen Verwendung reserviert.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur *sqlca*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

REXX-API-Syntax

```
BACKUP DATABASE dbalias USING :value [USER username USING password]

[TABLESPACE :tablespacenames] [ONLINE]

[LOAD vendor-library [OPTIONS vendor-options] [OPEN num-sessions SESSIONS] |
 TO :target-area |
 USE TSM [OPEN num-sessions SESSIONS] |
 USER_EXIT]

[ACTION caller-action] [WITH num-buffers BUFFERS] [BUFFERSIZE buffer-size]
[PARALLELISM parallelism-degree]
```

REXX-API-Parameter

dbalias
Aliasname der zu sichernden Datenbank.

value Eine zusammengesetzte REXX-Host-Variable, an die die Datenbanksicherungsinformationen zurückgegeben werden. Im Folgenden steht XXX für den Namen der Host-Variablen:

XXX.0	Anzahl Elemente in den Variablen (immer 2)
XXX.1	Die Zeitmarke des Sicherungsimage
XXX.2	Eine Anwendungs-ID, die den Agenten angibt, der die Anwendung bedient

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

password
Das Kennwort, das verwendet wird, um den Benutzernamen zu authentifizieren.

tablespacenames
Eine zusammengesetzte REXX-Host-Variable, die eine Liste der zu sichernden Tabellenbereiche enthält. Im Folgenden ist XXX der Name der Host-Variablen:

XXX.0	Anzahl zu sichernder Tabellenbereiche
XXX.1	Erster Tabellenbereichsname
XXX.2	Zweiter Tabellenbereichsname
XXX.3	und so weiter

vendor-library
Der Name der gemeinsam benutzten Bibliothek (DLL unter Windows-Betriebssystemen oder OS/2), die die Sicherungs- und Wiederherstellungs-E/A-Funktionen des Lieferanten enthält, die verwendet werden sollen. Kann den vollständigen Pfad enthalten. Wenn der vollständige Pfad nicht angegeben wird, wird standardmäßig der Pfad verwendet, in dem sich das Benutzer-Exit-Programm befindet.

vendor-options
Erforderliche Informationen für die Funktionen anderer Lieferanten.

num-sessions
Die Anzahl der E/A-Sitzungen, die mit Tivoli Storage Manager (TSM) oder einem Produkt eines anderen Lieferanten verwendet werden sollen.

target-area
Lokale Einheiten. Ermöglicht eine Kombination von Bändern, Platten oder Disketten. Geben Sie eine Liste an (siehe „Datenstruktur: SQLU-MEDIA-LIST" auf Seite 118). Unter OS/2 oder dem Windows-Betriebssystem können die Einträge nur Verzeichnispfade und keine Bandeinheitennamen sein.

caller-action
Gibt die auszuführende Aktion an. Gültige Werte sind:

SQLUB_BACKUP
Die Sicherungsoperation starten.

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

SQLUB_CONTINUE
Die Sicherungsoperation fortsetzen, nachdem der Benutzer die vom Dienstprogramm angeforderten Aktionen ausgeführt hat (setzen Sie z. B. das Anhängen eines neuen Bandes fort).

SQLUB_TERMINATE
Die Sicherungsoperation beenden, nachdem der Benutzer eine vom Dienstprogramm angeforderte Aktion nicht ausführen konnte.

SQLUB_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.

SQLUB_PARM_CHECK
Parameter auswerten, ohne eine Sicherungsoperation auszuführen.

num-buffers
Die Anzahl zu verwendender Sicherungspuffer.

buffer-size
Sicherungspuffergröße in 4 KB großen Zuordnungseinheiten. Der Mindestwert ist 8 Einheiten.

parallelism-degree
Grad der Parallelität (Anzahl Puffermanipulatoren).

Datenstruktur: SQLU-MEDIA-LIST

Diese Struktur wird für folgende Zwecke verwendet:

- Zur Angabe einer Liste von *Zieldatenträgern* für das Sicherungsimage (siehe „API zur Datenbanksicherung" auf Seite 109).
- Zur Angabe einer Liste von *Quellendatenträgern* für das Sicherungsimage (siehe „API zur Wiederherstellung von Datenbanken" auf Seite 140).
- Zur Übergabe von Daten an das DB2-Dienstprogramm LOAD.

Tabelle 1. Felder in der Struktur SQLU-MEDIA-LIST

Feldname	Datentyp	Beschreibung
MEDIA_TYPE	CHAR(1)	Ein Zeichen, das einen Datenträgertyp angibt.
SESSIONS	INTEGER	Gibt die Anzahl der Elemente in der Matrix an, auf die das *Zielfeld* dieser Struktur zeigt.
TARGET	Union	Dieses Feld ist ein Zeiger auf einen von drei Strukturtypen. Der Strukturtyp, auf den gezeigt wird, ist vom Wert des Feldes *media_type* bestimmt. Weitere Informationen zu Angaben in diesem Feld finden Sie im Zusammenhang mit der entsprechenden API.

Tabelle 2. Felder in der Struktur SQLU-MEDIA-LIST-TARGETS

Feldname	Datentyp	Beschreibung
MEDIA	Zeiger	Ein Zeiger auf eine *sqlu_media_entry*-Struktur.
VENDOR	Zeiger	Ein Zeiger auf eine *sqlu_vendor*-Struktur.
LOCATION	Zeiger	Ein Zeiger auf eine *sqlu_location_entry*-Struktur.

Tabelle 3. Felder in der Struktur SQLU-MEDIA-ENTRY

Feldname	Datentyp	Beschreibung
RESERVE_LEN	INTEGER	Länge des Feldes *media_entry*. Dies gilt für andere Programmiersprachen als C.
MEDIA_ENTRY	CHAR(215)	Pfad für ein Sicherungsimage, das von Sicherungs- und Wiederherstellungsdienstprogrammen verwendet wird.

Tabelle 4. Felder in der Struktur SQLU-VENDOR

Feldname	Datentyp	Beschreibung
RESERVE_LEN1	INTEGER	Länge des Feldes *shr_lib*. Dies gilt für andere Programmiersprachen als C.
SHR_LIB	CHAR(255)	Name einer gemeinsam benutzten Bibliothek von anderen Lieferanten, mit der Sie Daten speichern und abrufen können.

Tabelle 4. Felder in der Struktur SQLU-VENDOR (Forts.)

Feldname	Datentyp	Beschreibung
RESERVE_LEN2	INTEGER	Länge des Feldes *filename*. Dies gilt für andere Programmiersprachen als C.
FILENAME	CHAR(255)	Dateiname zum Angabe der Ladeeingabe-quelle, wenn Sie eine gemeinsam benutzte Bibliothek verwenden.

Tabelle 5. Felder in der Struktur SQLU-LOCATION-ENTRY

Feldname	Datentyp	Beschreibung
RESERVE_LEN	INTEGER	Länge des Feldes *location_entry*. Dies gilt für andere Programmiersprachen als C.
LOCATION_ENTRY	CHAR(256)	Name der Eingabedatendateien für das Dienstprogramm LOAD.

Gültige Werte für *MEDIA_TYPE* (definiert in `sqlutil`):

SQLU_LOCAL_MEDIA
Lokale Einheiten (Bänder, Platten oder Disketten)

SQLU_SERVER_LOCATION
Servereinheiten (Bänder, Platten oder Disketten; nur Laden). Diesen Wert können Sie nur für den Parameter *pDataFileList* angeben.

SQLU_TSM_MEDIA
TSM

SQLU_OTHER_MEDIA
Lieferantenbibliothek

SQLU_USER_EXIT
Benutzer-Exit (nur OS/2)

SQLU_PIPE_MEDIA
Benannte Pipe (nur für Lieferanten-APIs)

SQLU_DISK_MEDIA
Platte (nur für Lieferanten-APIs)

SQLU_DISKETTE_MEDIA
Diskette (nur für Lieferanten-APIs)

SQLU_TAPE_MEDIA
Band (nur für Lieferanten-APIs)

Datenstruktur: SQLU-MEDIA-LIST

Sprachsyntax

C-Struktur

```
/* File: sqlutil.h */
/* Structure: SQLU-MEDIA-LIST */
/* ... */
typedef SQL_STRUCTURE sqlu_media_list
{
  char            media_type;
  char            filler[3];
  sqlint32            sessions;
  union sqlu_media_list_targets target;
} sqlu_media_list;
/* ... */

/* File: sqlutil.h */
/* Structure: SQLU-MEDIA-LIST-TARGETS */
/* ... */
union sqlu_media_list_targets
{
  struct sqlu_media_entry        *media;
  struct sqlu_vendor             *vendor;
  struct sqlu_location_entry     *location;
};
/* ... */

/* File: sqlutil.h */
/* Structure: SQLU-MEDIA-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_media_entry
{
  sqluint32       reserve_len;
  char            media_entry[SQLU_DB_DIR_LEN+1];
} sqlu_media_entry;
/* ... */

/* File: sqlutil.h */
/* Structure: SQLU-VENDOR */
/* ... */
typedef SQL_STRUCTURE sqlu_vendor
{
  sqluint32       reserve_len1;
  char            shr_lib[SQLU_SHR_LIB_LEN+1];
  sqluint32       reserve_len2;
  char            filename[SQLU_SHR_LIB_LEN+1];
} sqlu_vendor;
/* ... */
```

```
/* File: sqlutil.h */
/* Structure: SQLU-LOCATION-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_location_entry
{
  sqluint32       reserve_len;
  char            location_entry[SQLU_MEDIA_LOCATION_LEN+1];
} sqlu_location_entry;
/* ... */
```

COBOL-Struktur

```
* File: sqlutil.cbl
01 SQLU-MEDIA-LIST.
    05 SQL-MEDIA-TYPE         PIC X.
    05 SQL-FILLER             PIC X(3).
    05 SQL-SESSIONS           PIC S9(9) COMP-5.
    05 SQL-TARGET.
        10 SQL-MEDIA          USAGE IS POINTER.
        10 SQL-VENDOR         REDEFINES SQL-MEDIA
        10 SQL-LOCATION       REDEFINES SQL-MEDIA
        10 FILLER             REDEFINES SQL-MEDIA
*

* File: sqlutil.cbl
01 SQLU-MEDIA-ENTRY.
    05 SQL-MEDENT-LEN         PIC 9(9) COMP-5.
    05 SQL-MEDIA-ENTRY        PIC X(215).
    05 FILLER                 PIC X.
*

* File: sqlutil.cbl
01 SQLU-VENDOR.
    05 SQL-SHRLIB-LEN         PIC 9(9) COMP-5.
    05 SQL-SHR-LIB            PIC X(255).
    05 FILLER                 PIC X.
    05 SQL-FILENAME-LEN       PIC 9(9) COMP-5.
    05 SQL-FILENAME           PIC X(255).
    05 FILLER                 PIC X.
*

* File: sqlutil.cbl
01 SQLU-LOCATION-ENTRY.
    05 SQL-LOCATION-LEN       PIC 9(9) COMP-5.
    05 SQL-LOCATION-ENTRY     PIC X(255).
    05 FILLER                 PIC X.
*
```

Datenstruktur: SQLU-TABLESPACE-BKRST-LIST

Diese Struktur können Sie verwenden, um eine Liste von Tabellenbereichsnamen anzugeben.

Tabelle 6. Felder in der Struktur SQLU-TABLESPACE-BKRST-LIST

Feldname	Datentyp	Beschreibung
NUM_ENTRY	INTEGER	Anzahl der Einträge in der Liste, auf die das Feld *tablespace* zeigt.
TABLESPACE	Zeiger	Ein Zeiger auf eine *sqlu_tablespace_entry*-Struktur.

Tabelle 7. Felder in der Struktur SQLU-TABLESPACE-ENTRY

Feldname	Datentyp	Beschreibung
RESERVE_LEN	INTEGER	Länge der Zeichenfolge, die im Feld *tablespace_entry* angegeben ist. Dies gilt für andere Programmiersprachen als C.
TABLESPACE_ENTRY	CHAR(19)	Name des Tabellenbereichs.

Sprachsyntax

C-Struktur

```
/* File: sqlutil.h */
/* Structure: SQLU-TABLESPACE-BKRST-LIST */
/* ... */
typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
  long            num_entry;
  struct sqlu_tablespace_entry *tablespace;
} sqlu_tablespace_bkrst_list;
/* ... */

/* File: sqlutil.h */
/* Structure: SQLU-TABLESPACE-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_tablespace_entry
{
  sqluint32       reserve_len;
  char            tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
  char            filler[1];
} sqlu_tablespace_entry;
/* ... */
```

COBOL-Struktur

```
* File: sqlutil.cbl
01 SQLU-TABLESPACE-BKRST-LIST.
    05 SQL-NUM-ENTRY          PIC S9(9) COMP-5.
    05 SQL-TABLESPACE         USAGE IS POINTER.
*

* File: sqlutil.cbl
01 SQLU-TABLESPACE-ENTRY.
    05 SQL-TBSP-LEN           PIC 9(9) COMP-5.
    05 SQL-TABLESPACE-ENTRY   PIC X(18).
    05 FILLER                 PIC X.
    05 SQL-FILLER             PIC X(1).
*
```

Beispiele für Sicherungssitzungen

CLP-Beispiele

```
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
```

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 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
```

Ein Beispiel eines DB2-Befehls-Scripts sowie Informationen zur Verwendung finden Sie in „Anhang F. Befehlszeilen-Script zur Wiederherstellung" auf Seite 477.

API-Beispiele

Beispielprogramme, die DB2-APIs und eingebettete SQL-Aufrufe enthalten, sowie Informationen zu deren Verwendung finden Sie in „Anhang E. Beispiele für Wiederherstellungsprogramme" auf Seite 411.

Optimieren der Leistung des Sicherungsdienstprogramms

Gehen Sie wie folgt vor, um die erforderliche Zeit für eine Sicherungsoperation zu verringern:

- 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 Tabellenbereich.
- 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 beim Lesen von Daten aus der Datenbank gestartet werden. Jeder Prozess bzw. Thread wird einem bestimmten Tabellenbereich zugeordnet. Wenn er das Sichern des Tabellenbereichs beendet hat, fordert er einen

anderen Bereich an. Beachten Sie jedoch, dass für jeden Prozess bzw. Thread sowohl Hauptspeicher- als auch CPU-Systemaufwand anfallen, d. h.,belassen Sie den Parameter PARALLELISM bei einem stark ausgelasteten System auf dem Standardwert 1.

- Erhöhen Sie die Sicherungspuffergröße.

  Die ideale Puffergröße für eine Sicherung ist ein Vielfaches der Speicherbereichsgröße des Tabellenbereichs. Wenn Sie mehrere Tabellenbereiche mit verschiedenen Speicherbereichsgrößen haben, geben Sie einen Wert an, der einem Vielfachen des größten Speicherbereichs entspricht.

- Erhöhen Sie die Anzahl von Puffern.

  Wenn Sie mehrere Puffer und E/A-Kanäle verwenden, sollten Sie mindestens doppelt so viele Puffer wie Kanäle verwenden, um sicherzustellen, dass die Kanäle nicht auf Daten warten müssen.

- Verwenden Sie mehrere Zieleinheiten.

Sicherungsdienstprogramm - 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.

Sicherungsdienstprogramm - Fehlerbehebung

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*.

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.

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:

Wiederherstellen - Übersicht

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

```
db2 restore db sample
```

Da die Datenbank SAMPLE vorhanden ist, wird in diesem Beispiel die folgende Nachricht zurückgegeben:

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

Wenn Sie `j` angeben und ein Sicherungsimage der Datenbank SAMPLE vorhanden ist, sollte die Wiederherstellungsoperation erfolgreich beendet werden können.

Wiederherstellen - Übersicht

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 haben Sie entweder die Möglichkeit, ein Sicherungsimage einzelner Tabellenbereiche zu verwenden, oder die Möglichkeit, ein Sicherungsimage einer vollständigen Datenbank zu verwenden und einen oder mehrere Tabellenbereiche aus diesem Image auszuwählen. Alle Protokolldateien, die diesen Tabellenbereichen ab dem Zeitpunkt der Erstellung des Sicherungsimage zugeordnet wurden, müssen vorhanden sein.

Für die Verwendung des Wiederherstellungsdienstprogramms erforderliche Zugriffsrechte und Berechtigungen

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf sie zuzugreifen. Berechtigungsstufen stellen eine Methode dar, Berechtigungen sowie höhere 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 Wiederherstellungsdienstprogramms

Vor der Verwendung des Wiederherstellungsdienstprogramms

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.

Aufrufen des Wiederherstellungsdienstprogramms

Das Wiederherstellungsdienstprogramm kann wie folgt aufgerufen werden:

- Über den Befehlszeilenprozessor (CLP).

  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
  ```

- Über das Notizbuch bzw. den Assistenten **Datenbank wiederherstellen** in der Steuerzentrale. 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.

  Basisinformationen zur Steuerzentrale finden Sie im Handbuch *Systemverwaltung*. Zusatzinformationen bietet die Onlinehilfefunktion der Steuerzentrale.
- Über die Anwendungsprogrammierschnittstelle (API) **sqlrestore**. Informationen zu dieser API finden Sie in „API zur Wiederherstellung von Datenbanken" auf Seite 140. Basisinformationen zum Erstellen von Anwendungen, die DB2-Verwaltungs-APIs enthalten, finden Sie im Handbuch *Application Building Guide*.

Erneutes Definieren von Tabellenbereichsbehältern während einer Wiederherstellungsoperation (Umgeleitete Wiederherstellung)

Während der Sicherung einer Datenbank wird ein Datensatz über 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 verschiedene 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. Beispiele für umgeleitete Wiederherstellung unter Verwendung des Befehlszeilenprozessors, eines Befehls-Scripts oder der Anwendungsprogrammierschnittstelle finden Sie in „Beispiele für Wiederherstellungssitzungen” auf Seite 152.

Während einer umgeleiteten Wiederherstellungsoperation werden Verzeichnis- und Dateibehälter automatisch erstellt, sofern sie nicht bereits existieren. 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. Auf ähnliche Weise könnten Sie einen DMS-Tabellenbereich aus Dateibehältern in Einheitenbehälter verschieben.

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, 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.

Die Datenbanknummer ist auch nach der Wiederherstellung der Datenbank unverändert, selbst wenn das Sicherungsimage eine andere Datenbanknummer besitzt. 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 Sicherungsimage.
- Ersetzen der Tabelleneinträge der wiederherzustellenden Tabellenbereiche.
- Beibehalten der Datei des Wiederherstellungsprotokolls, sofern sie nicht beschädigt ist. Wenn diese Datei beschädigt ist, kopiert der Datenbankmanager die Datei aus dem Sicherungsimage.
- Beibehalten der Authentifizierungsart 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 Konfigurationsdatei der Datenbank kopiert. Das Dienstprogramm überprüft den Protokollpfad: Falls der Pfad von der Datenbank nicht verwendet werden kann, wird die Datenbank dahingehend geändert, dass sie den Standardprotokollpfad verwendet.

Wiederherstellen in eine neue Datenbank

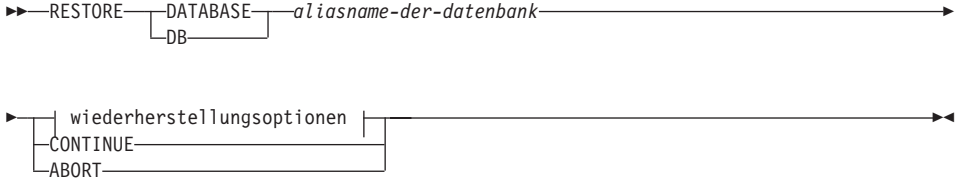
Sie können eine neue Datenbank erstellen und anschließend das Sicherungsimage einer vollständigen Datenbank in diese neue Datenbank wiederherstellen. Die Codepages des Sicherungsimage und der Zieldatenbank müssen übereinstimmen.

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

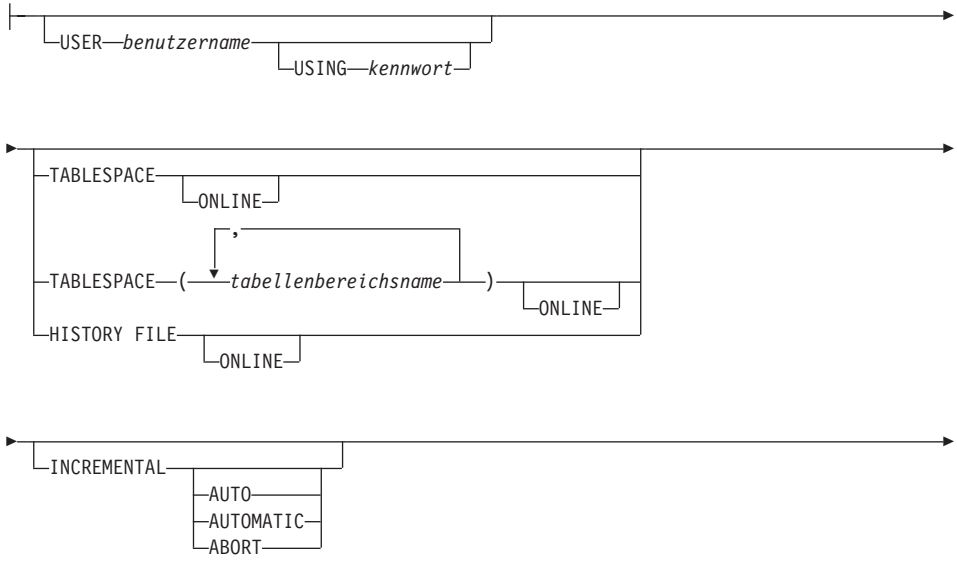
- Erstellen einer neuen Datenbank unter Verwendung des Aliasnamens der Datenbank, der im Parameter für den Aliasnamen der Zieldatenbank angegeben wurde. (Wenn für die Zieldatenbank kein Aliasname angegeben wurde, erstellt das Wiederherstellungsdienstprogramm eine Datenbank mit einem Aliasnamen, der im 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 Datenbank dahingehend geändert, dass sie den Standardprotokollpfad verwendet.
- Wiederherstellen der Authentifizierungsart aus dem Sicherungsimage.
- Wiederherstellen des Inhalts aus den Datenbankverzeichnissen im Sicherungsimage.
- Wiederherstellen der Datei des Wiederherstellungsprotokolls für die Datenbank.

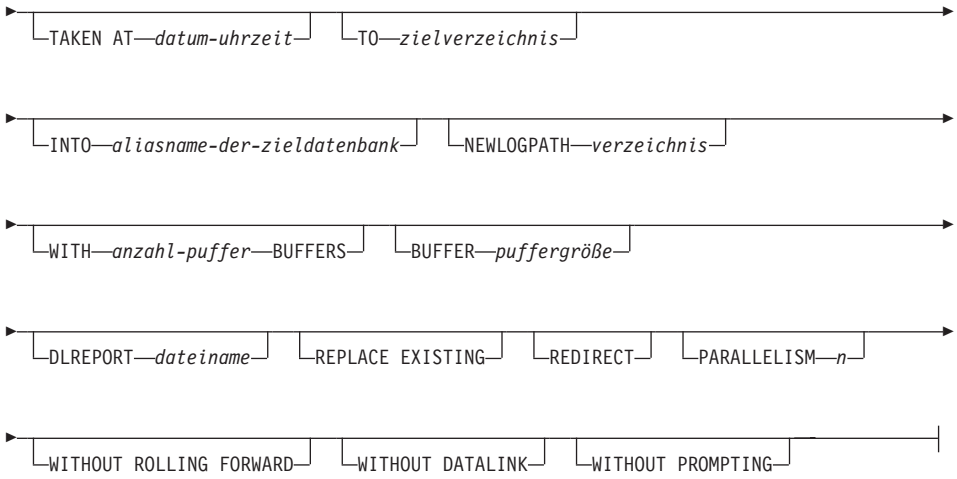
RESTORE DATABASE, Befehl

Befehlssyntax



wiederherstellungsoptionen:





Befehlsparameter

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

CONTINUE
Gibt an, dass die Behälter erneut definiert 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.

INCREMENTAL
Gibt eine manuelle kumulative Wiederherstellungsoperation an. Der Benutzer muss jeden Wiederherstellungsbefehl manuell absetzen.

AUTOMATIC/AUTO
Gibt eine automatische kumulative (inkrementelle) Wiederherstellungsoperation an.

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

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. Backup Services APIs (XBSA) sind offene Anwendungsprogrammierschnittstellen für Anwendungen oder Einsatzmittel, die eine Datenspeicherverwaltung für Sicherungs- oder Archivierungszwecke benötigen. Legato NetWorker ist ein Speichermanager, der zurzeit die XBSA-Schnittstelle unterstützt.

FROM verzeichnis/einheit
Das Verzeichnis oder die Einheit, in dem/auf der sich das Sicherungsimage befindet. Wenn die Parameter USE TSM, FROM und LOAD nicht angegeben werden, ist der Standardwert das aktuelle Verzeichnis.

Unter Windows-Betriebssystemen oder OS/2 darf das angegebene Verzeichnis kein Verzeichnis sein, das von DB2 generiert worden ist. Z. B. wurden die folgenden Befehle abgesetzt:

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

DB2 generiert Unterverzeichnisse unter dem Verzeichnis `c:\backup`, die ignoriert werden sollten. 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. Brechen Sie die Wiederherstellungsoperation ab, nachdem der Benutzer eine vom Dienstprogramm angeforderte Aktion nicht ausführen konnte.

Ein Band wird unter OS/2 nicht unterstützt. Unter OS/2 kann `0` oder `0:` angegeben werden, damit das Wiederherstellungsdienstprogramm ein Benutzer-Exit-Programm aufruft. (Dies kann nur passieren, wenn zur Sicherung der Datenbank ein Benutzer-Exit-Programm verwendet worden ist.) Bei der Wiederherstellung über ein Benutzer-Exit-Programm ist der Pfad zur Datenbank der einzige verwendete Verweis, um die Behälter zu suchen. Deshalb werden alle Behälter für diese Datenbank wiederhergestellt.

Die umgeleitete Wiederherstellung ist nicht zulässig, wenn ein Benutzer-Exit-Programm verwendet wird.

LOAD gemeinsame-bibliothek
Der Name der gemeinsam benutzten Bibliothek (DLL unter Windows-Betriebssystemen oder OS/2), 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 Benutzer-Exit-Programm befindet.

TAKEN AT datum-uhrzeit
Die Zeitmarke des Datenbanksicherungsimage. Die Zeitmarke wird nach erfolgreicher Fertigstellung der Sicherungsoperation angezeigt und ist ein Teil des Pfadnamens des Sicherungsimage. Sie wird in der Form *jjjjmmtthhmmss* angegeben. Sie können auch einen Teil einer Zeitmarke angeben. Wenn z. B. zwei unterschiedliche Sicherungsimages mit den Zeitmarken 19971001010101 und 19971002010101 vorhanden sind, wird nach Eingabe von `19971002` das Image mit der Zeitmarke `19971002010101` 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 wiederhergestellt wird.

Anmerkung: Geben Sie unter Windows-Betriebssystemen oder OS/2 nur den Laufwerkbuchstaben an, wenn Sie diesen Parameter verwenden. Wenn Sie einen längeren Pfad angeben, wird ein Fehler zurückgegeben.

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

NEWLOGPATH verzeichnis
Der vollständig qualifizierte Name 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 Verwendung 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. Der Standardwert ist 2. Sie können allerdings 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. Der Mindestwert für diesen Parameter ist 8 Seiten. Der Standardwert ist 1024 Seiten. Wenn die Puffergröße Null angegeben wird, wird der Wert des Konfigurationsparameters *restbufsz* des Datenbankmanagers als Pufferzuordnungsgröße verwendet.

Die Wiederherstellungspuffergröße muss ein positives ganzzahliges Vielfaches der Sicherungspuffergröße sein, die bei der Sicherungsoperation angegeben wurde. Wenn eine ungültige Puffergröße angegeben wird, werden Puffer der kleinsten zulässigen Größe zugeordnet.

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

DLREPORT dateiname

Wenn der Dateiname angegeben wird, muss er vollständig qualifiziert werden. Berichtet die Dateien, 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 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 Scripts 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 aus demselben Fenster oder derselben 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 DATABASE (siehe „ROLLFORWARD DATABASE, Befehl" auf Seite 171) aufgerufen werden, bevor die Datenbank wieder verwendet werden kann.

WITHOUT DATALINK
Gibt an, dass Tabellen mit DATALINK-Spalten in den Status DRP (DataLink 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. Der Standardwert ist 1.

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.

API zur Wiederherstellung von Datenbanken

C-API-Syntax

```
/* File: sqlutil.h */
/* API: Restore Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlurestore (
    char * pSourceDbAlias,
    char * pTargetDbAlias,
    sqluint32 BufferSize,
    sqluint32 RollforwardMode,
    sqluint32 DatalinkMode,
    sqluint32 RestoreType,
    sqluint32 RestoreMode,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    char * pTargetPath,
    sqluint32 NumBuffers,
    char * pReportFile,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaSourceList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    void * pRestoreInfo,
    void * pContainerPageList,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

Generische API-Syntax

```
/* File: sqlutil.h */
/* API: Restore Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgrestore (
   unsigned short SourceDbAliasLen,
   unsigned short TargetDbAliasLen,
   unsigned short TimestampLen,
   unsigned short TargetPathLen,
   unsigned short UserNameLen,
   unsigned short PasswordLen,
   unsigned short ReportFileLen,
   unsigned short Reserved2Len,
   char * pSourceDbAlias,
   char * pTargetDbAlias,
   sqluint32 BufferSize,
   sqluint32 RollforwardMode,
   sqluint32 DatalinkMode,
   sqluint32 RestoreType,
   sqluint32 RestoreMode,
   sqluint32 CallerAction,
   char * pApplicationId,
   char * pTimestamp,
   char * pTargetPath,
   sqluint32 NumBuffers,
   char * pReportFile,
   struct sqlu_tablespace_bkrst_list * pTablespaceList,
   struct sqlu_media_list * pMediaSourceList,
   char * pUserName,
   char * pPassword,
   void * pReserved2,
   sqluint32 VendorOptionsSize,
   void * pVendorOptions,
   sqluint32 Parallelism,
   unsigned short RestoreInfoSize,
   void * pRestoreInfo,
   unsigned short ContainerPageListSize,
   void * pContainerPageList,
   void * pReserved3,
   struct sqlca * pSqlca);
/* ... */
```

API-Parameter

SourceDbAliasLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Aliasnamens der Quellendatenbank in Byte darstellt.

TargetDbAliasLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge

des Aliasnamens der Zieldatenbank in Byte darstellt. Wird auf null gesetzt, wenn kein Aliasname der Zieldatenbank angegeben wird.

TimestampLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge der Zeitmarke in Byte darstellt. Wird auf null gesetzt, wenn keine Zeitmarke angegeben wird.

TargetPathLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Zielverzeichnisses in Byte darstellt. Wird auf null gesetzt, wenn kein Zielpfad angegeben wird.

UserNameLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Benutzernamens in Byte darstellt. Wird auf null gesetzt, wenn kein Benutzername angegeben wird.

PasswordLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Kennworts in Byte darstellt. Wird auf null gesetzt, wenn kein Kennwort angegeben wird.

ReportFileLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Namens der Berichtsdatei in Byte darstellt. Wird auf null gesetzt, wenn kein Berichtsdateiname angegeben wird.

Reserved2Len
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des reservierten Bereichs in Byte darstellt. Auf null gesetzt.

pSourceDbAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der Quellen-datenbank des Sicherungsimage enthält.

pTargetDbAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der Zieldatenbank enthält. Wenn der Wert dieses Parameters NULL ist, wird der Wert von *pSourceDbAlias* verwendet.

BufferSize
Eingabe. Sicherungspuffergröße in 4 KB großen Zuordnungseinheiten (Seiten). Der Mindestwert ist 8 Einheiten. Der Standardwert ist 1024 Einheiten.

Die angegebene Puffergröße muss gleich der für die Erstellung des Sicherungsimage verwendeten Puffergröße s oder ein ganzzahliges Vielfaches dieses Werts ein.

RollforwardMode
Eingabe. Gibt an, ob die Datenbank am Ende der Wiederherstellungsoperation in den Status *Aktualisierende Wiederherstellung anstehend* versetzt werden soll oder nicht. Gültige Werte (definiert in `sqlutil`) sind:

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

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

Wenn sich die Datenbank nach einer erfolgreichen Wiederherstellungsoperation im Status *Aktualisierende Wiederherstellung anstehend* befindet, muss die API zur aktualisierenden Wiederherstellung von Datenbanken (siehe „API zur aktualisierenden Wiederherstellung von Datenbanken" auf Seite 178) aufgerufen werden, bevor die Datenbank wieder verwendet werden kann.

DatalinkMode
Eingabe. Gibt an, ob Tabellen mit DATALINK-Spalten in den Status DRP (DataLink Reconcile Pending) versetzt werden sollen und ob eine Abstimmung der verbundenen Dateien ausgeführt werden soll. Gültige Werte (definiert in `sqlutil`) sind:

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

SQLUD_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.

RestoreType
Eingabe. Gibt die Art der Wiederherstellungsoperation an. Gültige Werte (definiert in `sqlutil`) sind:

SQLUD_FULL
Alles aus dem Sicherungsimage wiederherstellen. Diese Operation wird offline ausgeführt. Dieser Wert wird in Zukunft nicht mehr unterstützt. Verwenden Sie `SQLUD_DB`, um eine vollständige Datenbankwiederherstellungsoperation anzugeben.

SQLUD_DB
Alle Tabellenbereiche in der Datenbank wiederherstellen. Diese Operation wird offline ausgeführt.

SQLUD_ONLINE_TABLESPACE
Nur Sicherungsimages auf Tabellenbereichsebene wiederherstellen. Diese Operation wird online ausgeführt. Dieser Wert wird in Zukunft nicht mehr unterstützt. Verwenden Sie `SQLUD_TABLESPACE | SQLUD_ONLINE`, um eine Onlinewiederherstellungsoperation auf Tabellenbereichsebene anzugeben.

SQLUD_TABLESPACE
Nur die Sicherungsimages auf Tabellenbereichsebene wiederherstellen. Diese Operation kann online oder offline ausgeführt werden.

SQLUD_HISTORY
Nur die Datei des Wiederherstellungsprotokolls wiederherstellen.

SQLUD_INCREMENTAL
Eine manuelle kumulative Wiederherstellungsoperation ausführen.

SQLUD_AUTOMATIC
Eine automatische kumulative (inkrementelle) Wiederherstellungsoperation ausführen. Dieser Parameter muss mit `SQLUD_INCREMENTAL` angegeben werden, d. h. `SQLUD_INCREMENTAL | SQLUD_AUTOMATIC`.

RestoreMode
Eingabe. Gibt an, ob die Wiederherstellungsoperation offline oder online ausgeführt werden soll. Gültige Werte (definiert in `sqlutil`) sind:

SQLUD_OFFLINE
Eine Offlinewiederherstellungsoperation ausführen.

SQLUD_ONLINE
Eine Onlinewiederherstellungsoperation ausführen.

CallerAction
Eingabe. Gibt die Art der auszuführenden Aktion an. Gültige Werte (definiert in `sqlutil`) sind:

SQLUD_RESTORE
Die Wiederherstellungsoperation starten.

SQLUD_TERMINATE_INCREMENTAL
Eine Teilwiederherstellungsoperation vor der Fertigstellung beenden.

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

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

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

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

SQLUD_PARM_CHECK
Parameter auswerten, ohne die Wiederherstellungsoperation auszuführen.

SQLUD_RESTORE_STORDEF
Erstaufruf. Eine Neudefinition eines Tabellenbereichsbehälters anfordern.

CallerAction muss beim ersten Aufruf auf `SQLUD_RESTORE`, `SQLUD_NOINTERRUPT`, `SQLUD_RESTORE_STORDEF` oder `SQLUD_PARM_CHECK` gesetzt werden.

pApplicationId
Ausgabe. Geben Sie einen Puffer der Länge `SQLU_APPLID_LEN+1` (definiert in `sqlutil`) an. Das Wiederherstellungsdienstprogramm gibt eine Zeichenfolge zurück, die den Agenten angibt, der die Anwendung bedient. Kann mit den Datenbanksystemmonitor-APIs verwendet werden, um die Anwendung zu überwachen.

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

pTargetPath
Eingabe. Eine Zeichenfolge, die den relativen oder vollständig qualifizierten Namen des Zieldatenbankverzeichnisses enthält. Wird verwendet, wenn eine neue Datenbank während der Wiederherstellungsoperation erstellt werden soll.

NumBuffers
Eingabe. Die Anzahl Puffer, die für die Wiederherstellungsoperation verwendet werden.

pReportFile
Wenn der Dateiname angegeben wird, muss er vollständig qualifiziert werden. Berichtet die Dateien, 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.

pTablespaceList
Gibt mindestens einen wiederherzustellenden Tabellenbereich an. Wird verwendet, wenn Sie eine Untergruppe der Sicherungsimages oder einen Tabellenbereich aus einem Sicherungsimage auf Tabellenbereichsebene wiederherstellen.

Es gelten folgende Einschränkungen:

- Die Datenbank muss wiederherstellbar sein. Bei wiederherstellbaren Datenbanken ist der Konfigurationsparameter *logretain* der Datenbank auf den Wert „RECOVERY" gesetzt, und/oder der Konfigurationsparameter *userexit* der Datenbank ist aktiviert.
- Die Datenbank, in die wiederhergestellt wird, muss dieselbe Datenbank sein, die zur Erstellung des Sicherungsimage verwendet wurde. D. h., über die Tabellenbereichswiederherstellungsoperation können einer Datenbank keine Tabellenbereiche hinzugefügt werden.
- Diese Funktion ist nicht verfügbar, wenn Sie über einen Benutzer-Exit unter OS/2 wiederherstellen.
- Das Dienstprogramm zur aktualisierendenn Wiederherstellung stellt sicher, dass in einer MPP-Umgebung wiederhergestellte Tabellenbereiche mit allen anderen Knoten synchronisiert werden, die dieselben Tabellenbereiche enthalten.

Anmerkung: Beim Wiederherstellen eines Tabellenbereichs, der seit seiner Sicherung umbenannt wurde, müssen Sie den neuen Tabellenbereichsnamen verwenden, wenn Sie das Wiederherstellungsdienstprogramm aufrufen. Wenn der alte Name verwendet wird, wird der Tabellenbereich nicht gefunden.

pMediaSourceList
Eingabe. Quellendatenträger für das Sicherungsimage. Siehe „Datenstruktur: SQLU-MEDIA-LIST" auf Seite 118. Die Informationen, die das aufrufende Programm in dieser Struktur angeben muss, hängt von dem Wert des Felds *media_type* ab. Gültige Werte für dieses Feld (definiert in `sqlutil`) sind:

SQLU_LOCAL_MEDIA
Lokale Einheiten (eine Kombination von Bändern, Platten oder Disketten). Geben Sie eine Liste mit *sqlu_media_entry*-Strukturen an. Unter Windows-Betriebssystemen oder OS/2 können die Einträge nur Verzeichnispfade und keine Bandeinheitennamen sein.

SQLU_TSM_MEDIA
TSM. Es ist keine weitere Eingabe erforderlich. Die mit DB2 gelieferte gemeinsam benutzte Bibliothek von TSM wird verwendet. Wenn Sie eine andere Version 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 in der Struktur *sqlu_vendor* an.

SQLU_USER_EXIT
Benutzer-Exit. Es ist keine zusätzliche Eingabe erforderlich (nur unter OS/2 verfügbar).

pUserName
Eingabe. Eine Zeichenfolge, die den Benutzernamen enthält, der für eine Verbindung verwendet wird.

pPassword
Eingabe. Eine Zeichenfolge, die das zum Authentifizieren des Benutzernamens verwendete Kennwort enthält.

pReserved2
Zur zukünftigen Verwendung reserviert.

VendorOptionsSize
Eingabe. Die Länge des Felds mit den Lieferantenoptionen. Die Länge darf 65535 Byte nicht überschreiten.

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

Parallelism
Eingabe. Grad der partitionsinternen Parallelität (Anzahl Puffermanipulatoren)

RestoreInfoSize
Zur zukünftigen Verwendung reserviert.

pRestoreInfo
Zur zukünftigen Verwendung reserviert.

ContainerPageListSize
Zur zukünftigen Verwendung reserviert.

pContainerPageList
Zur zukünftigen Verwendung reserviert.

pReserved3
Zur zukünftigen Verwendung reserviert.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur *sqlca*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

REXX-API-Syntax

```
RESTORE DATABASE source-database-alias [USING :value] [USER username USING password]
[TABLESPACE :tablespacenames] [ONLINE | HISTORY FILE ]
[LOAD shared-library [OPTIONS vendor-options] [OPEN num-sessions SESSIONS] |
 FROM :source-area | USE TSM [OPEN num-sessions SESSIONS] | USER_EXIT]
[TAKEN AT timestamp] [TO target-directory] [INTO target-database-alias]
[ACTION caller-action] [WITH num-buffers BUFFERS] [BUFFERSIZE buffer-size]
[WITHOUT ROLLING FORWARD] [PARALLELISM parallelism-degree]
```

REXX-API-Parameter

source-database-alias
Aliasname der Quellendatenbank, aus der das Datenbanksicherungsimage erstellt wurde.

value Eine zusammengesetzte REXX-Host-Variable, an die die Datenbankwiederherstellungsinformationen zurückgegeben werden. Im Folgenden steht XXX für den Namen der Host-Variablen:

XXX.0	Anzahl Elemente in der Variablen (immer 1)
XXX.1	Eine Anwendungs-ID, die den Agenten angibt, der die Anwendung bedient

username
Gibt den für eine Verbindung verwendeten Benutzernamen an.

password
Das Kennwort, das verwendet wird, um den Benutzernamen zu authentifizieren.

tablespacenames
Eine zusammengesetzte REXX-Host-Variable, die eine Liste der wiederherzustellenden Tabellenbereiche enthält. Im Folgenden ist XXX der Name der Host-Variablen:

XXX.0	Anzahl wiederherzustellender Tabellenbereiche
XXX.1	Erster Tabellenbereichsname
XXX.2	Zweiter Tabellenbereichsname
XXX.3	und so weiter.

HISTORY FILE
Gibt an, dass nur die Protokolldatei aus dem Sicherungsimage wiederhergestellt werden soll.

shared-library
Der Name der gemeinsam benutzten Bibliothek (DLL unter Windows-Betriebssystemen oder OS/2), die die Wiederherstellungs-E/A-Funktionen des Lieferanten enthält, die verwendet werden sollen. Kann den vollständigen Pfad enthalten. Wenn kein vollständiger Pfad angegeben wird, wird standardmäßig der Pfad verwendet, in dem sich das Benutzer-Exit-Programm befindet.

vendor-options
Erforderliche Informationen für die Funktionen anderer Lieferanten.

num-sessions
Die Anzahl der E/A-Sitzungen, die mit Tivoli Storage Manager (TSM) oder einem Produkt eines anderen Lieferanten verwendet werden sollen.

source-area
Eine zusammengesetzte REXX-Host-Variable, die angibt, in welchem Verzeichnis oder auf welcher Einheit sich das Sicherungsimage befindet. Der Standardwert ist das aktuelle Verzeichnis. Unter Windows-Betriebssystemen oder OS/2 können die Einträge nur Verzeichnispfade und keine Bandeinheitennamen sein.

timestamp
Die Zeitmarke des Datenbanksicherungsimage

target-directory
Das Zieldatenbankverzeichnis

target-database-alias
Der Aliasname der Zieldatenbank. Wenn die Zieldatenbank nicht vorhanden ist, wird sie erstellt.

caller-action
Gibt die auszuführende Aktion an. Gültige Werte:

SQLUD_RESTORE
Die Wiederherstellungsoperation starten.

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

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

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

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

SQLUD_PARM_CHECK
Parameter auswerten, ohne die Wiederherstellungsoperation auszuführen.

SQLUD_RESTORE_STORDEF
Erstaufruf. Eine Neudefinition eines Tabellenbereichsbehälters anfordern.

num-buffers
Anzahl zu verwendender Sicherungspuffer

buffer-size
Sicherungspuffergröße in 4 KB großen Zuordnungseinheiten. Der Mindestwert ist 16 Einheiten.

parallelism-degree
Grad der partitionsinternen Parallelität (Anzahl Puffermanipulatoren)

Beispiele für Wiederherstellungssitzungen

CLP-Beispiele

Es folgt ein typisches Beispielszenario für die umgeleitete Wiederherstellung einer Datenbank mit dem Aliasnamen MYDB:

1. Setzen Sie einen Befehl RESTORE DATABASE mit der Option REDIRECT ab.

   ```
   db2 restore db mydb 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 mydb abort
   ```

2. Setzen Sie für jeden Tabellenbereich, dessen Behälter erneut definiert werden müssen, einen Befehl SET TABLESPACE CONTAINERS ab. Beispiel für OS/2:

   ```
   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 mydb 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.

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 kdr use tsm
(Mo) backup db kdr online incremental delta use tsm
(Di) backup db kdr online incremental delta use tsm
(Mi) backup db kdr online incremental use tsm
(Do) backup db kdr online incremental delta use tsm
(Fr) backup db kdr online incremental delta use tsm
(Sa) backup db kdr 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 kdr incremental automatic taken at (Do)
```

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

```
restore db kdr incremental taken at (Do)
restore db kdr incremental taken at (So)
restore db kdr incremental taken at (Mi)
restore db kdr incremental taken at (Do)
```

Ein Beispiel eines DB2-Befehls-Scripts sowie Informationen zur Verwendung finden Sie in „Anhang F. Befehlszeilen-Script zur Wiederherstellung" auf Seite 477.

API-Beispiele

Beispielprogramme, die DB2-APIs und eingebettete SQL-Aufrufe enthalten, sowie Informationen zu deren Verwendung finden Sie in „Anhang E. Beispiele für Wiederherstellungsprogramme" auf Seite 411.

Optimieren der Leistung des Wiederherstellungsdienstprogramms

Gehen Sie wie folgt vor, um die zur Durchführung einer Wiederherstellungsoperation benötigte Zeit zu verringern:

- Erhöhen Sie die Größe der Wiederherstellungspuffer.

  Die Größe der Wiederherstellungspuffer muss eine positive ganze Zahl und ein Vielfaches der Puffergröße sein, die bei der Sicherungsoperation verwendet wurde. Wird eine nicht korrekte Puffergröße angegeben, erhalten die zugeordneten Puffer die kleinstmögliche Größe.
- Erhöhen Sie die Anzahl von Puffern.

  Der Wert, den Sie angeben, muss ein Vielfaches der Anzahl der Seiten sein, die Sie für den Sicherungspuffer angegeben haben. Die minimale Anzahl beträgt 16 Seiten.

Wiederherstellungsdienstprogramm - 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.
- Bei Verwendung der DB2-Steuerzentrale können keine Sicherungsimages wiederhergestellt werden, die mit einer früheren Version von DB2 erstellt wurden.

Wiederherstellungsdienstprogramm - Einschränkungen

- 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.
- Unter OS/2 ist eine partielle oder selektive Wiederherstellung nicht möglich, wenn zur Wiederherstellung ein Benutzer-Exit-Programm verwendet wird.
- Wenn die Codepage der wiederherzustellenden Datenbank nicht mit der für eine Anwendung verfügbaren Codepage übereinstimmt, oder wenn der Datenbankmanager die Umsetzung zwischen der Codepage der Datenbank und der Codepage der Anwendung nicht unterstützt, ist die wiederhergestellte Datenbank nicht verwendbar.

Wiederherstellungsdienstprogramm - Fehlerbehebung

Es ist möglich, dass die Fehlermeldung SQL0970N zurückgegeben wird, wenn Sie versuchen, ein Sicherungsimage in eine neue Datenbank auf einem anderen System wiederherzustellen und in Ihrer Originaldatenbank Tabellenbereiche unter Verwendung eines absoluten Pfads definiert sind. Versuchen Sie in diesem Fall, die Tabellenbereichsbehälter auf dem neuen System mit Hilfe einer umgeleiteten Wiederherstellung zu definieren. Der Fehler entsteht dadurch, dass das Wiederherstellungsdienstprogramm versucht, den absoluten Pfad und Behälter zu verwenden, die auf dem neuen System nicht vorhanden sind.

Tritt während der Wiederherstellung einer Datenbank ein Systemfehler auf, können Sie erst dann wieder eine Verbindung zur Datenbank herstellen, wenn Sie das Wiederherstellungsdienstprogramm erneut aufgerufen und die Wiederherstellung erfolgreich beendet haben. Wenn ein Systemfehler während der Wiederherstellung eines Tabellenbereichs auftritt, ist lediglich der Tabellenbereich unbrauchbar, der gerade wiederhergestellt wurde. Die anderen Tabellenbereiche in der Datenbank können weiterhin verwendet werden.

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 Befehls DB2 ROLLFORWARD DATABASE müssen Sie lediglich den Aliasnamen der Datenbank angeben, die Sie aktualisierend wiederherstellen wollen. Zum Beispiel:

```
db2 rollforward db sample 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.
```

Eine Erläuterung dieser Felder finden Sie in „ROLLFORWARD DATABASE, Befehl" auf Seite 171.

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 einer Netzstromunterbrechung).

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 aktualisie-

renden 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 durchführen, steht dieser während der Operation nicht zur Verfügung. Die anderen Tabellenbereiche in der Datenbank *sind* jedoch verfügbar.

Wenn Sie eine Datenbank erstellen, wird für diese Datenbank zunächst nur die Umlaufprotokollierung aktiviert. Das bedeutet, dass die Protokolle wiederverwendet und nicht gespeichert oder archiviert werden. Bei Verwendung der Umlaufprotokollierung ist eine aktualisierende Wiederherstellung nicht möglich: es kann lediglich eine Wiederherstellung nach einem Systemabsturz bzw. eine Versionswiederherstellung durchgeführt werden (siehe „Wiederherstellungsprotokolle" auf Seite 36). Archivprotokolldateien erfassen die Änderungen, die nach dem Erstellen eines Sicherungsimage an einer Datenbank vorgenommen werden. Sie können diese Protokollierung (und aktualisierende Wiederherstellung) aktivieren, indem Sie den Datenbankkonfigurationsparameter *logretain* auf `RECOVERY` und/oder den Datenbankkonfigurationsparameter *userexit* auf `YES` setzen. Der Standardwert für beide Parameter lautet `NO`, da anfangs noch kein Sicherungsimage vorhanden ist, das zum Wiederherstellen der Datenbank verwendet werden kann. Wenn Sie den Wert eines oder beider Parameter ändern, erhält die Datenbank erneut den Status

Sicherung anstehend, und Sie müssen offline ein Sicherungsimage der Datenbank erstellen, bevor Sie sie weiter verwenden können.

Weitere Informationen zu den Datenbankkonfigurationsparametern, die sich auf die Protokollierungsfunktionen beziehen, finden Sie in „Konfigurationsparameter für die Datenbankprotokollierung" auf Seite 44.

Für die Verwendung des Dienstprogramms zur aktualisierenden Wiederherstellung erforderliche Zugriffsrechte und Berechtigungen

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf sie zuzugreifen. Berechtigungsstufen stellen eine Methode dar, Berechtigungen sowie höhere 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

Vor der Verwendung des Dienstprogramms zur aktualisierenden Wiederherstellung

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.

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

Aufrufen des Dienstprogramms zur aktualisierenden Wiederherstellung

Das Dienstprogramm zur aktualisierenden Wiederherstellung kann wie folgt aufgerufen werden:

- Über den Befehlszeilenprozessor (CLP).

  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
  ```

- Über das Notizbuch **Datenbank aktualisierend wiederherstellen** in der Steuerzentrale. 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.

Basisinformationen zur Steuerzentrale finden Sie im Handbuch *Systemverwaltung*. Zusatzinformationen bietet die Onlinehilfefunktion der Steuerzentrale.

- Über die Anwendungsprogrammierschnittstelle (API) **sqluroll**. Informationen zu dieser API finden Sie in „API zur aktualisierenden Wiederherstellung von Datenbanken" auf Seite 178. Basisinformationen zum Erstellen von Anwendungen, die DB2-Verwaltungs-APIs enthalten, finden Sie im Handbuch *Application Building Guide*.

In einer partitionierten Datenbankumgebung muss das Dienstprogramm zur aktualisierenden Wiederherstellung vom Katalogknoten der Datenbank aus aufgerufen werden.

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 Tabellenbereichsebene die Möglichkeit, kritische Teile der Datenbank häufiger als andere Teile zu sichern, und sind weniger zeitintensiv als Sicherungen der gesamten Datenbank.

Aktualisierendes Wiederherstellen von Änderungen in einem Tabellenbereich

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.

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 Tabelle reorganisiert wurde, 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.

Aktualisierendes Wiederherstellen von Änderungen in einem Tabellenbereich

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 Übersichtstabelle, die sich in einem anderen Tabellenbereich befindet
- Eine Übersichtstabelle für eine Tabelle in einem anderen Tabellenbereich

In diesem Fall müssen Sie beide Tabellenbereiche bis zum selben Zeitpunkt aktualisierend wiederherstellen. Wenn Sie dies nicht tun, wird die Übersichtstabelle am Ende der aktualisierenden Wiederherstellung in den Status *Überprüfung anstehend* versetzt.

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, befinden sich am Ende der aktualisierenden Wiederherstellung bis zu einem bestimmten Zeitpunkt beide Tabellenbereiche im Status *Überprüfung anstehend*. 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 bis zu einem bestimmten Zeitpunkt von Tabellenbereichen 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.)

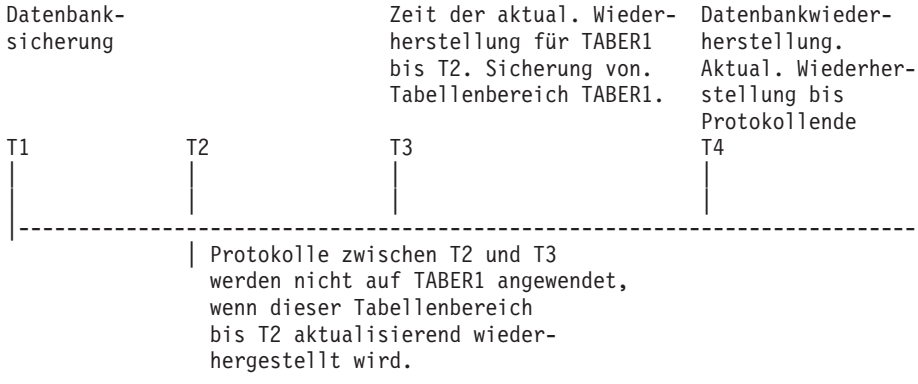


Abbildung 17. Sicherungsanforderung für Tabellenbereiche

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 Sicherungsimage 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 *Wie-*

derherstellung 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.

Aktualisierendes Wiederherstellen von Änderungen in einem 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.

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 DB2-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.

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 (siehe Handbuch *SQL Reference*). 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.

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.

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 (siehe „LIST HISTORY" auf Seite 366). 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, die die Tabellendaten enthalten, geschrieben werden können. 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 niedrigeres 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.

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 Tabellenbereiche für lange Objektdaten 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 Tabellenbereiche für lange Objektdaten.
- Die den Zeilenarten 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.

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.

Verwenden der Datei mit den Angaben zur Speicherposition der Ladekopie

Anhand der Registrierungsvariablen 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 Informationen zu den folgenden Punkten:

- Datenträgertyp
- Anzahl der zu verwendenden Datenträgereinheiten
- Speicherposition der Ladekopie, die bei einer Ladeoperation für eine Tabelle generiert wurde
- Dateiname der Ladekopie, falls 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 partitionierten Datenbankumgebung muss sich die Registrierungsvariable DB2LOADREC in der Datei `db2profile` befinden.
2. In einer partitionierten Datenbankumgebung muss die Datei der Ladekopie auf jedem Datenbankpartitionsserver vorhanden und der Dateiname (einschließlich des Pfads) derselbe sein.

3. Ist ein Eintrag in der Datei, die mit Registrierungsvariablen DB2LOADREC angegeben wurde, ungültig, wird die alte Datei mit den Angaben zur Speicherposition der Ladekopie verwendet, um Ersatzdaten für den ungültigen Eintrag bereitzustellen.

Folgende Informationen werden in der Datei mit den Angaben zur Speicherposition bereitgestellt. Die ersten fünf Parameter müssen gültige Werte aufweisen und werden zum Identifizieren der Ladekopie verwendet. Die gesamte Struktur wird für jede aufgezeichnete Ladekopie wiederholt. Zum 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 der Puffer für die Wiederherstellung
SESsionnumber  NULL             * Anzahl der Sitzungen für die Wiederherstellung
TYPeofmedia    L                * Datenträgertyp - L für lokale Einheit
                                                   A für TSM
                                                  O für andere Lieferanten
LOCationnumber 3                * Anzahl der 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
SES            NULL
BUF            NULL
TYP            A
TIM            19940325192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            TORONTO
SES            NULL
BUF            NULL
TYP            O
SHRlib         /@sys/lib/backup_vendor.a
```

Anmerkungen:

1. Für jedes Schlüsselwort sind die ersten drei Zeichen wichtig. Alle Schlüsselwörter sind in der angegebenen Reihenfolge erforderlich. Leerzeilen werden nicht akzeptiert.
2. Die Zeitmarke hat das Format *jjjjmmtthhmmss*.
3. Alle Felder sind verbindlich, mit Ausnahme von BUF und SES, die den Wert NULL haben können. Ist SES gleich NULL, wird der vom Konfigurationsparameter *numloadrecses* angegebene Wert verwendet. Ist BUF gleich NULL, ist der Standardwert SES+2.

4. Auch wenn lediglich ein Eintrag in der Datei mit den Angaben zur Speicherposition ungültig ist, wird die vorherige Datei mit den Angaben zur Speicherposition der Ladekopie zur Bereitstellung dieser Einträge verwendet.
5. Als Datenträgertyp kann eine lokale Einheit (`L` für Band, Platte oder Diskette), TSM (`A`) oder ein Datenträger anderer Lieferanten (`O`) angegeben werden. 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 `O`, ist der Name der gemeinsam benutzten Bibliothek erforderlich. Weitere Informationen zur Verwendung von TSM oder der Produkte anderer Lieferanten als Sicherungsdatenträger finden Sie in „Anhang G. Tivoli Storage Manager" auf Seite 485.
6. Der Parameter SHRlib zeigt auf eine Bibliothek, die eine Funktion zum Speichern der Ladekopiedaten hat.
7. Wenn Sie eine LOAD-Operation mit der Option COPY NO ausführen und nach Ausführung der Operation keine Sicherungskopie der Datenbank oder der betroffenen Tabellenbereiche erstellen, können Sie die Datenbank oder Tabellenbereiche nicht bis zu einem bestimmten Zeitpunkt wiederherstellen, der zeitlich nach der LOAD-Operation liegt. Das heißt, Sie können keine aktualisierende Wiederherstellung ausführen, um die Datenbank bzw. die Tabellenbereiche auf ihren Stand nach der Ausführung der LOAD-Operation wiederherzustellen. Sie können die Datenbank bzw. die Tabellenbereiche lediglich bis zu einem Zeitpunkt wiederherstellen, der vor dem Zeitpunkt der LOAD-Operation liegt.

Wenn Sie eine bestimmte Ladekopie verwenden wollen, können Sie die Datei des Wiederherstellungsprotokolls für die Datenbank verwenden, um die Zeitmarke dieser bestimmten LOAD-Operation festzustellen. In einer Umgebung mit partitionierten Datenbanken ist die Datei des Wiederherstellungsprotokolls für jede Datenbankpartition lokal.

Weitere Informationen zum Dienstprogramm LOAD finden Sie im Handbuch *Versetzen von Daten Dienstprogramme und Referenz*.

Synchronisation 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 unter 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 Systemuhren 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 in Gebrauch ist 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, 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.

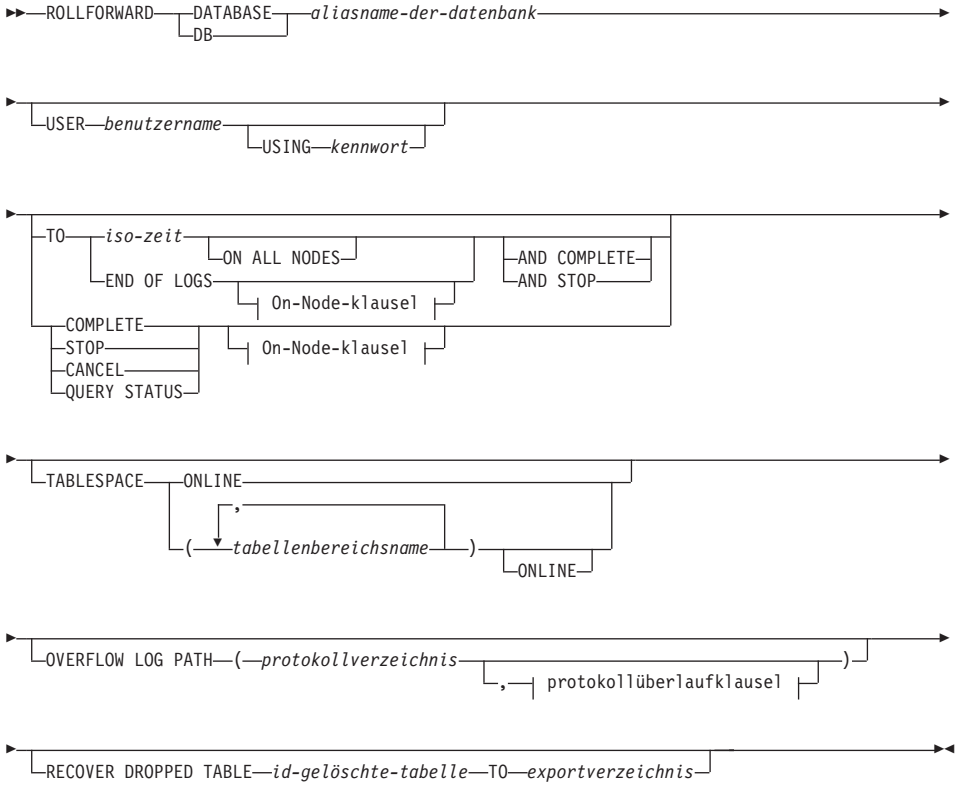
Synchronisation der Systemuhren

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. Weitere Informationen zur Einstellung des Konfigurationsparameters *max_time_diff* finden Sie im Handbuch *Systemverwaltung: Optimierung*.
- 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.

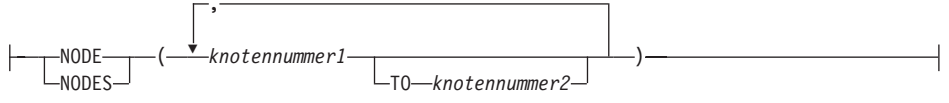
ROLLFORWARD DATABASE, Befehl

Befehlssyntax

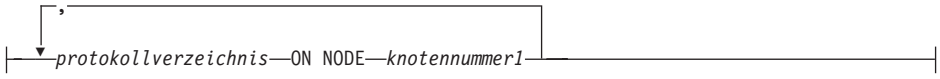


ON-NODE-Klausel:

Knotenlistenklausel:



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 UTC konvertiert.

END OF LOGS
Gibt an, dass alle festgeschriebenen Transaktionen aus allen Onlinearchivprotokolldateien angewendet werden sollen, die in dem durch den Datenbankkonfigurationsparameter *logpath* angegegenen Pfad enthalten sind.

ALL NODES
Gibt an, dass Transaktionen auf allen in der Datei `db2nodes.cfg` angegebenen Knoten aktualisierend wiederhergestellt werden sollen. Dies ist die Standardeinstellung, falls keine Knotenklausel angegeben ist.

EXCEPT
Gibt an, dass Transaktionen auf allen in der Datei `db2nodes.cfg` angegebenen Knoten aktualisierend wiederhergestellt werden sollen, mit Ausnahme der Knoten in der Knotenliste.

ON NODE / ON NODES
Stellt die Datenbank auf einer Knotenmenge aktualisierend wieder her.

knotennummer1
Gibt eine Knotennummer in der Knotenliste an

knotennummer2
Gibt die zweite Knotennummer an, damit alle Knoten von *knotennummer1* bis einschließlich *knotennumer2* in der Knotenliste enthalten sind.

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 Knoten in den Status *Wiederherstellung anstehend*, auf denen die aktualisierende Wiederherstellung gestartet wurde:

- Wenn keine aktualisierende Wiederherstellungsoperation für eine Datenbank ausgeführt wird, (das heißt, die Datenbank befindet sich im Status *Aktualisierende Wiederherstellung anstehend*,) versetzt diese Option die Datenbank in den Status *Wiederherstellung anstehend*.
- Wenn keine aktualisierende Wiederherstellungsoperation für einen Tabellenbereich ausgeführt wird, (das heißt, die Tabellenbereiche befinden sich im Status *Aktualisierende Wiederherstellung anstehend*,) muss eine Tabellenbereichsliste angegeben werden. Alle Tabellenbereiche in der Liste werden in den Status *Wiederherstellung anstehend* versetzt.
- Wenn eine aktualisierende Tabellenbereichswiederherstellung *ausgeführt* wird, (das heißt, mindestens ein Tabellenbereich befindet sich im Status *Aktualisierende Wiederherstellung läuft*,) 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 fortschreitende aktualisierende Wiederherstellungsoperation nicht beendet werden kann, weil einige der Tabellenbereiche in den Status *Aktualisierende Wiederherstellung anstehend* oder *Wiederherstellung anstehend* versetzt worden sind. Verwenden Sie den Befehl LIST TABLESPACES, um die Tabellenbereiche anzugeben, die sich im Status *Aktualisierende Wiederherstellung läuft* oder *Aktualisierende Wiederherstellung anstehend* befinden, anzuzeigen, wenn Sie sich nicht sicher sind.

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 Knoten zurückgegeben. Die zurückgegebenen Informationen umfassen die folgenden Felder:

Knotennummer

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 Knoten vorhanden sein, die aktualisierend wiederhergestellt werden. Wenn er vorhanden *ist*, muss er sich im korrekten Zustand 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 qualifizierte) Standardüberlaufprotokollpfad *für alle Knoten*. Für Datenbanken mit einer Partition kann ein relativer Überlaufprotokollpfad angegeben werden. 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.

protokollverzeichnis ON NODE
In einer Umgebung mit partitionierten Datenbanken kann ein anderer Protokollpfad den Standardüberlaufprotokollpfad für einen bestimmten Knoten überschreiben.

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 (siehe „LIST HISTORY" auf Seite 366).

TO exportverzeichnis
Gibt ein Verzeichnis an, in das Dateien geschrieben werden sollen, die die Tabellendaten enthalten. Alle Knoten müssen auf das Verzeichnis zugreifen können.

API zur aktualisierenden Wiederherstellung von Datenbanken

C-API-Syntax

```
/* File: sqlutil.h */
/* API: Rollforward Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqluroll (
    struct rfwd_input * pRfwdInput,
    struct rfwd_output * pRfwdOuput,
    struct sqlca * pSqlca);
/* ... */
```

Generische API-Syntax

```
/* File: sqlutil.h */
/* API: Rollforward Database */
/* ... */
SQL_API_RC SQL_API_RN
  sqlgroll (
    struct grfwd_input * grfwdin,
    struct rfwd_output * rfwdout,
    struct sqlca * sqlca);

SQL_STRUCTURE grfwd_input
{
   unsigned short DbAliasLen,
   unsigned short StopTimeLen,
   unsigned short UserNameLen,
   unsigned short PasswordLen,
   unsigned short OverflowLogPathLen,
   unsigned short ReportFileLen,      /* ANM.: Dieser Parameter wird für */
                                      /*       DB2 Data Links Manager    */
                                      /*       nicht mehr verwendet.     */
   sqluint32 Version,
   char * pDbAlias,
   unsigned short CallerAction,
   char * pStopTime,
   char * pUserName,
   char * pPassword,
   char * pOverflowLogPath,
   unsigned short NumChngLgOvrflw,
   struct sqlurf_newlogpath * pChngLogOvrflw,
   unsigned short ConnectMode,
   struct sqlu_tablespace_bkrst_list * pTablespaceList,
   short AllNodeFlag,
   short NumNodes,
   SQL_PDB_NODE_TYPE * pNodeList,
   short NumNodeInfo,
   unsigned short DlMode,             /* ANM.: Dieser Parameter wird für */
                                      /*       DB2 Data Links Manager    */
                                      /*       nicht mehr verwendet.     */
   char * pReportFile,                /* ANM.: Dieser Parameter wird für */
                                      /*       DB2 Data Links Manager    */
                                      /*       nicht mehr verwendet.     */
   char * pDroppedTblID,
   char * pExportDir
}
/* ... */
```

API-Parameter

pRfwdInput
Eingabe. Ein Zeiger auf die Struktur *rfwd_input*. Weitere Informationen zu dieser Struktur finden Sie in „Datenstruktur: RFWD-INPUT" auf Seite 188.

pRfwdOutput
Ausgabe. Ein Zeiger auf die Struktur *rfwd_output*. Weitere Informationen zu dieser Struktur finden Sie in „Datenstruktur: RFWD-OUTPUT" auf Seite 191.

DbAliasLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Aliasnamens der Datenbank in Byte darstellt.

StopTimeLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge der Stoppzeitparameters in Byte darstellt. Wird auf null gesetzt, wenn keine Stoppzeit angegeben wird.

UserNameLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Benutzernamens in Byte darstellt. Wird auf null gesetzt, wenn kein Benutzername angegeben wird.

PasswordLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge des Kennworts in Byte darstellt. Wird auf null gesetzt, wenn kein Kennwort angegeben wird.

OverflowLogPathLen
Eingabe. Eine 2 Byte große ganze Zahl ohne Vorzeichen, die die Länge der Überlaufprotokollpfads in Byte darstellt. Wird auf null gesetzt, wenn kein Überlaufprotokollpfad angegeben wird.

ReportFileLen
Eingabe. Dieser Parameter wird zurzeit nicht verwendet und sollte auf null gesetzt werden.

Version
Eingabe. Die Versions-ID des Parameters für eine aktualisierende Wiederherstellung. Sie ist als SQLUM_RFWD_VERSION definiert.

pDbAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der Datenbank enthält. Dies ist der im Systemdatenbankverzeichnis katalogisierte Aliasname.

CallerAction
Eingabe. Gibt die auszuführende Aktion an. Gültige Werte (definiert in `sqlutil`) sind:

SQLUM_ROLLFWD
Eine aktualisierende Wiederherstellung bis zu dem in *pPointInTime* angegebenen Zeitpunkt ausführen. Für eine Datenbankfehlerbehebung durch aktualisierende Wiederherstellung bleibt die Datenbank im Status *Aktualisierende Wieder-*

herstellung anstehend. Für eine aktualisierende Wiederherstellung auf Tabellenbereichsebene bis zu einem Zeitpunkt bleiben die Tabellenbereiche im Status *Aktualisierende Wiederherstellung läuft*.

SQLUM_STOP
Die Fehlerbehebung durch 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 SQLUM_COMPLETE ist gleichbedeutend.

SQLUM_ROLLFWD_STOP
Eine aktualisierende Wiederherstellung bis zu dem in *pPointInTime* angegebenen Zeitpunkt ausführen und die Fehlerbehebung durch aktualisierende Wiederherstellung beenden. Der Status *Aktualisierende Wiederherstellung anstehend* der Datenbank oder des Tabellenbereichs wird inaktiviert. Der Parameter SQLUM_ROLLFWD_COMPLETE ist gleichbedeutend.

SQLUM_QUERY
Die Werte für *pNextArcFileName*, *pFirstDelArcFileName*, *pLastDelArcFileName* und *pLastCommitTime* abfragen. Den Datenbankstatus und eine Knotenzahl zurückgeben.

SQLUM_PARM_CHECK
Parameter auswerten, ohne die aktualisierende Wiederherstellungsoperation auszuführen.

SQLUM_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 Wiederherstellungsoperation tatsächlich läuft. Sie kann verwendet werden, wenn die Operation angehalten ist (d. h. auf ein STOP wartet) oder wenn während der aktualisierenden Wiederherstellungsoperation ein Systemfehler aufgetreten ist. Die Option sollte mit Bedacht verwendet werden.

Eine aktualisierende Wiederherstellung von Datenbanken könnte eine Wiederherstellungsoperation 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:

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

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

SQLUM_LOADREC_TERMINATE
Alle von der Wiederherstellung verwendeten Einheiten beenden.

pStopTime
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 `SQLUM_QUERY`, `SQLUM_PARM_CHECK` und alle Wiederherstellungsaktionen des aufrufenden Programms (`SQLUM_LOADREC_`*xxx*) NULL sein.

pUserName
Eingabe. Eine Zeichenfolge, die den Benutzernamen der Anwendung enthält. Kann NULL sein.

pPassword
Eingabe. Eine Zeichenfolge, die das Kennwort für den angegebenen Benutzernamen (wenn vorhanden) enthält. Kann NULL sein.

pOverflowLogPath
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 (siehe „sqlfxdb - Get Database Configuration" im Handbuch *Administrative API Reference*), bevor sie durch dieses 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 Tabellenbereichswiederherstellung 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 Partition kann der Überlaufprotokollpfad relativ sein, wenn der Server lokal ist.

NumChngLgOvrflw
Nur MPP. Die Anzahl geänderter Überlaufprotokollpfade. Diese neuen Protokollpfade überschreiben nur den Standardüberlaufprotokollpfad für den angegebenen Knoten.

pChngLogOvrflw
Nur MPP. Ein Zeiger auf eine Struktur, die die vollständig qualifizierten Namen der geänderten Überlaufprotokollpfade enthält. Diese neuen Protokollpfade überschreiben nur den Standardüberlaufprotokollpfad für den angegebenen Knoten.

ConnectMode
Eingabe. Gültige Werte (definiert in `sqlutil`) sind:

SQLUM_OFFLINE
Aktualisierende Offlinewiederherstellung. Dieser Wert muss für eine aktualisierende Datenbankwiederherstellung angegeben werden.

SQLUM_ONLINE
Aktualisierende Onlinewiederherstellung.

pTablespaceList
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.

AllNodeFlag
Nur MPP. Eingabe. Gibt an, ob die aktualisierende Wiederherstellungsoperation auf alle in `db2nodes.cfg` definierten Knoten angewendet werden soll. Gültige Werte sind:

SQLURF_NODE_LIST
Auf Knoten in einer Knotenliste anwenden, die in *pNodeList* übergeben wird.

SQLURF_ALL_NODES
Auf alle Knoten anwenden. *pNodeList* sollte NULL sein. Dies ist der Standardwert.

SQLURF_ALL_EXCEPT
Auf alle Knoten anwenden, mit Ausnahme der Knoten in einer in *pNodeList* übergebenen Knotenliste.

SQLURF_CAT_NODE_ONLY
Nur auf den Katalogknoten anwenden. *pNodeList* sollte NULL sein.

NumNodes
Eingabe. Gibt die Anzahl Knoten im Bereich *pNodeList* an.

pNodeList
Eingabe. Ein Zeiger auf einen Bereich mit Knotennummern, auf denen eine aktualisierende Wiederherstellung ausgeführt werden soll.

NumNodeInfo
Eingabe. Definiert die Größe des Ausgabeparameters *pNodeInfo*, der groß genug sein muss, um Statusinformationen für jeden aktualisierend wiederherzustellenden Knoten zu enthalten. In einer Umgebung mit einer einzelnen Partition sollte dieser Parameter auf 1 gesetzt werden. Der Wert dieses Parameters sollte gleich der Knotenzahl sein, für die diese API aufgerufen wird.

DlMode
Eingabe. Dieser Parameter wird zurzeit nicht verwendet und sollte auf null gesetzt werden.

pReportFile
Eingabe. Dieser Parameter wird zurzeit nicht verwendet und sollte auf NULL gesetzt werden.

pDroppedTblID
Eingabe. Eine Zeichenfolge, die die ID der gelöschten Tabelle enthält, deren Wiederherstellung versucht wird.

pExportDir
Eingabe. Das Verzeichnis, in das die gelöschten Tabellendaten exportiert werden.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur *sqlca*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

REXX-API-Syntax

```
ROLLFORWARD DATABASE database-alias [USING :value] [USER username USING password]
[rollforward_action_clause | load_recovery_action_clause]
where rollforward_action_clause stands for:
    { TO point-in-time [AND STOP] |
      {
        [TO END OF LOGS [AND STOP] | STOP | CANCEL | QUERY STATUS | PARM CHECK }
        [ON {:nodelist | ALL NODES [EXCEPT :nodelist]}]
      }
    }
    [TABLESPACE {ONLINE |:tablespacenames [ONLINE]} ]
    [OVERFLOW LOG PATH default-log-path [:logpaths]]
and load_recovery_action_clause stands for:
    LOAD RECOVERY { CONTINUE | DEVICE_TERMINATE | TERMINATE }
```

REXX-API-Parameter

database-alias
Aliasname der aktualisierend wiederherzustellenden Datenbank.

value Eine zusammengesetzte REXX-Host-Variable, die die Ausgabewerte enthält. Im Folgenden steht XXX für den Namen der Host-Variablen:

XXX.0	Anzahl Elemente in der Variablen
XXX.1	Die Anwendungs-ID
XXX.2	Anzahl der von den Knoten empfangenen Antworten
XXX.2.1.1	Erste Knotennummer
XXX.2.1.2	Erste Statusinformation
XXX.2.1.3	Erste als nächste benötigte Archivierungsdatei
XXX.2.1.4	Erste zu löschende erste Archivierungsdatei
XXX.2.1.5	Erste zu löschende letzte Archivierungsdatei
XXX.2.1.6	Erste letzte COMMIT-Zeit
XXX.2.2.1	Zweite Knotennummer
XXX.2.2.2	Zweite Statusinformation
XXX.2.2.3	Zweite als nächste benötigte Archivierungsdatei
XXX.2.2.4	Zweite zu löschende erste Archivierungsdatei
XXX.2.2.5	Zweite zu löschende letzte Archivierungsdatei
XXX.2.2.6	Zweite letzte COMMIT-Zeit
XXX.2.3.x	und so weiter.

username
Gibt den Benutzernamen an, unter dem die Datenbank aktualisierend wiederhergestellt werden soll.

password
Das Kennwort, das verwendet wird, um den Benutzernamen zu authentifizieren.

point-in-time
Eine Zeitmarke in ISO-Format, *jjjj-mm-tt-hh.mm.ss.nnnnnn* (Jahr, Monat, Tag, Stunde, Minuten, Sekunden, Mikrosekunden), die in Weltzeit (Coordinated Universal Time, UTC) ausgedrückt wird.

tablespacenames
Eine zusammengesetzte REXX-Host-Variable, die eine Liste der aktualisierend wiederherzustellenden Tabellenbereiche enthält. Im Folgenden ist XXX der Name der Host-Variablen:

XXX.0	Anzahl aktualisierend wiederherzustellender Tabellenbereiche
XXX.1	Erster Tabellenbereichsname
XXX.2	Zweiter Tabellenbereichsname
XXX.x	und so weiter.

default-log-path
Der Standardüberlaufprotokollpfad, der während der Wiederherstellung nach archivierten Protokollen durchsucht werden soll.

logpaths
Eine zusammengesetzte REXX-Host-Variable, die eine Liste der alternativen Protokollpfade enthält, die während der Wiederherstellung nach archivierten Protokollen durchsucht werden sollen. Im Folgenden ist XXX der Name der Host-Variablen:

XXX.0	Anzahl geänderter Überlaufprotokollpfade
XXX.1.1	Erster Knoten
XXX.1.2	Erster Überlaufprotokollpfad
XXX.2.1	Zweiter Knoten
XXX.2.2	Zweiter Überlaufprotokollpfad
XXX.3.1	Dritter Knoten
XXX.3.2	Dritter Überlaufprotokollpfad
XXX.x.1	und so weiter.

nodelist

Eine zusammengesetzt REXX-Host-Variable, die eine Knotenliste enthält. Im Folgenden ist XXX der Name der Host-Variablen:

XXX.0	Anzahl Knoten
XXX.1	Erster Knoten
XXX.2	Zweiter Knoten
XXX.*x*	und so weiter.

Datenstruktur: RFWD-INPUT

Diese Struktur wird verwendet, um Informationen an die API zur aktualisierenden Wiederherstellung zu übergeben (siehe „API zur aktualisierenden Wiederherstellung von Datenbanken" auf Seite 178.

Tabelle 8. Felder in der Struktur RFWD-INPUT

Feldname	Datentyp	Beschreibung
VERSION	sqluint32	Version für die aktualisierende Wiederherstellung
PDBALIAS	Zeiger	Aliasname der Datenbank
CALLERACTION	UNSIGNED SHORT	Aktion
PSTOPTIME	Zeiger	Stoppzeit
PUSERNAME	Zeiger	Benutzername
PPASSWORD	Zeiger	Kennwort
POVERFLOWLOGPATH	Zeiger	Überlaufsprotokollpfad
NUMCHNGLGOVRFLW	UNSIGNED SHORT	Anzahl geänderter Überlaufprotokollpfade (nur MPP)
PCHNGLOGOVRFLW	Struktur	Geänderte Überlaufprotokollpfade (nur MPP)
CONNECTMODE	UNSIGNED SHORT	Verbindungsmodus
PTABLESPACELIST	Struktur	Ein Zeiger auf eine Liste von Tabellenbereichsnamen. Weitere Informationen zu dieser Struktur finden Sie in „Datenstruktur: SQLU-TABLESPACE-BKRST-LIST" auf Seite 122.
ALLNODEFLAG	SHORT	Markierung für alle Knoten
NUMNODES	SHORT	Größe der Knotenliste
PNODELIST	Zeiger	Liste der Knotennummern
NUMNODEINFO	SHORT	Größe von *pNodeInfo* in der Datenstruktur RFWD-OUTPUT (siehe „Datenstruktur: RFWD-OUTPUT" auf Seite 191)
DLMODE	UNSIGNED SHORT	Dieser Parameter wird momentan nicht verwendet.
PREPORTFILE	Zeiger	Dieser Parameter wird momentan nicht verwendet.
PDROPPEDTBLID	Zeiger	Eine Zeichenfolge mit der ID der gelöschten Tabelle, deren Wiederherstellung versucht wird
PEXPORTDIR	Zeiger	Das Verzeichnis, in das die gelöschten Tabellendaten exportiert werden
NODENUM	SQL_PDB_NODE_TYPE	Knotennummer
PATHLEN	UNSIGNED SHORT	Länge des neuen Protokollpfads
LOGPATH	CHAR(255)	Neuer Überlaufprotokollpfad

Sprachsyntax

C-Struktur

```
/* File: sqlutil.h */
/* Structure: RFWD-INPUT */
/* ... */
SQL_STRUCTURE rfwd_input
{
  sqluint32              version;
  char                   *pDbAlias;
  unsigned short         CallerAction;
  char                   *pStopTime;
  char                   *pUserName;
  char                   *pPassword;
  char                   *pOverflowLogPath;
  unsigned short         NumChngLgOvrflw;
  struct sqlurf_newlogpath *pChngLogOvrflw;
  unsigned short         ConnectMode;
  struct sqlu_tablespace_bkrst_list *pTablespaceList;
  short                  AllNodeFlag;
  short                  NumNodes;
  SQL_PDB_NODE_TYPE      *pNodeList;
  short                  NumNodeInfo;
  unsigned short         DlMode;         /* Dieser Parameter wird */
                                         /*   momentan nicht verwendet. */
  char                   *pReportFile;   /* Dieser Parameter wird */
                                         /*   momentan nicht verwendet. */
  char                   *pDroppedTblID;
  char                   *pExportDir;
};
/* ... */

/* File: sqlutil.h */
/* Structure: SQLURF-NEWLOGPATH */
/* ... */
SQL_STRUCTURE sqlurf_newlogpath
{
  SQL_PDB_NODE_TYPE      nodenum;
  unsigned short         pathlen;
  char                   logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
};
/* ... */
```

Datenstruktur: RFWD-INPUT

COBOL-Struktur

```
* File: sqlutil.cbl
01 SQL-RFWD-INPUT.
    05 SQL-VERSION            PIC 9(9) COMP-5.
    05 SQL-DBALIAS            USAGE IS POINTER.
    05 SQL-CALLERACTION       PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-STOPTIME           USAGE IS POINTER.
    05 SQL-USERNAME           USAGE IS POINTER.
    05 SQL-PASSWORD           USAGE IS POINTER.
    05 SQL-OVERFLOWLOGPATH    USAGE IS POINTER.
    05 SQL-NUMCHANGE          PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-CHNG-LOG-OVRFLW  USAGE IS POINTER.
    05 SQL-CONNECTMODE        PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-TABLESPACE-LIST  USAGE IS POINTER.
    05 SQL-ALLNODEFLAG        PIC S9(4) COMP-5.
    05 SQL-NUMNODES           PIC S9(4) COMP-5.
    05 SQL-NODELIST           USAGE IS POINTER.
    05 SQL-NUMNODEINFO        PIC S9(4) COMP-5.
    05 SQL-DLMODE             PIC 9(4) COMP-5.  * Dieser Parameter wird
                                                 *   momentan nicht verwendet.
    05 SQL-REPORTFILE         USAGE IS POINTER. * Dieser Parameter wird
                                                 *   momentan nicht verwendet.
    05 SQL-DROPPEDTBLID       USAGE IS POINTER.
    05 SQL-EXPORTDIR       USAGE IS POINTER.
*

* File: sqlutil.cbl
01 SQLURF-NEWLOGPATH.
    05 SQL-NODENUM            PIC S9(4) COMP-5.
    05 SQL-PATHLEN            PIC 9(4) COMP-5.
    05 SQL-LOGPATH            PIC X(254).
    05 FILLER                 PIC X.
    05 FILLER                 PIC X(1).
*
```

Datenstruktur: RFWD-OUTPUT

Diese Struktur wird verwendet, um Informationen von der API zur aktualisierenden Widerherstellung von Datenbanken (siehe „API zur aktualisierenden Wiederherstellung von Datenbanken" auf Seite 178 im Handbuch) zu übergeben.

Tabelle 9. Felder in der Struktur RFWD-OUTPUT

Feldname	Datentyp	Beschreibung
PAPPLICATIONID	Zeiger	Die Adresse eines Puffers der Länge `SQLU_APPLID_LEN+1` (definiert in `sqlutil`), der eine von der API zurückgegebene Anwendungs-ID speichern soll. Anhand dieser Kennung kann mit den Datenbanksystemmonitor-APIs die Anwendung überwacht werden. Wenn diese Informationen irrelevant sind, geben Sie den Nullzeiger an. In einer Umgebung mit partitionierten Datenbanken wird nur die Anwendungs-ID für den Katalogknoten zurückgegeben.
PNUMREPLIES	Zeiger	Anzahl der empfangenen Knotenantworten. Jeder Knoten, der antwortet, füllt eine *sqlurf_info*-Struktur in *pNodeInfo* aus. In einer Umgebung mit einer Partition weist dieser Parameter den Wert 1 auf.
PNODEINFO	Struktur	Knotenantwortinformationen. Ein benutzerdefinierter Bereich von *NumNodeInfo* *sqlurf_info*-Strukturen.

Tabelle 10. Felder in der Struktur SQLURF-INFO

Feldname	Datentyp	Beschreibung
NODENUM	SQL_PDB_NODE_TYPE	Knotennummer.
STATE	LONG	Statusinformation.
NEXTARCLOG	UNSIGNED CHAR(13)	Ein zwölf Byte großer Puffer für den zurückgegebenen Namen der nächsten erforderlichen archivierten Protokolldatei. Wenn eine andere auszuführende Aktion als `SQLUM_QUERY` angegeben wird, zeigt der in diesem Feld zurückgegebene Wert an, dass beim Zugriff auf die Datei ein Fehler eingetreten ist. Mögliche Ursachen: • Die Datei wurde weder im Datenbankprotokollverzeichnis noch im Pfad gefunden, der mit dem Parameter für den Überlaufprotokollpfad angegeben ist. • Das Benutzer-Exit-Programm konnte die Archivierungsdatei nicht zurückgeben.

Tabelle 10. Felder in der Struktur SQLURF-INFO (Forts.)

Feldname	Datentyp	Beschreibung
FIRSTARCDEL	UNSIGNED CHAR(13)	Ein zwölf Byte großer Puffer für den zurückgegebenen Namen der ersten archivierten Protokolldatei, die für die Wiederherstellung nicht mehr benötigt wird. Diese Datei und alle Dateien bis einschließlich *lastarcdel* können Sie versetzen, um auf der Platte Platz freizugeben. Wenn z. B. die Werte, die in *firstarcdel* und *lastarcdel* zurückgegeben werden, `S0000001.LOG` und `S0000005.LOG` sind, können die folgenden Protokolldateien versetzt werden: • `S0000001.LOG` • `S0000002.LOG` • `S0000003.LOG` • `S0000004.LOG` • `S0000005.LOG`
LASTARCDEL	UNSIGNED CHAR(13)	Ein zwölf Byte großer Puffer für den zurückgegebenen Namen der letzten archivierten Protokolldatei, die aus dem Datenbankprotokollverzeichnis entfernt werden kann.
LASTCOMMIT	UNSIGNED CHAR(27)	Eine Zeichenfolge aus 26 Zeichen, die eine Zeitmarke im ISO-Format enthält. Dieser Wert ist die Zeitmarke der letzten Transaktion, die festgeschrieben wurde, nachdem die aktualisierende Wiederherstellung endete.

Mögliche Werte für *STATE* (definiert in `sqlutil`):

SQLURFQ_NOT_AVAILABLE
Es konnte keine Verbindung zum Knoten hergestellt werden.

SQLURFQ_NOT_RFW_PENDING
Die Datenbank befindet sich nicht im Status *Aktualisierende Wiederherstellung anstehend*.

SQLURFQ_DB_RFW_PENDING
Die Datenbank befindet sich im Status *Aktualisierende Wiederherstellung anstehend*.

SQLURFQ_TBL_RFW_PENDING
Der Tabellenbereich befindet sich im Status *Aktualisierende Wiederherstellung anstehend*.

SQLURFQ_DB_RFW_IN_PROGRESS
Die Datenbank befindet sich im Status *Aktualisierende Wiederherstellung läuft*.

SQLURFQ_TBL_RFW_IN_PROGRESS
Der Tabellenbereich befindet sich im Status *Aktualisierende Wiederherstellung läuft*

SQLURFQ_DB_RFW_STOPPING
Die aktualisierende Wiederherstellung der Datenbank wurde während der Verarbeitung einer STOP-Anforderung unterbrochen.

SQLURFQ_TBL_RFW_STOPPING
Die aktualisierende Wiederherstellung des Tabellenbereichs wurde während der Verarbeitung einer STOP-Anforderung unterbrochen.

Sprachsyntax

C-Struktur

```
/* File: sqlutil.h */
/* Structure: RFWD-OUTPUT */
/* ... */
SQL_STRUCTURE rfwd_output
{
  char            *pApplicationId;
  long            *pNumReplies;
  struct sqlurf_info *pNodeInfo;
};
/* ... */

/* File: sqlutil.h */
/* Structure: SQLURF-INFO */
/* ... */
SQL_STRUCTURE sqlurf_info
{
  SQL_PDB_NODE_TYPE      nodenum;
  long            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];
};
/* ... */
```

Datenstruktur: RFWD-OUTPUT

COBOL-Struktur

```
* File: sqlutil.cbl
01 SQL-RFWD-OUTPUT.
    05 SQL-APPLID            USAGE IS POINTER.
    05 SQL-NUMREPLIES        USAGE IS POINTER.
    05 SQL-P-NODE-INFO       USAGE IS POINTER.
*

* File: sqlutil.cbl
01 SQLURF-INFO.
    05 SQL-NODENUM           PIC S9(4) COMP-5.
    05 FILLER                PIC X(2).
    05 SQL-STATE             PIC S9(9) COMP-5.
    05 SQL-NEXTARCLOG        PIC X(12).
    05 FILLER                PIC X.
    05 SQL-FIRSTARCDEL       PIC X(12).
    05 FILLER                PIC X.
    05 SQL-LASTARCDEL        PIC X(12).
    05 FILLER                PIC X.
    05 SQL-LASTCOMMIT        PIC X(26).
    05 FILLER                PIC X.
    05 FILLER                PIC X(2).
*
```

Beispiele für Sitzungen zur aktualisierenden Wiederherstellung

CLP-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 sie ihre Namen angeben.

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 Benutzer-Exit Archivprotokolldateien speichert:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   overflow log path (/logs)
```

Beispiel 5 (MPP)

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 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 (MPP)

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 berichtet SQL4906N.

```
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 abgelehnt. 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 (MPP)

Nachdem Sie einen Tabellenbereich auf allen Knoten wiederhergestellt haben, stellen Sie ihn wie folgt aktualisierend bis zu 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 zu 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 (MPP)

Stellen Sie wie folgt einen Tabellenbereich wieder her, der sich auf den in der Datei db2nodes.cfg aufgelisteten acht Knoten (3 to 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 (MPP)

Stellen Sie wie folgt sechs kleine Tabellenbereiche aktualisierend wieder her, die sich auf einer einzelnen Knotengruppe (auf Knoten 6) 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.

Ein Beispiel eines DB2-Befehls-Scripts sowie Informationen zur Verwendung finden Sie in „Anhang F. Befehlszeilen-Script zur Wiederherstellung" auf Seite 477.

API-Beispiele

Beispielprogramme, die DB2-APIs und eingebettete SQL-Aufrufe enthalten, sowie Informationen zu deren Verwendung finden Sie in „Anhang E. Beispiele für Wiederherstellungsprogramme" auf Seite 411.

Aktualisierende Wiederherstellung - 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 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.

Aktualisierende Wiederherstellung - Fehlerbehebung

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.

Wenn das Dienstprogramm zur aktualisierenden Wiederherstellung eine zur Nichtwiederherstellbarkeit führende Operation feststellt (z. B. LOAD mit NO COPY), wird der zugeordnete Tabellenbereich in den Status *Aktualisierende Wiederherstellung anstehend* versetzt. Damit dieser Status für den Tabellenbereich wieder aufgehoben wird, müssen Sie eine Wiederherstellung von einem neueren Sicherungsimage auf Tabellenbereichs- oder Datenbankebene vornehmen.

Die Fehlermeldung SQL1271 wird als Warnung zurückgegeben, dass sich ein Tabellenbereich im Status *Wiederherstellung anstehend* oder *Aktualisierende Wiederherstellung anstehend* befindet (die Warnung wird selbst dann ausgegeben, wenn AND STOP noch nicht angefordert wurde). Geschieht dies während der aktualisierenden Wiederherstellung einer Datenbank, bedeutet dies, dass mindestens ein Tabellenbereich durch das Dienstprogramm zur aktualisierenden Wiederherstellung offline genommen wurde. Während der aktualisierenden Wiederherstellung eines *Tabellenbereichs* kann dies bedeuten, dass einer der betroffenen Tabellenbereiche durch das Dienstprogramm zur aktualisierenden Wiederherstellung offline genommen wurde oder dass ein anderer (nicht aufgelisteter) Tabellenbereich vorhanden ist, der noch aktualisierend wiederhergestellt werden muss.

Die Fehlermeldung SQL1272 wird zurückgegeben, wenn alle aktualisierend wiederhergestellten Tabellenbereiche (die entweder in der Liste angegeben wurden oder den Status der anstehenden aktualisierenden Wiederherstellung hatten) durch das Dienstprogramm zur aktualisierenden Wiederherstellung offline genommen wurden. Dies kann bedeuten, dass die Bereiche eine Tabelle enthalten, für die eine zur Nichtwiederherstellbarkeit führende LOAD-Operation ausgeführt wurde, oder eine Tabelle, für die NOT LOGGED INITIALLY verarbeitet wird. Wenn dieser Fehler zurückgegeben wird, sollte die aktualisierende Wiederherstellung der Tabellenbereiche bereits gestoppt sein, d. h. nicht länger laufen.

Teil 2. Hohe Verfügbarkeit

Kapitel 5. 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").

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. Weitere Informationen zur Parallelverarbeitung finden Sie im Handbuch *Systemverwaltung*.
- Die Systeme müssen nach einem Hardware- oder Softwarefehler oder einem Störfall schnell wiederherstellbar sein. Sie sollten eventuell das Journaled File System verwenden. Ein *Journaled File System* ist ein Dateisystem, das automatisch Änderungen an seiner Struktur protokolliert. Dadurch sind Änderungen an der Dateisystemstruktur vor Systemabstürzen geschützt, es sei denn, das Speichermedium geht verloren oder wird beschädigt. Nach einem Systemabsturz werden die Änderungen in diesen Protokollen auf das Dateisystem in der Reihenfolge angewendet, in der sie ursprünglich protokolliert worden sind. Journaled File Systems können Sie schnell wiederherstellen. Vorzugsweise werden sie für Umgebungen mit großen Dateisystemen oder für Umgebungen, in denen die Datenintegrität besonders wichtig ist, verwendet.

Die schnelle Wiederherstellung hängt entscheidend vom Vorhandensein einer getesteten Sicherungs- und Wiederherstellungsstrategie ab. Weitere Informationen zu Wiederherstellungsstrategien finden Sie in „Kapitel 1. Entwickeln einer guten Sicherungs- und Wiederherstellungsstrategie" auf Seite 3.

- 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.

Zur Funktionsübernahme muss 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 Benutzer-Exit-Programms, 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 die neue ausgesetzte E/A-Unterstützung verwenden, um die neue Datenbank schnell zu initialisieren (siehe „Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank" auf Seite 206). Die sekundäre Datenbank auf der Bereitschaftsmaschine stellt die Protokolldateien ständig aktualisierend wieder her. 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 sekundären Datenbank auf der Bereitschaftsmaschine verbunden.

Die Unterstützung der Funktionsübernahme erhalten Sie auch über plattformspezifische Software, die Sie auf dem System installieren können. Zum Beispiel:

- High Availability Cluster Multi-Processing, Enhanced Scalability für AIX

  Weitere Informationen zu HACMP/ES finden Sie in „Kapitel 6. Hohe Verfügbarkeit unter AIX" auf Seite 211 oder im White Paper mit dem Titel „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 NT oder Windows 2000

  Weitere Informationen zu MSCS finden Sie in „Kapitel 7. Hohe Verfügbarkeit unter dem Windows-Betriebssystem" auf Seite 259.
- Sun Cluster oder VERITAS Cluster Server für die Solaris-Betriebsumgebung

  Weitere Informationen zu Sun Cluster 2.x finden Sie in „Kapitel 8. Hohe Verfügbarkeit in der Solaris-Betriebsumgebung" auf Seite 293; Informationen zu Sun Cluster 3.0 finden Sie im White Paper mit dem Titel „DB2 and High Availability on Sun Cluster 3.0", das auf der Website „DB2 UDB and DB2 Connect Online Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar ist. Weitere Informationen zu VERITAS Cluster Server finden Sie im White Paper mit dem Titel „DB2 and High Availability on VERITAS Cluster Server", das ebenfalls auf der Website „DB2 UDB and DB2 Connect Online Support" verfügbar ist.
- Multi-Computer/ServiceGuard für Hewlett-Packard

  Weitere Informationen zu HP MC/ServiceGuard finden Sie im White Paper mit dem Titel „IBM DB2 EE v.7.1 Implementation and Certification With Hewlett-Packard's MC/ServiceGuard High Availability Software", das auf der Website „DB2 UDB and DB2 Connect Online Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar ist.

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 zusammenarbeitet. Die einzelnen Prozessoren werden innerhalb des Clusters als Knoten bezeichnet. 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 Client-Anwendung 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.

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 Spiegeldatenbank eine vollständige Implementierung, d. h. das Teilen einer Spiegeldatenbank ohne Herunterfahren des Systems, bereitgestellt wird. 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. *Teilen einer Spiegeldatenbank* bedeutet, eine Sicherungskopie der Spiegeldatenbank zu erstellen.

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:

- Zur Erstellung eines Datenbankklons

  Einen Lesezugriffsklon der primären Datenbank können Sie z. B. verwenden, um Berichte zu erstellen.
- Als Bereitschaftsdatenbank
- Als Sicherungsimage

Diesen Befehl können Sie nur für die getrennte Spiegeldatenbank absetzen, und auf dieser Datenbank müssen Sie vor ihrem Einsatz zuerst das Programm **db2inidb** ausführen (siehe „db2inidb - Spiegeldatenbank initialisieren" auf Seite 360).

In einer Umgebung mit partitionierten Datenbanken müssen Sie den Befehl **db2inidb** auf jeder Partition ausführen, bevor Sie das getrennte Image einer beliebigen Partition verwenden können. Das Tool können Sie auf allen Partitionen gleichzeitig ausführen.

Erstellen eines Datenbankklons

Ein Klon einer Datenbank kann eine „Offlinesicherung" der primären Datenbank (Live-Datenbank) sein. Sie können jedoch die geklonte Datenbank nicht sichern, noch ihr Image auf dem Originalsystem wiederherstellen oder es mit Hilfe der Protokolldateien, die auf dem Originalsystem generiert wurden, aktualisierend wiederherstellen.

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 einen geeigneten Befehl auf Betriebssystemebene, um die Spiegeldatenbank 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. Stellen Sie von einer anderen Maschine aus die Verbindung zur gespiegelten Datenbank her.

5. Starten Sie das Datenbankexemplar:

   ```
   db2start
   ```

6. Initialisieren Sie die gespiegelte Datenbank als Klon der primären Datenbank:

   ```
   db2inidb aliasname-der-datenbank as snapshot
   ```

   Anmerkung: Mit diesem Befehl werden Transaktionen rückgängig gemacht, die während des Teilens gerade verarbeitet werden.

Verwenden der geteilten Spiegeldatenbank als Bereitschaftsdatenbank

Während die gespiegelte Datenbank (Bereitschaftsdatenbank) ständig anhand der Protokolle aktualisierend wiederhergestellt wird, werden neue Protokolle, die gerade von der primären Datenbank erstellt werden, ständig aus dem primären System abgerufen. Gehen Sie wie folgt vor, um die 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 einen geeigneten Befehl auf Betriebssystemebene, um die Spiegeldatenbank 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. Stellen Sie die Verbindung zwischen der gespiegelten Datenbank und einem anderen Exemplar her.
5. Versetzen Sie die gespiegelte Datenbank in den Status *Aktualisierende Wiederherstellung anstehend*:

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

   Wenn Sie über DMS-Tabellenbereiche (vom Datenbankmanager verwaltete Tabellenbereiche) verfügen, können Sie eine vollständige Datenbanksicherung erstellen, damit der Systemaufwand für eine Sicherung auf der Produktionsdatenbank entfällt.
6. Definieren Sie ein Benutzer-Exit-Programm, um die neuesten Protokolldateien vom primären System abzurufen.
7. Stellen Sie die Datenbank bis zum Ende der Protokolle aktualisierend wieder her.
8. Rufen Sie Protokolldateien ab, und stellen Sie die Datenbank bis zum Ende der Protokolle aktualisierend wieder her, bis die primäre Datenbank ausfällt.

Verwenden der geteilten Spiegeldatenbank als Sicherungsimage

Gehen Sie wie folgt vor, um die 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 einen geeigneten Befehl auf Betriebssystemebene, um die Spiegeldatenbank 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. Verwenden Sie Befehle auf Betriebssystemebene, um die gespiegelten Daten und Protokolle auf das primäre System zu kopieren.
5. Starten Sie das Datenbankexemplar:
   ```
   db2start
   ```
6. Initialisieren Sie die gespiegelte Datenbank als „Sicherungsimage", das Sie zum Zurückkopieren der Daten von abgeteilten Datenträgern auf Datenträger des Originalsystems verwenden können. (Kopieren Sie nicht das Dateisystem zurück, das die Protokolldateien enthält, da die Protokolle während der aktualisierenden Wiederherstellung benötigt werden.)
   ```
   db2inidb aliasname-der-datenbank as mirror
   ```
7. Stellen Sie die Datenbank (auf dem Originalsystem) bis zum Ende der Protokolle aktualisierend wieder her.

Kapitel 6. Hohe Verfügbarkeit unter AIX

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 (siehe *HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide*). 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 Paging-Bereich 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 Script-Dateien, die auszuführen sind, wenn Ereignisse eintreten, sind Teil dieser Definition.

  Weitere Informationen zu benutzerdefinierten Ereignissen und der rules-Datei finden Sie in „HACMP ES-Ereignisüberwachung und benutzerdefinierte Ereignisse" auf Seite 232.
- HACMP-Client-Dienstprogramme 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 *Heartbeats* 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 Arten von Ereignissen: Standardereignisse, die innerhalb der Funktionen von HACMP ES abgefangen werden, und benutzerdefinierte Ereignisse, die mit 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).

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.

Im Allgemeinen arbeitet DB2 Universal Database Enterprise - Extended Edition (UDB EEE) im Modus der gegenseitigen Übernahme mit Partitionen auf jedem Knoten. Eine Ausnahme bildet eine Konstellation, in der der Katalogknoten Teil einer Bereitschaftskonfiguration ist.

Bei der Planung einer umfangreichen DB2-Installation auf einem RISC System/6000 SP mit HACMP ES müssen Sie besonderes Augenmerk auf die Verteilung der Knoten des Clusters innerhalb der oder zwischen den RISC System/6000 SP-Rahmen (Frames) legen. 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.

Eine weitere Überlegung zur Planung 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 EEE 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). Dies dient als Ausgangspunkt für das Ausfüllen der HACMP-Arbeitsblätter, die der Dokumentation beigefügt sind.

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 zugreifen können. Für DB2 UDB EEE 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 11. 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
```

Abb. 18 auf Seite 216 und Abb. 19 auf Seite 217 zeigen ein Beispiel für die Konfiguration eines DB2-SSA-E/A-Subsystems sowie einige Gesichtspunkte der Planung, die zur Sicherstellung einer hochverfügbaren Konfiguration externer Platten und der Möglichkeit des konfliktfreien Zugriffs auf alle Datenträgergruppen gehört.

DB2-SSA-E/A-Subsystemkonfiguration - Kein einzelner Fehlerpunkt

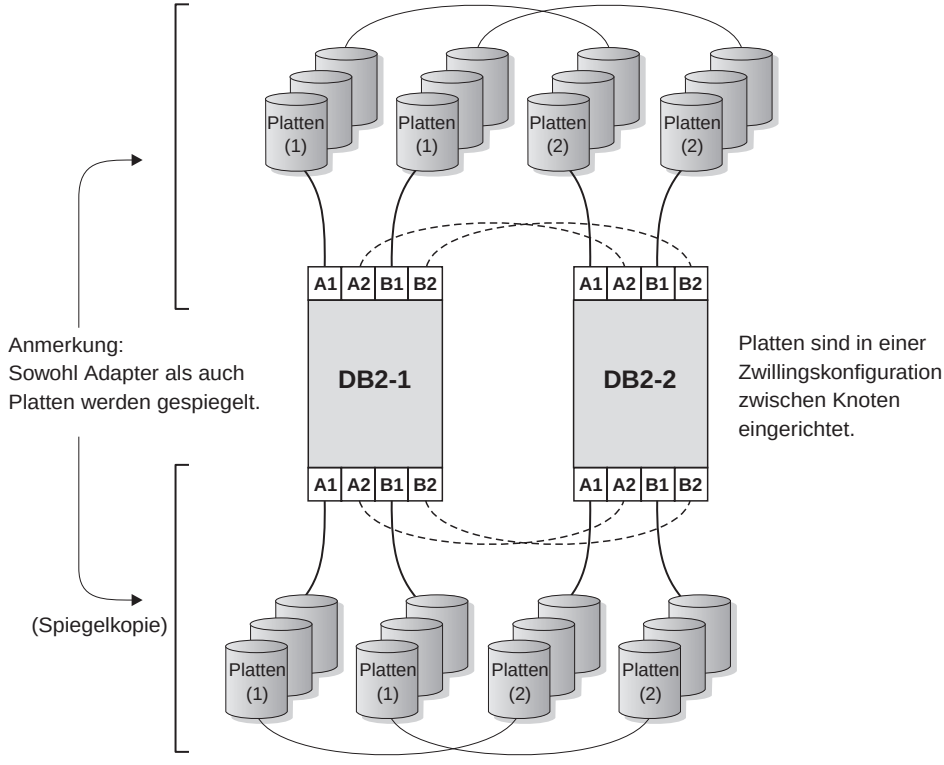


Abbildung 18. Kein einzelner Fehlerpunkt

DB2-SSA-E/A-Subsystemkonfiguration -
Konfiguration von Datenträgergruppe (vg) und logischer Datenträger (lv)

DB2-Datenbank testdata für Dateisystem /database exemplarname powertp

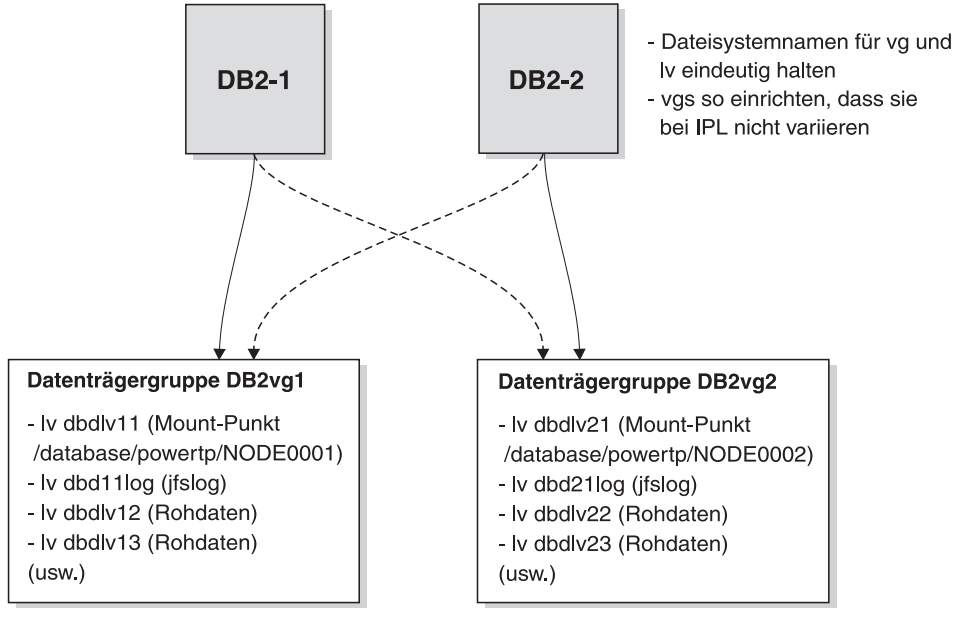


Abbildung 19. Konfiguration von Datenträgergruppe und logischem Datenträger

Konfigurieren einer DB2-Datenbankpartition

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 Script-Datei `rc.db2pe` gehört zum Lieferumfang von DB2 UDB EEE (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 müssen konfiguriert werden, um einen ord-

nungsgemäßen Betrieb sicherzustellen, wenn zwei Datenbankpartitionen auf demselben physischen Knoten ausgeführt werden.)

Wenn Sie einen Anwendungsserver in einer HACMP-Konfiguration einer DB2-Datenbankpartition erstellen, geben Sie auf folgende Weise `rc.db2pe` als Start- und Stopp-Script an:

```
/usr/bin/rc.db2pe <exemplar> <dpn> <sekundäre dpn> start <switch verwenden>
/usr/bin/rc.db2pe <exemplar> <dpn> <sekundäre dpn> stop <switch verwenden>

Dabei gilt Folgendes:

<exemplar> ist der Exemplarname.
<dpn> ist die Datenbankpartitionsnummer.
<sekundäre dpn> ist die sekundäre Datenbankpartitionsnummer (nur in
   Konfigurationen mit gegenseitiger Übernahme). In Bereitschafts-
   konfigurationen gleich <dpn>.
<switch verwenden> ist in der Regel leer. Ist dies der Fall, zeigt dieser
   Parameter an, dass das SP-Switch-Netzwerk für das Feld hostname
   in der Datei db2nodes.cfg verwendet wird (der gesamte Verkehr für DB2
   wird über den SP-Switch geleitet).
   Falls nicht leer, ist der verwendete Name der Host-Name des zu
   verwendenden SP-Knotens.
```

Der DB2-Befehl LIST DATABASE DIRECTORY wird innerhalb von `rc.db2pe` zum Auffinden aller Datenbanken verwendet, die für diese Datenbankpartition konfiguriert sind. Die Script-Datei sucht anschließend die Dateien `/usr/bin/reg.parms.`*`DATENBANK`* und `/usr/bin/failover.parms.`*`DATENBANK`*, wobei DATENBANK jeweils die Datenbanken darstellt, die für diese Datenbankpartition konfiguriert sind. In einer Konfiguration zur gegenseitigen Übernahme ist es empfehlenswert, diese Parameterdateien `reg.parms.`*`xxx`* und `failover.parms.`*`xxx`* zu erstellen. In der Datei `failover.parms.`*`xxx`* sollten die Einstellungen für BUFFPAGE, DBHEAP und ggf. andere Parameter, die die Pufferpools betreffen, angepasst werden, um der Möglichkeit, dass mehr als ein Pufferpool vorhanden sein kann, Rechnung zu tragen. Die Beispieldateien `reg.parms.SAMPLE` und `failover.parms.SAMPLE` sind für Sie vorbereitet.

Einer der wichtigen Parameter in dieser Umgebung ist der Konfigurationsparameter *start_stop_time* des Datenbankmanagers, dessen Standardwert 10 Minuten ist. `rc.db2pe` setzt diesen Parameter jedoch auf 2 Minuten. Sie sollten diesen Parameter durch `rc.db2pe` auf einen Wert von 10 Minuten oder etwas mehr setzen. In diesem Kontext ist die angegebene Zeitdauer das Intervall zwischen dem Ausfall der Partition und der Wiederherstellung der Partition. Wenn in den Anwendungen, die auf einer Partition ausgeführt werden, häufig COMMIT-Anforderungen abgesetzt werden, sollten zehn Minuten nach dem Ausfall einer Datenbankpartition ausreichen, um nicht festgeschriebene Transaktionen rückgängig zu machen (ROLLBACK) und einen Konsistenzzustand in der Datenbank in dieser Partition zu erreichen. Wenn Sie eine hohe Auslastung oder viele Partitionen haben, müssen Sie die Zeitdauer eventuell verlän-

gern, um die Wahrscheinlichkeit zu verringern, dass Zeitlimitüberschreitungen auftreten, bevor die ROLLBACK-Operation beendet ist.

Die folgenden Beispiele beschreiben eine Konfiguration im Bereitschaftsmodus und eine Konfiguration zur gegenseitigen Übernahme. In beiden Beispielen enthalten die Ressourcengruppen eine Service-IP-Switch-Aliasadresse. Diese Adresse dient zu folgenden Zwecken:

- NFS-Zugriff auf einen Dateiserver für die Dateisysteme des DB2-Exemplareigners
- Sonstiger Client-Zugriff, der für den Fall einer Funktionsübernahme, einer TSM-Verbindung (Tivoli Storage Manager, früher ADSM) oder einer ähnlichen Operation aufrechterhalten werden muss

Wenn in Ihrer Implementierung diese Aliasnamen nicht erforderlich sind, können sie entfernt werden. Wenn sie entfernt werden, stellen Sie sicher, dass in der Script-Datei `rc.db2pe` der Parameter *MOUNT_NFS* auf den Wert `NO` gesetzt wird.

Beispiel für eine Konfiguration im Bereitschaftsmodus

Dieses Beispiel geht davon aus, dass eine Konfiguration im Bereitschaftsmodus zwischen den physischen Knoten 1 und 2 besteht und dass der Name des DB2-Exemplars POWERTP lautet. Die Datenbankpartition ist 1, die Datenbank heißt TESTDATA und befindet sich im Dateisystem `/database`.

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1     (<<< dies ist die Switch-Aliasadresse)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 1 stop
```

Beispiel für eine Konfiguration für gegenseitige Übernahme

Dieses Beispiel geht davon aus, dass eine Konfiguration für gegenseitige Übernahme zwischen den physischen Knoten 1 und 2 besteht und dass der Name des DB2-Exemplars POWERTP lautet. Die Datenbankpartitionen sind 1 und 2, die Datenbank heißt TESTDATA und befindet sich im Dateisystem `/database`.

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1     (<<< dies ist die Switch-Aliasadresse)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 2 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 2 stop

Resource group name: db2_pd_2
```

```
Node Relationship: cascading
Participating nodenames: node2_eth, node1_eth
Service_IP_label: nfs_switch_2     (<<< dies ist die Switch-Aliasadresse)
Filesystems: /database/powertp/NODE0002
Volume Groups: DB2vg2
Application Servers: db2_dp2_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 2 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 2 1 stop
```

Konfiguration eines NFS-Serverknotens

Das Script `rc.db2pe` kann außerdem dazu verwendet werden, über NFS angehängte Verzeichnisse paralleler DB2-Exemplarbenutzerverzeichnisse verfügbar zu machen. Dies kann erreicht werden, indem der Parameter *MOUNT_NFS* in der Script-Datei `rc.db2pe` auf den Wert `YES` gesetzt wird und das NFS-Serverpaar für die Übernahme wie folgt konfiguriert wird:

- Konfigurieren Sie das Ausgangsverzeichnis, und exportieren Sie es als "root" mit Hilfe von `/etc/exports` und des Befehls **exportfs** an die IP-Adresse, die auf den Knoten im selben Teilnetz wie die IP-Adresse des NFS-Servers verwendet wird. Geben Sie sowohl die HACMP-Boot-Adresse als auch die Serviceadresse an. Die IP-Adresse des NFS-Servers ist die gleiche Adresse wie die Serviceadresse in HACMP, die von einem Ausweichknoten übernommen werden kann. Das Ausgangsverzeichnis des DB2-Exemplareigners sollte direkt über NFS angehängt und nicht automatisch angehängt (Automount) werden. (Die Verwendung der Automount-Einrichtung wird von den Scripts nicht unterstützt, wenn es sich um das Ausgangsverzeichnis des DB2-Exemplareigners handelt.)
- Erstellen Sie über SMIT oder eine Basiskonfiguration einen separaten Eintrag in `/etc/filesystems` für dieses Dateisystem, so dass alle Knoten in der parallelen DB2-Gruppierung, einschließlich des Dateiservers, mit Hilfe des NFS-Dateisystembefehls anhängen können.

  Zum Beispiel kann ein JFS-Dateisystem `/nfshome` an alle Knoten als `/dbhome` exportiert werden. Jeder Knoten erstellt ein NFS-Dateisystem `/dbname`, das dem Dateisystem `nfs_server:/nfshome` entspricht. Daher wäre das Ausgangsverzeichnis des DB2-Exemplareigners `/dbhome/powertp`, wenn der Exemplarname "powertp" wäre.

  Stellen Sie sicher, dass in `/etc/filesystems` die NFS-Parameter "hard", "bg", "intr" und "rw" für das Anhängen verwendet werden.
- Stellen Sie sicher, dass die Definitionen des DB2-Exemplareigners, die dem Ausgangsverzeichnis `/dbhome/powertp` in `/etc/passwd` zugeordnet sind, auf allen Knoten identisch sind.

  Die Benutzerdefinitionen in einer SP-Umgebung werden typischerweise auf der Steuer-Workstation erstellt, und "supper" oder "pcp" werden zur Verteilung von `/etc/passwd`, `/etc/security/passwd`, `/etc/security/user` und `/etc/security/group` an alle Knoten verwendet.
- Konfigurieren Sie *nicht* die zu exportierenden NFS-Dateisysteme ("nfs_filesystems to export") in HACMP-Ressourcengruppen für die Datenträger-

gruppe und das Dateisystem, das exportiert wird. Konfigurieren Sie diese Merkmale normal für NFS. Die Scripts für den NFS-Server steuern das Exportieren der Dateisysteme.

- Stellen Sie sicher, dass die Hauptgerätenummer der Datenträgergruppe, auf der sich das Dateisystem befindet, auf dem Primärknoten und dem Übernahmeknoten gleich sind. Dies wird durch die Verwendung von **importvg** mit der Option `-V` erreicht.
- Überprüfen Sie, ob der Parameter *MOUNT_NFS* in der Script-Datei `rc.db2pe` auf den Wert `YES` gesetzt ist, und ob jeder Knoten das anzuhängende NFS-Dateisystem in `/etc/filesystems` hat. Andernfalls ist `rc.db2pe` nicht in der Lage, das Dateisystem anzuhängen und DB2 zu starten.
- Wenn der DB2-Exemplareigner bereits erstellt wurde und Sie die Benutzerverzeichnisstruktur in das Dateisystem, das Sie erstellen, kopieren, stellen Sie sicher, dass Sie die Verzeichnisstruktur mit dem Befehl **tar** (`-cvf`) komprimieren. Dadurch bleiben symbolische Verbindungen erhalten.
- Vergessen Sie nicht, sowohl die Adapter als auch die Platten für die logischen Datenträger sowie die Dateisystemprotokolle des Dateisystems, das Sie erstellen, zu spiegeln.

Beispiel für eine NFS-Serverübernahmekonfiguration

Dieses Beispiel geht davon aus, dass es ein NFS-Serverdateisystem `/nfshome` in der Datenträgergruppe `nfsvg` über die IP-Adresse "nfs_server" gibt. Der DB2-Exemplarname ist POWERTP und das Ausgangsverzeichnis `/dbhome/powertp`.

```
Resource group name: nfs_server
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_server     (<<< dies ist die Switch-Aliasadresse)
Filesystems: /nfshome
Volume Groups: nfsvg
Application Servers: nfs_server_app
Application Server Start Script: /usr/bin/rc.db2pe powertp NFS SERVER start
Application Server Stop Script: /usr/bin/rc.db2pe powertp NFS SERVER stop
```

In diesem Beispiel gilt:

- `/etc/filesystems` auf allen Knoten würde einen Eintrag für `/dbhome` als anzuhängendes `nfs_server:/nfshome` enthalten. `nfs_server` ist eine Service-IP-Switch-Aliasadresse.
- `/etc/exports` auf dem Knoten `nfs_server` und dem Ausweichknoten würde die Boot-Adresse und Serviceadresse einschließen und einen Eintrag für `/nfsfs -root=nfs_switch_1, nfs_switch_2, ...` enthalten.

Überlegungen zur Konfiguration des SP-Switch

Bei der Implementierung von HACMP ES mit dem SP-Switch sind folgende Punkte zu beachten:

- Auf dem SP-Switch gibt es ″Basisadressen″ und ″Aliasadressen″. Die Basisadressen sind diejenigen, die im SP System Data Repository (SDR) definiert sind und durch `rc.switch` konfiguriert werden, wenn das System ″gebootet″ wird. Die Aliasadressen sind IP-Adressen, die neben der Basisadresse in die `css0`-Schnittstelle mit Hilfe des Befehls **ifconfig** mit einem Aliasattribut konfiguriert werden. Beispiel:

  ```
  ifconfig css0 inet alias sw_alias_1 up
  ```

- Bei der Konfiguration der DB2-Datei `db2nodes.cfg` sollten sowohl für das Feld ″hostname″ als auch für das Feld ″netname″ Namen der SP-Switch-Basis-IP-Adressen verwendet werden. Die Switch-IP-Aliasadressen dienen *nur* der Erhaltung der NFS-Konnektivität. Die DB2-Funktionsübernahme wird durch den Neustart von DB2 mit Hilfe des Befehls **db2start** (RESTART) erreicht (der `db2nodes.cfg` aktualisiert).
- Verwechseln Sie die Switch-Adressen nicht mit den Aliasnamen in `etc/hosts`. Die SP-Switch-Adressen und die SP-Switch-IP-Aliasadressen sind entweder in `etc/hosts` oder in DNS reale Adressen. Die Aliasadressen für einen Switch sind nicht andere Namen für die SP-Switch-Basisadresse. Jeder hat eine eigene separate Adresse.
- Die SP-Switch-Basisadressen sind immer auf einem Knoten vorhanden, wenn er aktiv ist. HACMP ES konfiguriert und versetzt diese Adressen nicht zwischen Knoten.
- Wenn Sie beabsichtigen, SP-Switch-Aliasadressen zu verwenden, konfigurieren Sie diese für HACMP als Boot- und Serviceadressen für Überwachungssignale (Heartbeating) und IP-Adressübernahme. Wenn Sie nicht beabsichtigen, SP-Switch-Aliasadressen zu verwenden, konfigurieren Sie die SP-Switch-Basisadresse für HACMP als Serviceadresse *nur* für Überwachungssignale (keine IP-Adressübernahme). Konfigurieren Sie in keiner Konfiguration Aliasadressen *und* die Switch-Basisadresse. Eine solche Konfiguration wird von HACMP ES nicht unterstützt.
- Nur die SP-Switch-Aliasadressen (und nicht die SP-Switch-Basisadressen) werden zwischen Knoten bei einer IP-Übernahmekonfiguration versetzt.
- Der Bedarf an Aliasnamen für SP-Switch ergibt sich aus der Tatsache, dass es nur einen SP-Switch-Adapter pro Knoten geben kann. Die Verwendung von Aliasadressen ermöglicht es, dass ein Knoten die Switch-IP-Aliasadresse eines anderen Knotens übernehmen kann, ohne einen anderen Switch-Adapter hinzufügen zu müssen. Dies ist auf Knoten nützlich, auf denen Adapterplätze nur begrenzt vorhanden sind. Weitere Informationen zur Durchführung von Wiederherstellungen nach Ausfällen von SP-Switch-Adaptern finden Sie im Abschnitt zu Netzwerkausfällen in „HACMP ES-Script-Dateien” auf Seite 236.
- Wenn Sie den SP-Switch zur Übernahme der IP-Adresse konfigurieren, müssen Sie zwei zusätzliche IP-Aliasadressen pro Knoten erstellen: eine als Boot-Adresse und eine als Serviceadresse.

- Vergessen Sie nicht, ″HPS″ in der Definition des HACMP ES-Netzwerknamens für eine IP-Basisadresse bzw. eine IP-Aliasadresse für einen SP-Switch zu verwenden.
- `rc.cluster` in HACMP führt beim Starten von HACMP automatisch **ifconfig** für die SP-Switch-Boot-Adresse aus. Außer der Erstellung von IP-Adresse und Name sowie deren Definition für HACMP sind keine weiteren Konfigurationsschritte erforderlich.
- Der Knoten Eprimary des SP-Switch ist der Server, der die Befehle Estart, Efence und Eunfence implementiert. Die HACMP-Scripts versuchen, die Befehle Eunfence oder Estart für einen Knoten auszuführen, wenn HACMP gestartet wird, und den Switch verfügbar zu machen, wenn er als einer seiner Netzwerke definiert sein sollte. Aus diesem Grund ist sicherzustellen, dass der Eprimary-Knoten beim Starten von HACMP verfügbar ist. Der HACMP-Code wartet bis zu 12 Minuten auf die Durchführung einer Eprimary-Funktionsübernahme, bevor er mit einem Fehler die Verarbeitung beendet.
- Der Eprimary-Knoten des SP-Switch wird von SP PSSP (Parallel System Support Program) zwischen Knoten versetzt, nicht von HACMP. Wenn ein Eprimary-Knoten offline geschaltet wird, übergibt PSSP die Aufgaben des Eprimary-Knotens automatisch an einen Ausweichknoten. Das Switch-Netzwerk ist von dieser Änderung nicht betroffen und bleibt aktiv.

Beispiele für DB2-HACMP-Konfigurationen

Die folgenden Beispiele zeigen verschiedene Konfigurationen zur Unterstützung von Funktionsübernahmen und erläutern, was im Fall eines Ausfalls geschieht.

Im Fall von DB2-HACMP-Konfigurationen für gegenseitige Übernahme (Abb. 20 auf Seite 224, Abb. 21 auf Seite 225 und Abb. 22 auf Seite 226) gilt:

- HACMP-Adapter werden für Ethernet- sowie für SP-Switch-Boot- und SP-Switch-Aliasadressen definiert, Basisadressen bleiben unberührt. Beachten Sie, dass eine Zeichenfolge ″HPS″ im HACMP-Netzwerknamen zu verwenden ist.
- Das Verzeichnis `NFS_server/nfshome` wird als `/dbhome` auf allen Knoten über Switch-Aliasadressen angehängt.
- Die Datei `db2nodes.cfg` enthält die SP-Switch-Basisadressen. Die Datei `db2nodes.cfg` wird durch den Befehl **db2start** (RESTART) nach einer Funktionsübernahme für eine DB2-Datenbankpartition (logischer Knoten) geändert.
- Die Boot-Aliasadressen für die SP-Switches werden nicht gezeigt.
- Knoten können sich in verschiedenen SP-Rahmen befinden.

DB2-HACMP-Konfiguration: Gegenseitige Übernahme mit NFS-Funktionsübernahme

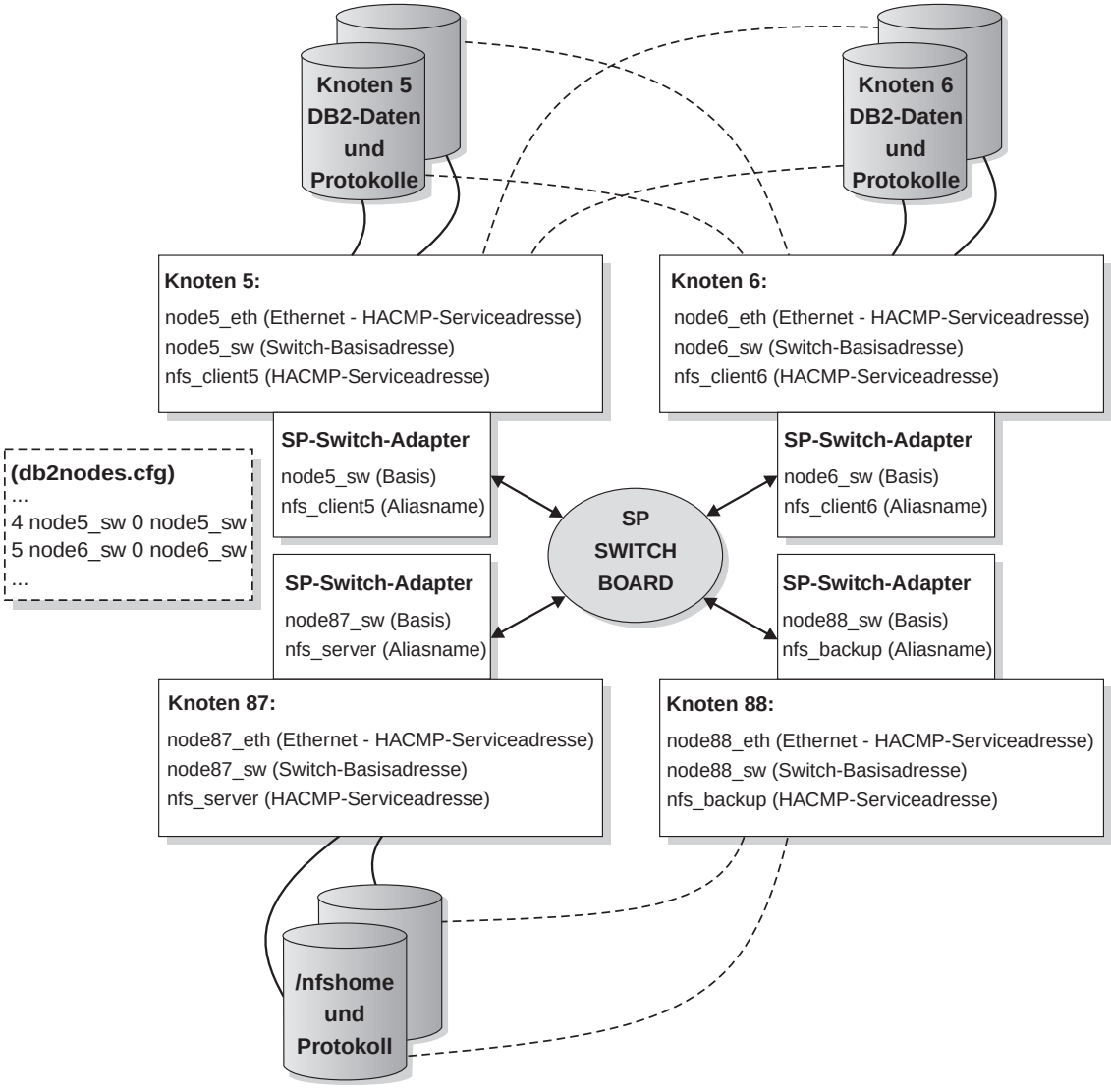


Abbildung 20. Gegenseitige Übernahme mit NFS-Funktionsübernahme - Normal

DB2-HACMP-Konfiguration:

Gegenseitige Übernahme mit NFS-Funktionsübernahme - NFS-Funktionsübernahme

- nfs_server SP-Switch-Alias-IP-Adr. u. nfs angehängt. /nfshome von Knoten 87 auf 88 versetzt.
- SP-Switch-ARP-Code hat Funktionen zum Aktualisieren aller Switch-ARP-Caches bei dieser Übernahme.

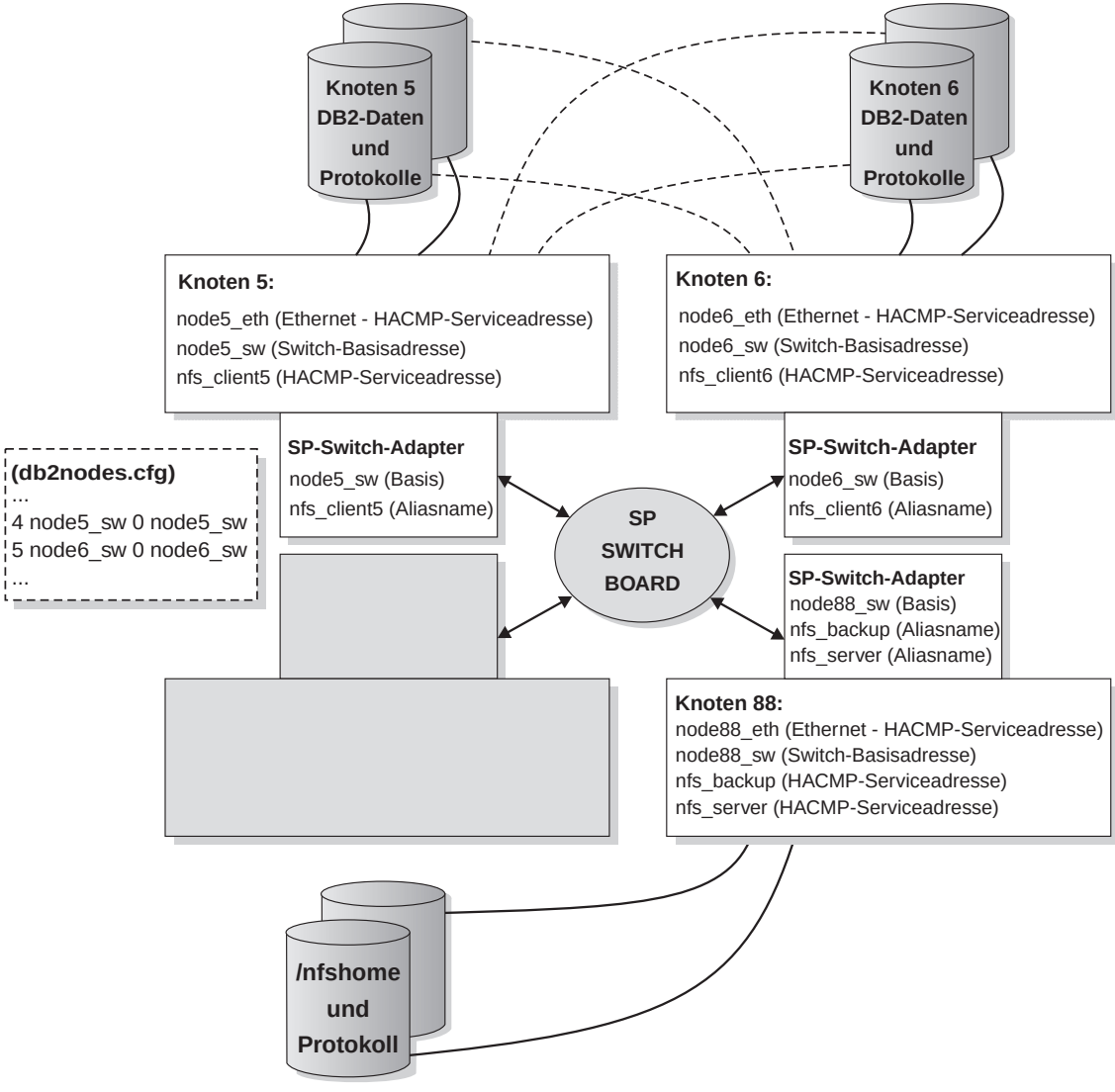


Abbildung 21. Gegenseitige Übernahme mit NFS-Funktionsübernahme - NFS-Funktionsübernahme

DB2-HACMP-Konfiguration:
Gegenseitige Übernahme mit NFS-Funktionsübernahme - DB2-Funktionsübernahme

- Übernahme der Switch-IP-Adresse ermöglicht anderen Servern (wie ADSM) das Beibehalten der Konnektivität.
- Knoten 5 führt zwei logische Knoten von DB2 aus.

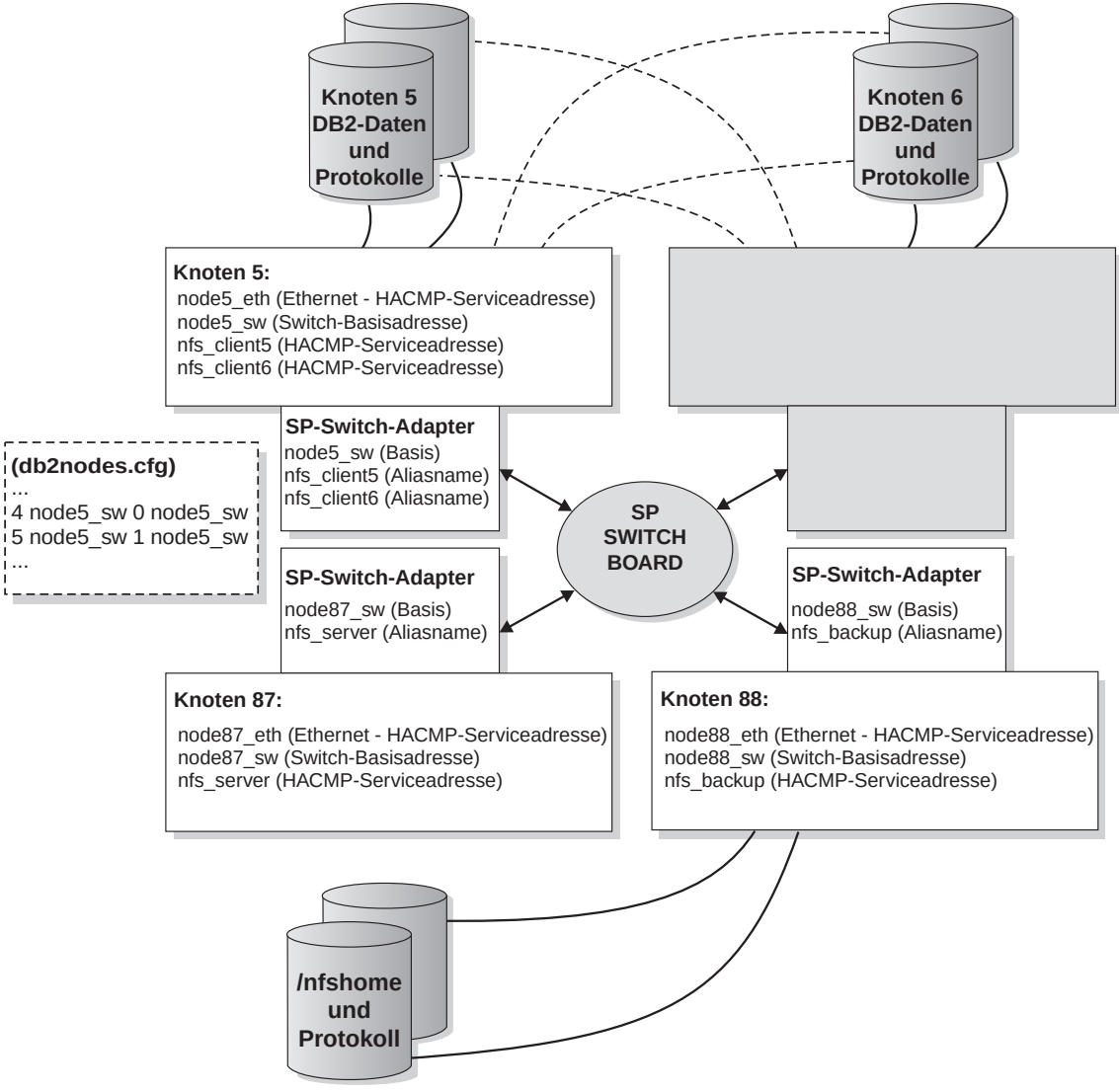


Abbildung 22. Gegenseitige Übernahme mit NFS-Funktionsübernahme - DB2-Funktionsübernahme

Im Fall von DB2-HACMP-Konfigurationen im Bereitschaftsmodus (Abb. 23 auf Seite 228 und Abb. 24 auf Seite 229) gilt:

- HACMP-Adapter werden für Ethernet- sowie für SP-Switch-Boot- und SP-Switch-Aliasadressen definiert, Basisadressen bleiben unberührt. Beachten Sie, dass eine Zeichenfolge ″HPS″ im HACMP-Netzwerknamen zu verwenden ist.
- Das Verzeichnis `NFS_server/nfshome` wird als `/dbhome` auf allen Knoten über Switch-Aliasadressen angehängt.
- Die Datei `db2nodes.cfg` enthält die SP-Switch-Basisadressen. Die Datei `db2nodes.cfg` wird durch den Befehl **db2start** (RESTART) nach einer Funktionsübernahme für eine DB2-Datenbankpartition (logischer Knoten) geändert.
- Die Boot-Aliasadressen für die SP-Switches werden nicht gezeigt.

DB2-HACMP-Konfiguration:
Bereitschaftsmodus mit NFS-Funktionsübernahme - Normal

Anmerkung: Ein Knoten im Bereitschaftsmodus kann je nach Plattenverkabelung mehrere Knoten sichern.

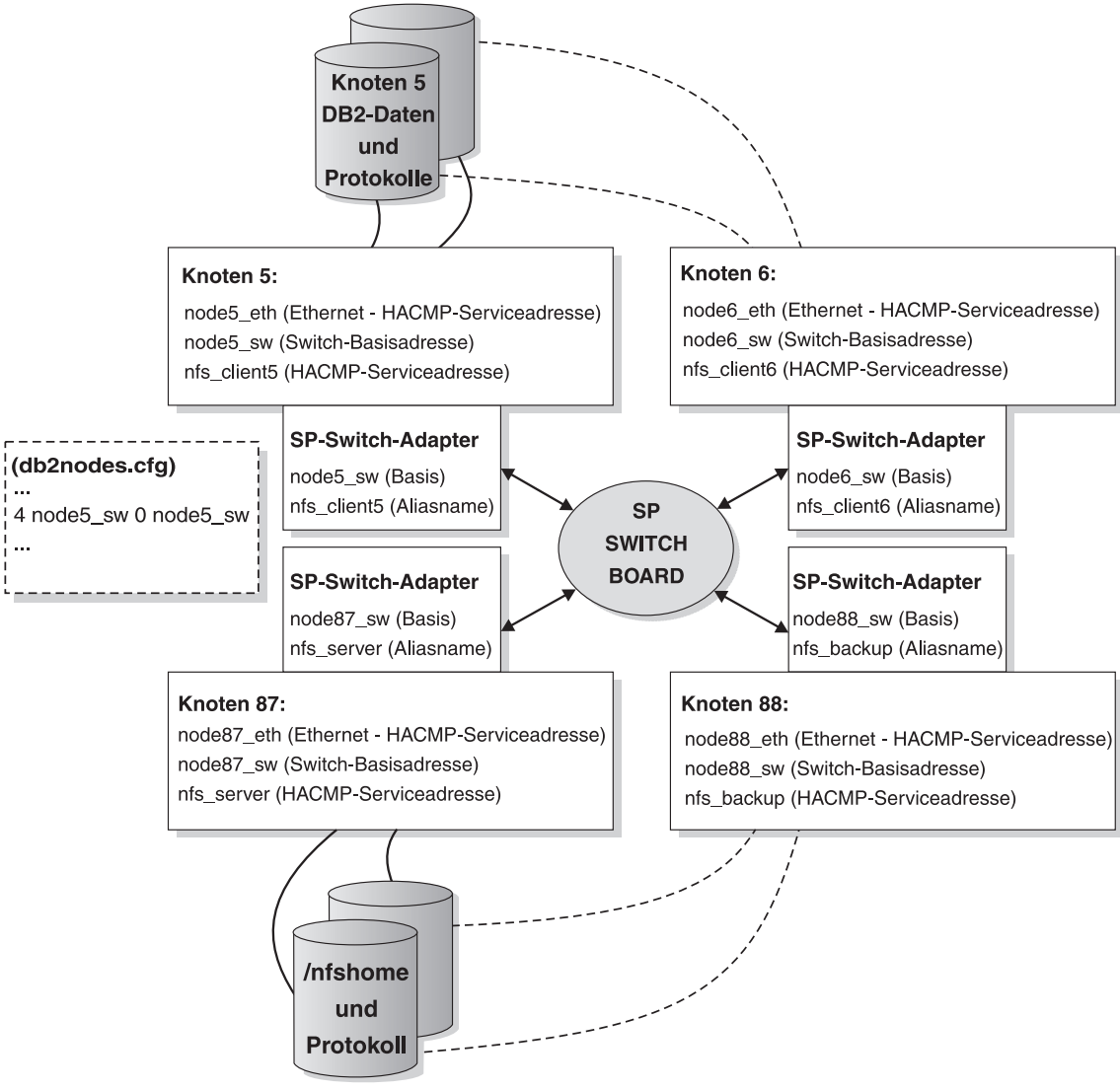


Abbildung 23. Bereitschaft mit NFS-Funktionsübernahme - Normal

DB2-HACMP-Konfiguration:
Bereitschaftsmodus mit NFS-Funktionsübernahme - DB2-Funktionsübernahme

Anmerkung: Ein Knoten im Bereitschaftsmodus kann je nach Plattenverkabelung mehrere Knoten sichern.

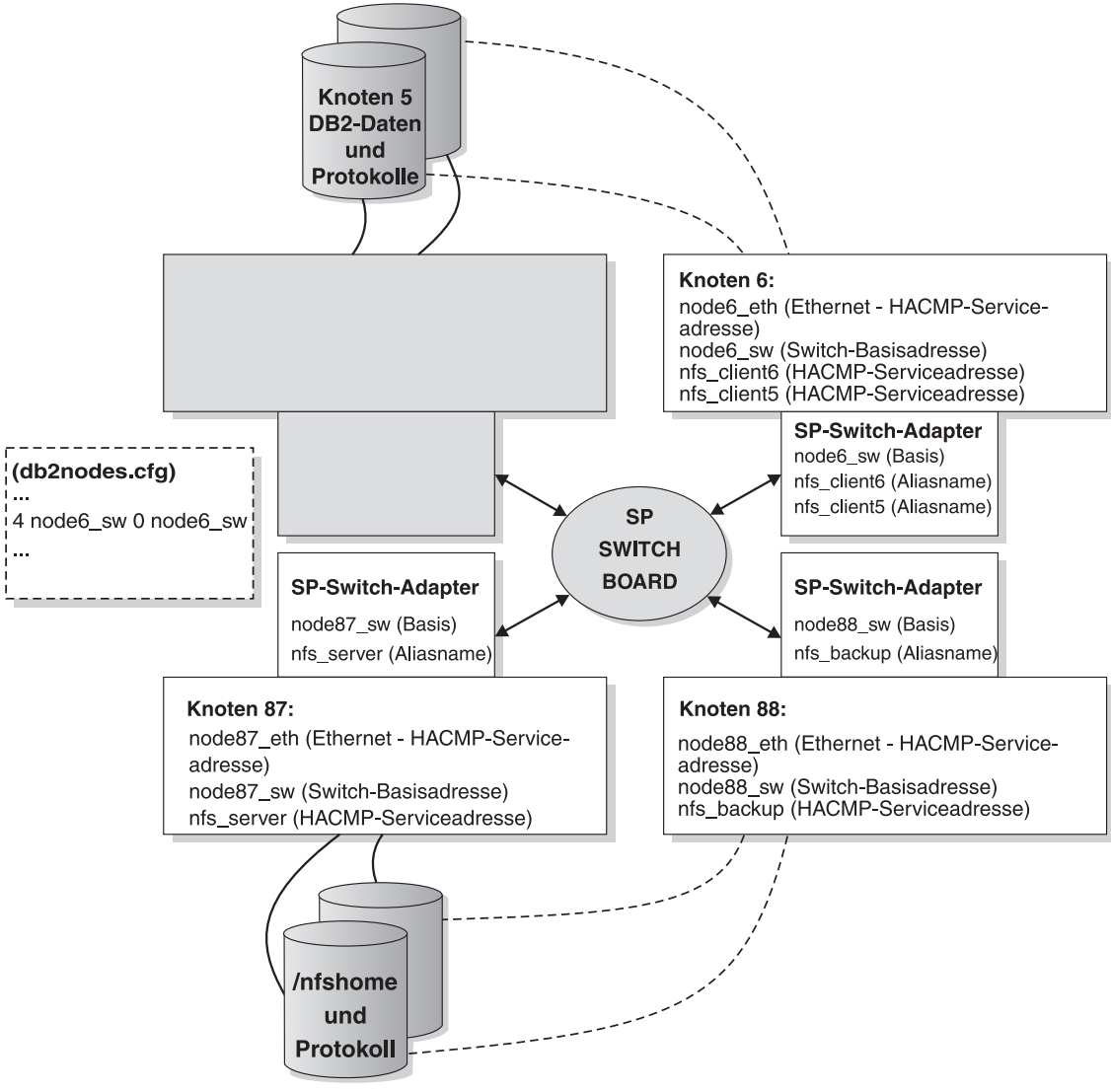


Abbildung 24. Bereitschaft mit NFS-Funktionsübernahme - DB2-Funktionsübernahme

Clusterkonfiguration

Im Fall von DB2-HACMP-Konfigurationen für gegenseitige Übernahme ohne NFS-Übernahme (Abb. 25 und Abb. 26 auf Seite 231) gilt:

- HACMP-Adapter werden für Ethernet und SP-Switch-Basisadressen definiert. Beachten Sie, dass es, wenn Basisadressen als Serviceadressen für HACMP konfiguriert werden, keine Boot-Adresse (nur ein Überwachungssignal) gibt. Vergessen Sie nicht, dass eine Zeichenfolge ″HPS″ im HACMP-Netzwerknamen für den SP-Switch zu verwenden ist.
- Die Datei `db2nodes.cfg` enthält die SP-Switch-Basisadressen. Die Datei `db2nodes.cfg` wird durch den Befehl **db2start** (RESTART) nach einer Funktionsübernahme für eine DB2-Datenbankpartition (logischer Knoten) geändert.
- Es werden keine NFS-Übernahmefunktionen gezeigt.
- Knoten können sich in verschiedenen SP-Rahmen befinden.

DB2-HACMP-Konfiguration: Gegenseitige Übernahme ohne NFS-Funktionsübernahme - Normal

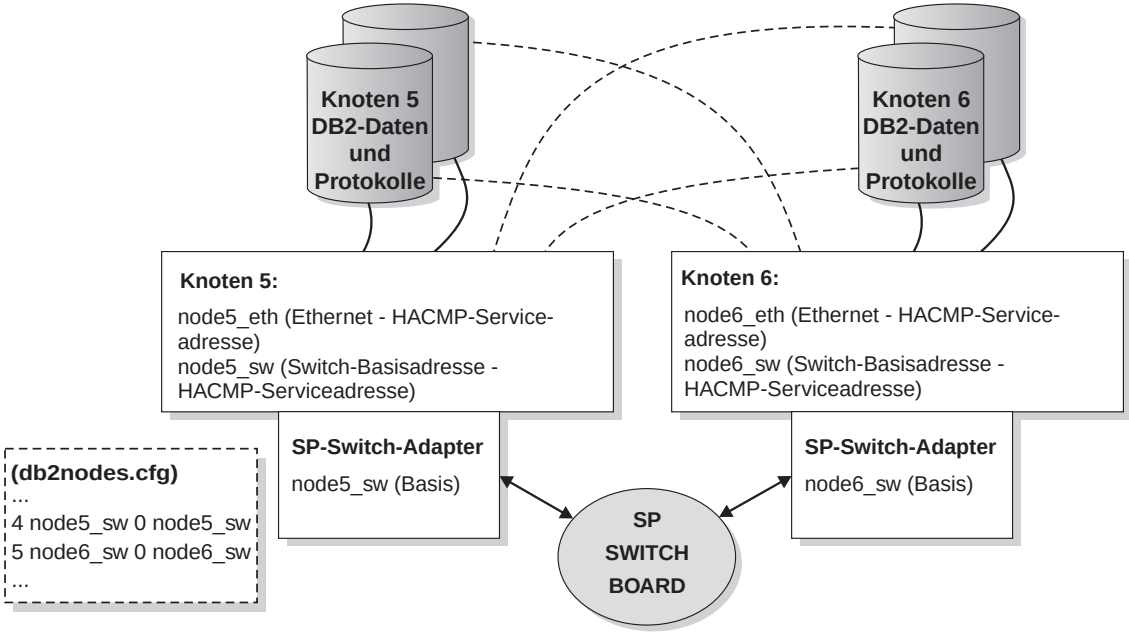


Abbildung 25. Gegenseitige Übernahme ohne NFS-Funktionsübernahme - Normal

DB2-HACMP-Konfiguration:
Gegenseitige Übernahme ohne NFS-Funktionsübernahme - DB2-Funktionsübernahme
- Knoten 5 führt zwei logische Knoten von DB2 aus.

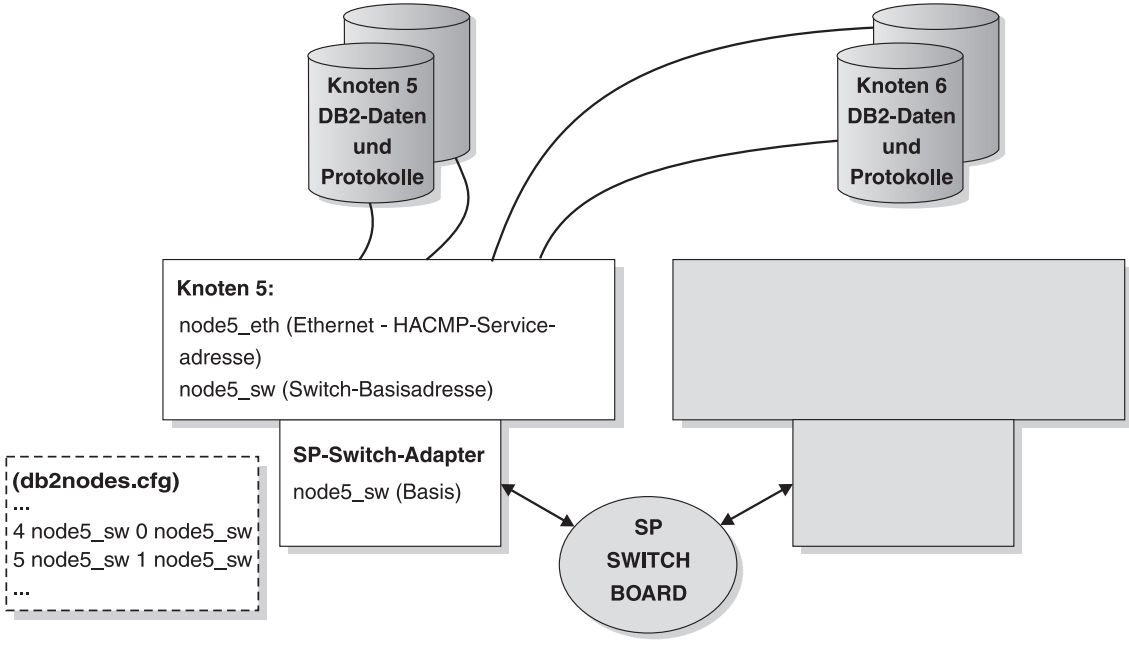


Abbildung 26. Gegenseitige Übernahme ohne NFS-Funktionsübernahme - DB2-Funktionsübernahme

Empfehlungen für den DB2-HACMP-Start

Es wird empfohlen, HACMP nicht für den Start bei Systemstart in `/etc/inittab` anzugeben. HACMP sollte nach dem Booten der Knoten manuell gestartet werden. Dadurch wird eine unterbrechungsfreie Wartung des ausgefallenen Knotens ermöglicht.

Betrachten Sie als Beispiel für eine Wartung mit Unterbrechung den Fall, in dem auf einem Knoten ein Hardwarefehler mit Systemabsturz auftritt. Die Funktionsübernahme wird von HACMP automatisch eingeleitet und die Wiederherstellung erfolgreich durchgeführt. Allerdings muss der ausgefallene Knoten repariert werden. Wäre HACMP in `/etc/inittab` zum Starten bei erneutem Booten konfiguriert, würde dieser Knoten nach Abschluss des Bootens versuchen, sich erneut einzugliedern, was in diesem Fall jedoch nicht wünschenswert ist.

Zur unterbrechungsfreien Wartung sollte ein manuelles Starten von HACMP auf jedem Knoten in Betracht gezogen werden. Auf diese Weise können ausgefallene Knoten repariert und wieder integriert werden, ohne die anderen Knoten zu beeinträchtigen. Das Script `ha_cmd` wird zur Steuerung der HACMP-Befehle zum Starten und Stoppen über die Steuer-Workstation bereitgestellt.

Anmerkung: Wenn zum ersten Mal ein DB2-Exemplar erstellt wird, wird der folgende Eintrag an die Datei `/etc/inittab` angehängt:

```
rcdb2:2:once:/etc/rc.db2 > /dev/console 2>&1 # Autostart DB2 Services
```

Wenn HACMP oder HACMP ES aktiviert ist, aktualisieren Sie die Datei `/etc/inittab`, indem Sie die obige Zeile vor dem HACMP-Eintrag einordnen. Das folgende Beispiel zeigt einen HACMP-Eintrag in der Datei `/etc/inittab`:

```
clinit:a:wait:touch /usr/sbin/cluster/.telinit # HACMP for AIX
```

Der Eintrag muss der letzte Eintrag in der Datei `/etc/inittab` sein.

HACMP ES-Ereignisüberwachung und benutzerdefinierte Ereignisse

Das Herunterfahren von DB2-Datenbankpartitionen auf einem physischen AIX-Knoten, wenn der Paging-Bereich einen bestimmten Prozentsatz der Füllung erreicht, oder das erneute Starten einer DB2-Datenbankpartition bzw. die Einleitung einer Übernahmeoperation, wenn ein Prozess auf einem bestimmten Knoten unterbrochen wird, sind zwei Beispiele für benutzerdefinierte Ereignisse. Beispiele, die benutzerdefinierte Ereignisse illustrieren, wie zum Beispiel das Herunterfahren einer Datenbankpartition und das Erzwingen eines Transaktionsabbruchs, um Paging-Bereich freizugeben, befinden sich im Unterverzeichnis `samples`.

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 kommt es zu einem Blockieren des Knotens.

Das folgende Beispiel zeigt eine Ereignisdefinition für das Ereignis `node_up`:

```
##### Beginning of the Event Definition: node_up
#
TE_JOIN_NODE
0
/usr/sbin/cluster/events/node_up.rp
2
0
# 6) Resource variable - only used for event management events

# 7) Instance vector - only used for event management events

# 8) Predicate  - only used for event management events
```

```
# 9) Rearm predicate - only used for event management events

###### End of the Event Definition: node_up
```

Dies ist nur ein Beispiel für die Ereignisdefinitionen, die sich in der Datei `rules.hacmprd` befinden können. In diesem Beispiel wird das Wiederherstellungsprogramm `/usr/sbin/cluster/events/node_up.rp` ausgeführt, wenn das Ereignis `node_up` eintritt. Werte sind für den Status, den Wiederherstellungstyp und die Wiederherstellungsebene angegeben. Es gibt vier leere Zeilen für Ressourcenvariable, Exemplarvariable, Vergleichselement und Rearm-Vergleichselement.

Sie können andere Ereignisse definieren, um auf andere als die Standardereignisse von HACMP ES zu reagieren. Zum Beispiel muss die Datei `rules.hacmprd` geändert werden, um das Ereignis zu definieren, dass das Dateisystem `/tmp` zu über 90 % gefüllt ist.

Zahlreiche Ereignisse sind in IBM Parallel System Support Program (PSSP) vordefiniert. Diese Ereignisse können (durch Verwendung innerhalb benutzerdefinierter Ereignisse) wie folgt genutzt werden:

1. Stoppen Sie den Cluster.
2. Editieren Sie die Datei `rules.hacmprd`. Fertigen Sie eine Sicherungskopie der Datei an, bevor Sie sie ändern. Fügen Sie das vordefinierte PSSP-Ereignis manuell hinzu. Wenn Sie Synchronisierungspunkte über alle Knoten im Cluster hinweg benötigen, verwenden Sie im Wiederherstellungsprogramm den Befehl **barrier**. (Weitere Informationen zum Befehl **barrier** und zur Synchronisierung von Wiederherstellungsprogrammen finden Sie in den HACMP-Handbüchern zu Konzepten, Installation und Verwaltung.)
3. Starten Sie den Cluster erneut. Die Datei `rules.hacmprd` wird im Hauptspeicher gespeichert, wenn Cluster Manager gestartet wird. Starten Sie alle Cluster erneut, um die Änderungen präzise zu implementieren. Es sollten sich keine inkonsistenten Angaben in der rules-Datei in einem Cluster befinden.
4. Cluster Manager verwendet alle Ereignisse in der Datei `rules.hacmprd`.

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.

Die Ressourcenstatus werden durch verschiedene Ressourcenvariablen dargestellt. Ressourcenbedingungen werden durch Ausdrücke wiedergegeben, die als Vergleichselemente bezeichnet werden.

Die Ereignisverwaltung (Event Management) empfängt Ressourcenvariablen vom Ressourcenmonitor (Resource Monitor), der den Status spezieller Systemressourcen beobachtet und diesen Status in verschiedene Ressourcenvariablen umsetzt. Diese Variablen werden in regelmäßigen Abständen an die Ereignisverwaltung (Event Management) übermittelt. Die Ereignisverwaltung wendet Vergleichselemente, die durch HACMP ES Cluster Manager in der Datei `rules.hacmprd` angegeben werden, für jede Ressourcenvariable an. Wenn das Vergleichselement als wahr ausgewertet wird, wird ein Ereignis generiert und an Cluster Manager gesendet. Cluster Manager initiiert das Wahlprotokoll, und die Wiederherstellungsprogrammdatei (`xxx.rp`) wird (nach Ereignispriorität) auf einer Gruppe von Knoten ausgeführt, die durch ″node sets″ im Wiederherstellungsprogramm angegeben sind.

Die Wiederherstellungsprogrammdatei (*xxx*`.rp`) besteht aus mindestens einer Wiederherstellungsprogrammzeile. Jede Zeile wird im folgenden Format deklariert:

```
beziehung     auszuführender-befehl     erwarteter-status     NULL
```

Zwischen den einzelnen Werten in der Zeile muss mindestens ein Leerzeichen sein. ″beziehung″ ist ein Wert, der zu der Entscheidung herangezogen wird, welches Programm auf welchen Knoten auszuführen ist. Es werden drei Werte für die Beziehung unterstützt:

- All. Der angegebene Befehl oder das Programm wird auf allen Knoten des aktuellen HACMP-Clusters ausgeführt.
- Event. Der angegebene Befehl oder das Programm wird nur auf den Knoten ausgeführt, auf denen das Ereignis auftrat.
- Other. Der angegebene Befehl oder das Programm wird auf allen Knoten ausgeführt, auf denen das Ereignis *nicht* auftrat.

″auszuführender-befehl″ ist eine in Anführungszeichen gesetzte Zeichenfolge mit oder ohne Angabe eines vollständigen Pfads zu einem ausführbaren Programm. Nur zum Lieferumfang von HACMP gehörende Ereignis-Scripts können eine relative Pfaddefinition verarbeiten. Andere Scripts oder Programme müssen eine vollständige Pfadangabe verwenden, auch wenn sie sich im selben Verzeichnis wie die HACMP-Ereignis-Scripts befinden.

″erwarteter-status″ ist der Rückkehrcode des angegebenen Befehls bzw. Programms. Dabei kann entweder ein Ganzzahlwert oder ein ″x″ angegeben werden. Wenn ″x″ verwendet wird, ignoriert Cluster Manager den Rückkehrcode. Alle anderen Codes müssen mit dem erwarteten Rückkehrcode übereinstimmen, ansonsten erkennt Cluster Manager einen Ereignisfehler. Die Behandlung dieses Ereignisses blockiert den Prozess, bis eine Behebung (durch einen manuellen Eingriff) erfolgt. Ohne manuellen Eingriff wird der Knoten nicht

mit den anderen Knoten synchronisiert. Die Synchronisierung über alle Knoten hinweg ist eine Voraussetzung für Cluster Manager, um alle Knoten steuern zu können.

″NULL″ ist ein zur zukünftigen Verwendung reserviertes Feld. Das Wort ″NULL″ muss am Ende jeder Zeile außer der barrier-Zeile stehen. Wenn mehrere Wiederherstellungsbefehle zwischen zwei barrier-Befehlen (oder vor dem ersten) angegeben werden, werden die Wiederherstellungsbefehle auf dem Knoten selbst und zwischen den Knoten parallel ausgeführt.

Der Befehl barrier dient zur Synchronisierung aller Befehle über alle Clusterknoten hinweg. Wenn ein Knoten auf die barrier-Anweisung im Wiederherstellungsprogramm trifft, initiiert Cluster Manager das barrier-Protokoll auf diesem Knoten. Da das barrier-Protokoll ein zweiphasiges Protokoll ist, werden alle Knoten benachrichtigt, dass beide Phasen abschlossen wurden, wenn alle Knoten die barrier-Anweisung im Wiederherstellungsprogramm erreicht und ″gewählt″ haben, das Protokoll zu bestätigen.

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 Shell-Scripts oder Binärbefehle sein können. (In HACMP für AIX stimmen die Wiederherstellungsbefehle mit den HACMP-Ereignis-Scripts 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.

HACMP ES-Script-Dateien

Die folgenden Beispiel-Scripts zur Fehlerbehebung durch Funktionsübernahme und für benutzerdefinierte Ereignisse gehören zum Lieferumfang von DB2 UDB EEE. Die Script-Dateien befinden sich im Verzeichnis `$INSTNAME/sqllib/samples/hacmp/es`. Diese Scripts sind in der vorliegenden Form funktionsfähig. Die Wiederherstellungsaktion kann aber auch angepasst werden.

- Wiederherstellungs-Script für DB2-Datenbankpartition `rc.db2pe`. Dies ist die Script-Datei, die zum Starten und Stoppen der HACMP-Konfiguration in

einer Datenbankpartition verwendet wird. Das Script funktioniert auch als HACMP-Start- und-Stopp-Script für einen NFS-Server des DB2-Exemplareigners.

- DB2-spezifische benutzerdefinierte Ereignisse für HACMP ES. Sechs Standardereignisse sind bereitgestellt: eines für Prozesswiederherstellung, zwei für Paging-Bereich und drei für NFS- und Automount-Wiederherstellung (automatisches Anhängen von Verzeichnissen).
- Funktionsübernahme von einem NFS-Dateiserver des DB2-Exemplars. Dieses Script vollzieht die Funktionsübernahme und Wiederherstellung des Dateisystemservers für ein DB2-Exemplar durch einen Ausweichserver.
- Netzwerkfunktionsübernahme. Die Scripts `network_up_complete`, `network_back`, `network_down_complete` und `network_down` ermöglichen SP-DB2-Datenbankpartitionen im Fall eines Ausfalls ihres SP-Switch-Adapters eine Funktionsübernahme.
- Scripts zur Definition von Überwachungsereignissen für SP GUI Perspectives. Die Überwachung der Funktionsübernahme und der benutzerdefinierten Wiederherstellung wird durch Event Perspectives und Hardware Perspectives ermöglicht. Weitere Informationen zu Perspectives finden Sie in der Dokumentation für PSSP-Verwaltung (PSSP Administration).
- Scripts zum Installieren und Entfernen von Kern-Scripts (Core Scripts) und Ereignissen auf den HACMP ES-Knoten
- Script-Dateien zum Erstellen und Entfernen der Fehlerverwaltungsressourcen von SP Perspectives (pman) zur Überwachung der HACMP- und DB2-Konfiguration

Die Wiederherstellungs-Scripts müssen auf jedem Knoten installiert werden, der Wiederherstellungsoperationen ausführen soll. Die Script-Dateien können zentral von der SP-Steuer-Workstation bzw. einem anderen zuvor festgelegten SP-Knoten aus installiert werden:

1. Kopieren Sie die Scripts aus dem Verzeichnis `$INSTNAME/sqllib/samples/hacmp/es` entweder auf die SP-Steuer-Workstation oder auf einen anderen SP-Knoten, auf dem die Befehle **pcp** und **pexec** ausgeführt werden können. Diese Befehle sind für die Installationsoperation erforderlich.
2. Passen Sie die Dateien `reg.parms.SAMPLE` und `failover.parms.SAMPLE` an Ihre Umgebung an, indem Sie Werte für Schlüsselparameter (wie BUFFPAGE) für Übernahmekonfigurationen einstellen. Typischerweise gilt für Konfigurationen für gegenseitige Übernahme, dass die Einstellungen für den Störungsfall auf Werte unterhalb der Hälfte der normalen Einstellungen gesetzt werden. Außerdem müssen Sie eine in einen von Ihnen gewählten Namen (anstelle von ″SAMPLE″) umbenannte Kopie dieser Dateien verwenden.
3. Passen Sie (nach Bedarf) die fünf Parameter NFS_RETRIES, START_RETRIES, MOUNT_NFS, STOP_RETRIES und FAILOVER in der Datei

rc.db2pe an. Die RETRY- und FAILOVER-Einstellungen sollten für die meisten Implementierungen angemessen sein. Die Einstellung des Parameters MOUNT_NFS sollte abhängig davon konfiguriert werden, ob Sie das Paket für NFS-Serververfügbarkeit verwenden. Sie sollten diese Einstellung angeben, wenn Sie wollen, dass rc.db2pe das NFS-Ausgangsverzeichnis des DB2-Exemplareigners für Sie anhängt und überprüft. Die Einstellung des Parameters FAILOVER auf den Wert "YES" ruft db2_proc_restart auf und startet einen Versuch, eine DB2-Datenbankpartition erneut zu starten. Schlägt die Neustartoperation fehl, schließt HACMP mit einer Funktionsübernahme.

4. Passen Sie db2_paging_action, db2_proc_recovery und nfs_auto_recovery in der Ereignisdatei an. Editieren Sie pwq, um diese Einstellung in den DB2-Exemplareigner zu ändern. Passen Sie db2_paging_action an, um die Aktion anzugeben, die auszuführen ist, wenn der Paging-Bereich zu über 90 % gefüllt wird. (Wenn dies geschieht, wird die DB2-Datenbankpartition gestoppt.) Ändern Sie das Script, wenn weitere Wiederherstellungsaktionen erforderlich sind.
5. Verwenden Sie db2_inst_ha zur Installation der Scripts und Ereignisse auf den Knoten, die Sie angeben. (HACMP ES muss auf den Knoten vorinstalliert sein, bevor Sie beginnen.) Die Syntax von db2_inst_ha lautet:

   ```
   db2_inst_ha $INSTNAME/sqllib/samples/hacmp/es <knotenliste> <DATENBANKNAME>

   Dabei gilt Folgendes:

      $INSTNAME/sqllib/samples/hacmp/es
         ist das Verzeichnis mit den Scripts und den Ereignissen
      <knotenliste> ist die Knotenangabe im pcp- oder pexec-Format;
         z. B. 1-16 oder 1,2,3,4
      <DATENBANKNAME> ist der Name der Datenbank für die Dateien mit den
         regulären Parametern und den Funktionsübernahmeparametern.
   ```

   Die Dateien reg.parms.SAMPLE und failover.parms.SAMPLE werden auf jeden Knoten kopiert und in reg.parms.DATENBANKNAME umbenannt. db2_inst_ha kopiert Dateien auf jeden Knoten in das Verzeichnis /usr/bin und aktualisiert die HACMP-Ereignisdateien:

   ```
   /usr/sbin/cluster/events/rules.hacmprd
   /usr/sbin/cluster/events/network_up_complete
   /usr/sbin/cluster/events/network_down_complete
   ```

6. Konfigurieren Sie Ihr System und die Scripts mit HACMP.
7. Verwenden Sie den Befehl **create_db2_events**, um die Überwachungsereignisse für Fehlerverwaltungsressourcen (pman) und SP GUI Perspectives zu installieren. Es sind noch weitere Konfigurations- und Anpassungsschritte in Perspectives erforderlich. Weitere Informationen zu Perspectives finden Sie im Handbuch *PSSP Administration Guide*.
8. Verwenden Sie den Befehl ha_db2stop, um die Datenbankpartitionen herunterzufahren, ohne dass eine HACMP ES-Fehlerbehebung durch

Funktionsübernahme stattfindet. Kopieren Sie zur Verwendung dieses Befehls die Datei in das Ausgangsverzeichnis des Datenbankbenutzers, und stellen Sie sicher, dass Berechtigungen und Eigentumsrecht für den betreffenden Benutzer definiert sind. Geben Sie dann als der betreffende Benutzer zum Stoppen der Datenbank ohne Fehlerbehebung durch Funktionsübernahme folgenden Befehl ein:

```
ha_db2stop
```

Anmerkung: Sie müssen warten, bis der Befehl beendet ist. Durch Drücken der Tasten `Strg-C` oder durch Abbrechen des Prozesses kann die Fehlerbehebung durch Funktionsübernahme vorzeitig wieder aktiviert werden, und einige Datenbankpartitionen werden vielleicht nicht gestoppt.

Operationen der DB2-Wiederherstellungs-Scripts mit HACMP ES

HACMP ES ruft die DB2-Wiederherstellungs-Scripts auf folgende Weise auf:

- `node_up_local` (beim Start eines Knotens)

  HACMP fordert mit der `node_up-Sequenz` Datenträgergruppen, logische Datenträger, Dateisysteme und IP-Adressen an, die in den Ressourcengruppen angegeben sind, deren Eigner (durch Hintereinanderschalten (Cascading)) der Knoten ist bzw. die diesem Knoten (durch Rotation) zugeordnet sind.

  Wenn `node_up_local_complete` ausgeführt wird, wird die Anwendungsserverdefinition, die `rc.db2pe` enthält, initiiert, um die Datenbankpartition zu starten, die in den Anwendungsserverdefinitionen auf diesem physischen Knoten angegeben ist.

  Anmerkung: `rc.db2pe` passt bei Ausführung im Startmodus die in der Datei `reg.parms.DATENBANK` angegebenen DB2-Parameter für jede DATENBANK im Datenbankverzeichnis an, die einer Parameterdatei (parms-Datei) entspricht.

  Auf jedem Knoten spielt sich diese Abfolge von Aktionen beim Start ab. Wenn Sie mehrere HACMP-Cluster haben und diese parallel starten, werden mehrere Knoten gleichzeitig hochgefahren.

- `node_down_remote` (Funktionsübernahme)

  HACMP fordert die Datenträgergruppen, logischen Datenträger, Dateisysteme und IP-Adressen an, die in der Ressourcengruppe auf dem designierten Übernahmeknoten angegeben sind.

  Bei Ausführung von `node_down_remote_complete` führt HACMP `rc.db2pe` als Start-Script für den Anwendungsserver aus, das in der Ressourcengruppe für diese Datenbankpartition angegeben ist.

  Anmerkung: Bei Ausführung im Modus der gegenseitigen Übernahme stoppt `rc.db2pe` die auf dem Knoten ausgeführte DB2-

Datenbankpartition, passt die DB2-Parameter in `failover.parms.DATENBANK` für jede DATENBANK im Datenbankverzeichnis an, die einer Parameterdatei (parms) entspricht, und startet anschließend beide Datenbankpartitionen auf dem physischen Übernahmeknoten.

- `node_up_remote` (Reintegration eines ausgefallenen Knotens - Ressourcengruppe für hintereinandergeschaltete gegenseitige Übernahme)

  Bei der Ausführung von `node_up_remote` auf dem alten Übernahmeknoten bewirkt die Anwendungsserverdefinition, dass `rc.db2pe` im Stoppmodus ausgeführt wird.

  Anmerkung: Wenn `rc.db2pe` in einem Reintegrationsmodus (bei gegenseitiger Übernahme) ausgeführt wird, stoppt `rc.db2pe` beide auf dem Knoten aktiven Datenbankpartitionen, passt die DB2-Parameter an, die in `reg.parms.DATENBANK` für jede Datenbank im Datenbankverzeichnis angegeben sind, die einer Parameterdatei (parms) entspricht, und startet anschließend nur die Datenbankpartition, die auf diesem physischen Übernahmeknoten behalten werden soll.

  Der alte Übernahmeknoten gibt die Datenträgergruppen, logischen Datenträger, Dateisysteme und IP-Adressen frei, die in Ressourcengruppen, deren Eigner der wieder integrierte Knoten sein soll, angegeben sind.

  HACMP fordert erneut Datenträgergruppen, logische Datenträger, Dateisysteme und IP-Adressen an, die in der Ressourcengruppe angegeben sind, deren Eigner jetzt der sich wieder integrierende Knoten ist.

  Wenn `node_up_local_complete` ausgeführt wird, wird die Anwendungsserverdefinition initiiert, die `rc.db2pe` enthält, um die in der Anwendungsserverdefinition auf diesem wieder zu integrierenden physischen Knoten angegebene DB2-Datenbankpartition zu starten.

  Anmerkung: `rc.db2pe` passt bei Ausführung im Startmodus die in der Datei `reg.parms.DATENBANK` angegebenen DB2-Parameter für jede DATENBANK im Datenbankverzeichnis an, die einer Parameterdatei (parms-Datei) entspricht.

- `node_down_local` (Knotenstopp oder Stopp mit Übernahme)

  Wenn `node_down_local` auf dem zu stoppenden Knoten ausgeführt wird, bewirkt die Anwendungsserverdefinition, dass `rc.db2pe` im Stoppmodus ausgeführt wird.

  Anmerkung: Bei Ausführung im Stoppmodus passt `rc.db2pe` die DB2-Parameter an, die in `failover.parms.DATENBANK` für jede DATENBANK im Datenbankverzeichnis angegeben sind, die einer

Parameterdatei (parms) entspricht, und stoppt anschließend die DB2-Datenbankpartition (dies geschieht für die Übernahme).

HACMP gibt die Datenträgergruppen, logischen Datenträger, Dateisysteme und IP-Adressen frei, die in Ressourcengruppen angegeben sind, deren Eigner jetzt der Knoten ist.

- `db2_proc_recovery` (DB2-Prozessabbruch)

  Alle Knoten führen das Script `db2_proc_restart` aus. Der Knoten, auf dem der Fehler auftrat, startet die richtige DB2-Datenbankpartition.

- `db2_paging_recovery` (Wiederherstellung von Paging-Bereich)

  Alle Knoten führen das Script `db2_paging_action` aus. Wenn auf einem Knoten mehr als 70 % des Paging-Bereichs gefüllt sind, wird ein wall-Befehl abgesetzt. Wenn auf einem Knoten mehr als 90 % des Paging-Bereichs gefüllt sind, werden die DB2-Datenbankpartitionen auf diesem physischen Knoten gestoppt und dann erneut gestartet.

- `nfs_auto_recovery` (Fehler in NFS-Prozess oder beim automatischen Anhängen)

  Alle Knoten führen das Script `rc.db2pe` im NFS-Modus aus. Wird die Ausführung eines NFS-Prozesses gestoppt, wird er erneut gestartet. In ähnlicher Weise wird der Prozess für automatisches Anhängen bei einem Ausfall erneut gestartet.

- `network_down_complete` (Netzwerkfehler - SP-Switch)

  Das Script `net_down` wird aufgerufen. Dieses Script überprüft, ob das Netzwerk ein SP-Switch-Netzwerk und außer Funktion ist. Ist dies der Fall, wartet das Script ein benutzerdefiniertes Zeitintervall ab. Das Standardzeitintervall sind 100 Sekunden.

  Ist das SP-Switch-Netzwerk wieder verfügbar, wie durch das Ereignis `network_up_complete` mitgeteilt wird, wird keine Wiederherstellung durchgeführt.

  Wenn das Zeitlimit erreicht ist, wird HACMP mit Funktionsübernahme gestoppt.

Anmerkung: Alle Ereignisse können über die SP-Fehlerverwaltung (Problem Management) und die SP Perspectives-GUI überwacht werden.

Andere Script-Dienstprogramme

Es stehen noch weitere Script-Dienstprogramme für Sie bereit:

- `ha_cmd` ist ein Befehl, der zum Starten von HACMP auf SP-Knoten über die Steuer-Workstation dient. Die Syntax ist wie folgt:

  ```
  ha_cmd <knotenbereich> <START|STOP|TAKE|FORCE>

  Dabei gilt Folgendes:
  ```

```
<knotenbereich> ist ein SP-Knotenbereich im pcp- oder pexec-Format.
   Z. B.: "ha_cmd 3-6 START" startet HACMP auf den Knoten 3,4,5,6.
          "ha_cmd 5 TAKE" beendet HACMP auf Knoten 5
          für gegenseitige Übernahme.
```

- `ha_mon` ist ein Befehl zur Überwachung von `hacmp_out`-Dateien von HACMP von der SP-Steuer-Workstation aus. Die Syntax ist wie folgt:

  ```
  ha_mon <knoten>

  Dabei gilt Folgendes:

     <knoten> ist der zu überwachende SP-Knoten.

     ha_mon wendet "tail -f" auf die Datei /tmp/hacmp.out auf dem
     angegebenen Knoten an.
  ```

- `db2_turnoff_recov` ist ein Befehl zur temporären Inaktivierung sämtlicher HACMP-Wiederherstellung (ohne Funktionsübernahme), der für äußerst seltene Fälle gedacht ist. Es werden keine Wiederherstellungen für DB2-Prozesse, Seitenwechsel (Paging), NFS-Prozesse oder automatisches Anhängen (Automount) eingeleitet. Diese Funktion entfernt die Ereigniszeilengruppen für die betreffende Wiederherstellung aus der rules-Datei von HACMP. HACMP muss gestoppt und erneut gestartet werden. Die Syntax ist wie folgt:

  ```
  db2_turnoff_recov <knotenliste>
  ```

- `db2_turnon_recov` ist ein Befehl zur Wiederaktivierung der HACMP-Wiederherstellung (ohne Funktionsübernahme). Dieser Befehl dient nach Ausführung des Befehls `db2_turnoff_recov` zur Wiederherstellung der rules-Datei von HACMP, so dass die Wiederherstellung nach benutzerdefinierten Ereignissen wieder erfolgen kann. HACMP muss gestoppt und erneut gestartet werden. Die Syntax ist wie folgt:

  ```
  db2_turnon_recov <knotenliste>
  ```

Überwachen von HACMP-Clustern

Es wurden Scripts vorbereitet, mit denen Ereignisse für die SP-Fehlerverwaltung (pman) erstellt werden können, um neben den in HACMP ES bereits vorhandenen Überwachungsprogrammen die HACMP-SP-Konfiguration für DB2 zu überwachen. Zur Überwachung des HACMP-Status von der SP-Steuer-Workstation aus sind folgende Schritte erforderlich:

- Installieren Sie den HACMP-Client-Code auf der Steuer-Workstation.
- Editieren Sie die Datei `/usr/sbin/cluster/etc/clhosts`, und fügen Sie die Ethernet-SP-IP-Adressen der Knoten ein, die Sie überwachen wollen.
- Rufen Sie den Befehl `startsrc -s clinfo` auf, um die Überwachung der Cluster zu starten.

HACMP stellt eine Schnittstelle zur Überwachung der Cluster bereit (`/usr/sbin/cluster/clstat`).

Die Überwachung der Fehlerverwaltung mit der GUI von SP Perspectives für HACMP ES-Ereignisse und benutzerdefinierte Ereignisse wird folgendermaßen verwendet:

1. Rufen Sie den Befehl `create_db2_events <knotenliste>` auf, wobei *knotenliste* Knoten im pcp- oder pexec-Format enthält. Dieses Script erstellt fünf pman-Ereignisse für die Überwachung durch Perspectives.

   Anmerkung: Die Ressourcenvariablen `PSSP.pm.User_state12-16` werden zur Erstellung dieser Ereignisse verwendet. Wenn diese Ressourcenvariablen bereits für einen anderen Zweck verwendet werden, müssen `create_db2_events` und `update_db2_events` zur Verwendung anderer Variablen aktualisiert werden.
2. Starten Sie Perspectives auf der Steuer-Workstation. Wählen Sie über die Klickstartleiste die Perspective für Ereignisse aus. Es sollten fünf Ereignisse angezeigt werden: `db2_hacmp_recovery`, `db2_process_recovery`, `db2_paging_err`, `db2_nfs_err` und `Errlog_PERM_entry`.
3. Klicken Sie jedes Ereignis doppelt an. Registrieren Sie in der daraufhin erscheinenden Anzeige (innerhalb der Definitionstabelle) eine Bedingung für das Ereignis. Klicken Sie neben dem Abwärtspfeil bei `Name: "unnamed"`, und wählen Sie denselben Namen aus wie für das Ereignis, das Sie als Bedingung angeben. Wählen Sie die Indexzunge `Response Options` aus. Klicken Sie den Druckknopf oben in der Anzeige an (″Send Message to Perspectives event session″). Sie können Befehle, Errlog-Einträge sowie SNMP-Traps für Vorkommen dieser Ereignisse angeben. Die Ereignisprotokollanzeigen bleiben nur über Sitzungen von Perspective hinweg erhalten. Daher kann es wünschenswert sein, für jede Sitzung AIX-Fehlerprotokolleinträge zu erstellen. Wählen Sie **OK** aus, und schließen Sie das Fenster.
4. Wählen Sie über die Klickstartleiste von Perspectives die Perspective für Hardware aus.
5. Wenn die GUI für den Hardwarerahmen angezeigt wird, wählen Sie ″View″ und anschließend ″Monitor″ aus. Sie erhalten eine Liste von Ereignissen, die für Ihr SP überwacht werden können. Blättern Sie zum Ende der Liste. Dort sind folgende zusätzliche Ereignisse aufgeführt: eines für die HACMP DB2-Wiederherstellung (`db2_ha_ind`) und das andere für SP-Knoten-PERM-Fehler (`Errlog_PERM_mon`). Wählen Sie die Ereignisse aus, die Sie überwachen wollen. (Wenn ein Ereignis eintritt, zeigt der Knoten ein rotes ″X″ an. Wenn alle überwachten Bedingungen in Ordnung sind, ist die Knotenanzeige grün.) In der Regel werden `host_responds`, `switch_responds` und `node_power_LED` verwendet. Darüber hinaus können Sie die DB2-HACMP-Wiederherstellung und die PERM-Fehler auf dem Knoten überwachen.

Anmerkung: Die Variablen `db2_hacmp_mon` und `db2_hacmp_recovery` für pman und Perspectives geben nicht den HACMP-Clusterstatus wieder. Diese Variablen zeigen statt dessen den Status der `rc.db2pe`-Operation zum Starten bzw. Stoppen von DB2 an. Der „echte" HACMP-Status wird vom HACMP-Monitor `clstat` gezeigt, der den HACMP-Clusterstatus wiedergibt. Wenn Sie wollen, dass `db2_hacmp_ind` Überwachungswerte liefert, die dem HACMP-Status nahe kommen, fügen Sie die folgende Zeile in Ihre Datei `/etc/inittab` ein:

```
haind:2:wait:/usr/bin/db2_update_events HAIND OFF 2>&1 >/dev/null
```

Wenn Sie die Verwendung von NetView für Ihre Implementierung planen, sollten Sie die Verwendung von HAVIEW (gehört zu HACMP) zur Überwachung Ihrer Konfiguration in Betracht ziehen. Informationen zur Konfiguration dieses Produkts entnehmen Sie bitte der NetView-Dokumentation.

Installation von SP HACMP ES unter DB2

Im Folgenden finden Sie eine Übersicht in Einzelschritten über die Installations- und Migrationsprozesse, die Ihnen bei der Planung für die Installation von HACMP ES mit DB2 Universal Database Hilfestellung leisten soll.

Neuinstallation von SP HACMP ES unter DB2

Gehen Sie wie folgt vor, um HACMP ES zu installieren:

1. Installieren Sie das Betriebssystem AIX auf jedem SP-Knoten (nach den Anweisungen der Handbücher zur SP-Installation und SP-Verwaltung). Sorgen Sie dafür, dass ein angemessener Paging-Bereich auf der Steuer-Workstation und auf jedem der SP-Knoten verfügbar ist. Stellen Sie sicher, dass die Switch-Konfiguration zusammen mit allen anderen änderbaren Konfigurationsparametern durchdacht und implementiert ist. Richten Sie die SP-Überwachung (Perspectives) ein, die Sie verwenden wollen. Stellen Sie sicher, dass die SP-Befehle dsh, pcp und pexec funktionieren.
2. Legen Sie das Datenbanklayout fest. Dies beinhaltet mindestens die Anzahl der zu verwendenden Knoten, die Zuordnung von DB2-Datenbankpartitionen zu physischen Knoten, die Plattenvoraussetzungen pro Knoten oder Partition und Überlegungen zu Tabellenbereichen. Außerdem sollten Sie überlegen, wer der DB2-Hauptexemplareigner sein soll und welche Zugriffsberechtigungen für diesen und andere Benutzer erforderlich sein sollen.
3. Planen Sie Ihre externe SSA-Plattenkonfiguration, einschließlich redundanter Adapter, gespiegelter Platten und Zwillingskonfiguration von Platten.

4. Füllen Sie anhand Ihres Datenbanklayouts und Ihrer SSA-Konfiguration die HACMP-Arbeitsblätter in den Handbüchern zur HACMP-Planung, Installation und Verwaltung aus.
5. Implementieren Sie Ihre externe SSA-Plattenkonfiguration. Stellen Sie sicher, dass die Mikrocodeversionen über alle Laufwerke hinweg konsistent sind, und verwenden Sie das Dienstprogramm Maymap zum Prüfen und Ausfüllen etwaiger Lücken in Ihren Arbeitsblättern.
6. Installieren Sie DB2 UDB EEE auf jedem SP-Knoten.
7. Installieren Sie HACMP ES auf jedem SP-Knoten.
8. Installieren Sie das Paket DB2 UDB EEE HACMP ES on SP Package mit Hilfe des Befehls **db2_inst_ha**.
9. Erstellen Sie den DB2-Hauptexemplarbenutzer, und stellen Sie sicher, dass er auf alle Knoten zugreifen kann. Zu diesem Zeitpunkt ist dies kein hochverfügbarer Benutzer. Es kann sich vorläufig um einen SP-Benutzer auf der SP-Steuer-Workstation handeln.
10. Erstellen Sie Ihr DB2-Exemplar und Ihre Datenbank. Stellen Sie sicher, dass es betriebsbereit ist, indem Sie den Befehl **db2start** und anschließend den Befehl **db2stop** aufrufen, bevor Sie mit dem nächsten Schritt fortfahren.
11. Wenn Sie die Datenbank vor dem Hinzufügen von HACMP mit Daten füllen wollen, tun Sie dies jetzt.
12. Konfigurieren Sie anhand der HACMP-Arbeitsblätter und der Informationen in diesem Dokument HACMP ES in der SP-Knotentopologie und den Ressourcengruppen.
13. Ändern Sie den gegenwärtigen Benutzer, angefangen bei Ihrem NFS-Serverknoten für den DB2-Hauptexemplarbenutzer, indem Sie auf allen Knoten `/etc/security/user` und `/etc/passwd` nach den Angaben in diesem Dokument ändern. Dieser Benutzer wird zu einem hochverfügbaren NFS-Benutzer. Dieser Knoten und der zugehörige Ausweichknoten aktualisieren `/etc/exports`. Alle Knoten werden in die Lage versetzt, dieses Verzeichnis mit Hilfe von NFS (mit einem Eintrag in `/etc/filesystems` auf jedem Knoten) über die IP-Aliasadressen des Switch anzuhängen.
14. Packen Sie das Ausgangsverzeichnis des Hauptexemplareigners mit "tar", und entpacken Sie es an der neuen Speicherposition.
15. Erstellen Sie ein NFS-Dateisystem auf jedem der SP-Knoten, um ein neues Exemplarausgangsverzeichnis anzuhängen.
16. Starten Sie HACMP auf dem NFS-Serverknoten. Überprüfen Sie anhand der Datei `/tmp/hacmp.out`, ob HACMP erfolgreich anläuft. Der Befehl **ha_mon** kann zur Überwachung der Schreibvorgänge für diese Datei verwendet werden.

17. Starten Sie nacheinander die anderen Knoten, und überprüfen Sie den erfolgreichen Anlauf durch Untersuchen der Datei `/tmp/hacmp.out`. Der Befehl **ha_mon** kann zur Überwachung der Schreibvorgänge für diese Datei verwendet werden.
18. Richten Sie die optionale Überwachung durch Perspectives und Fehlerverwaltung (Problem Management) ein.
19. Überprüfen Sie die Übernahmefunktionalität auf jedem Knoten, indem Sie eine parallele Wartungsaktion auf jedem Knoten simulieren. Der Befehl **ha_cmd** (unter Angabe der Option TAKE) kann dazu verwendet werden, HACMP auf ordnungsgemäße Weise mit Funktionsübernahme zu stoppen. Überprüfen Sie anhand der Datei `/tmp/hacmp.out` und mit Hilfe Ihrer Überwachungsprogramme den Erfolg der Funktionsübernahmen und Reintegrationen.

Migration von SP HACMP ES für DB2

Wenn Sie eine Migration von einer Nicht-HACMP-Installation auf eine Installation mit HACMP durchführen wollen, machen Sie sich mit der folgenden Übersicht vertraut:

1. Wandeln Sie die Konfiguration Ihrer externen Platten in eine hochverfügbare, gespiegelte Zwillingskonfiguration um. Fügen Sie zusätzliche Hardware und Platten hinzu, um diese Konfiguration herzustellen. Beachten Sie, dass Namen verschiedener logischer Datenträger auf verschiedenen Knoten innerhalb einer Zwillingskonfiguration eindeutig sein *müssen*. Dies gilt für Datenträgergruppen, logische Datenträger und Dateisysteme.
2. Schließen Sie die HACMP-Planung ab, und füllen Sie die entsprechenden Arbeitsblätter in diesem Dokument aus.
3. Implementieren Sie die Änderungen an Ihrer externen SSA-Plattenkonfiguration. Stellen Sie sicher, dass die Mikrocodeversionen über alle Laufwerke hinweg konsistent sind, und verwenden Sie das Dienstprogramm Maymap zum Prüfen und Ausfüllen etwaiger Lücken in Ihren Arbeitsblättern.

   Anmerkung: SSA-Platten in einer RAID-5-Konfiguration werden unterstützt. Die einzig zulässige Konfiguration sind zwei SSA-Adapter in derselben RAID-Schleife. Für eine HACMP-Konfiguration mit einer Zwillingskonfiguration der RAID-Platten wird nur ein Adapter pro Knoten unterstützt. In dieser Konfiguration ist der Adapter ein einzelner Fehlerpunkt beim Zugriff auf die Platten. Weitere Konfigurationsmaßnahmen zur Erkennung von Adapterausfällen und zu deren Umstufung in ein HACMP-Funktionsübernahmeereignis werden deshalb empfohlen. AIX-Fehleraufzeichnung ist die einfachste Möglichkeit, einen Knoten für die Funktionsübernahme zu konfigurieren, falls der SSA-

Adapter ausfallen sollte. Weitere Informationen zur AIX-Fehleraufzeichnung finden Sie im Handbuch *HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide*.

4. Installieren Sie HACMP ES auf jedem SP-Knoten.
5. Installieren Sie das Paket DB2 UDB EEE HACMP ES on SP Package mit Hilfe des Befehls **db2_inst_ha**.
6. Konfigurieren Sie anhand der HACMP-Arbeitsblätter und der Informationen in diesem Dokument HACMP ES in der SP-Knotentopologie und den Ressourcengruppen.
7. Ändern Sie den gegenwärtigen Benutzer, angefangen bei Ihrem NFS-Serverknoten für den DB2-Hauptexemplarbenutzer, indem Sie auf allen Knoten `/etc/security/user` und `/etc/passwd` nach den Angaben in diesem Dokument ändern. Dieser Benutzer wird zu einem hochverfügbaren NFS-Benutzer. Dieser Knoten und der zugehörige Ausweichknoten aktualisieren `/etc/exports`. Alle Knoten werden in die Lage versetzt, dieses Verzeichnis mit Hilfe von NFS (mit einem Eintrag in `/etc/filesystems` auf jedem Knoten) über die IP-Aliasadressen des Switch anzuhängen.
8. Packen Sie das Ausgangsverzeichnis des Hauptexemplareigners mit "tar", und entpacken Sie es an der neuen Speicherposition.
9. Erstellen Sie ein NFS-Dateisystem auf jedem der SP-Knoten, um ein neues Exemplarausgangsverzeichnis anzuhängen.
10. Starten Sie HACMP auf dem NFS-Serverknoten. Überprüfen Sie anhand der Datei `/tmp/hacmp.out`, ob HACMP erfolgreich anläuft. Der Befehl **ha_mon** kann zur Überwachung der Schreibvorgänge für diese Datei verwendet werden.
11. Starten Sie nacheinander die anderen Knoten, und überprüfen Sie den erfolgreichen Anlauf durch Untersuchen der Datei `/tmp/hacmp.out`. Der Befehl **ha_mon** kann zur Überwachung der Schreibvorgänge für diese Datei verwendet werden.
12. Richten Sie die optionale Überwachung durch Perspectives und Fehlerverwaltung (Problem Management) ein.
13. Überprüfen Sie die Übernahmefunktionalität auf jedem Knoten, indem Sie eine parallele Wartungsaktion auf jedem Knoten simulieren. Der Befehl **ha_cmd** (unter Angabe der Option TAKE) kann dazu verwendet werden, HACMP auf ordnungsgemäße Weise mit Funktionsübernahme zu stoppen. Überprüfen Sie anhand der Datei `/tmp/hacmp.out` und mit Hilfe Ihrer Überwachungsprogramme den Erfolg der Funktionsübernahmen und Reintegrationen.

SP HACMP ES-Arbeitsblätter für DB2

Die folgenden Arbeitsblätter sind für die Verwendung mit den HACMP-Arbeitsblättern vorgesehen, die bei der Vorbereitung der externen SSA-Plattenkonfiguration ausgefüllt werden sollten (und die in den HACMP-Handbüchern zur Planung, Installation und Verwaltung zu finden sind). Es ist jeweils ein ausgefülltes Beispiel und ein leeres Arbeitsblatt vorgesehen.

Die Datenbankkonfiguration auf externen Platten, die auf dem ersten Musterarbeitsblatt dokumentiert ist, wird in der folgenden Abbildung gezeigt. Die Anweisung zur Erstellung der Datenbank lautet:

```
db2 create database pwq on /newdata
```

Sowohl die externen SSA-Adapter als auch die externen SSA-Platten werden gespiegelt und in einer Zwillingskonfiguration für logische Datenträger ohne einen einzelnen Fehlerpunkt konfiguriert. Die Grafik zeigt eine Konfiguration in ähnlicher Weise wie die Ausgabe des Befehls **maymap**. Maymap ist ein Dienstprogramm (über AIXTOOLS verfügbar), das die externe SSA-Plattenkonfiguration anzeigt und bei der Planung der Konfiguration eingesetzt werden sollte.

Beispielkonfiguration für externe Platten einer DB2-Datenbank mit vier Knoten

- Darstellung der Zwillingskonfiguration für hohe Verfügbarkeit

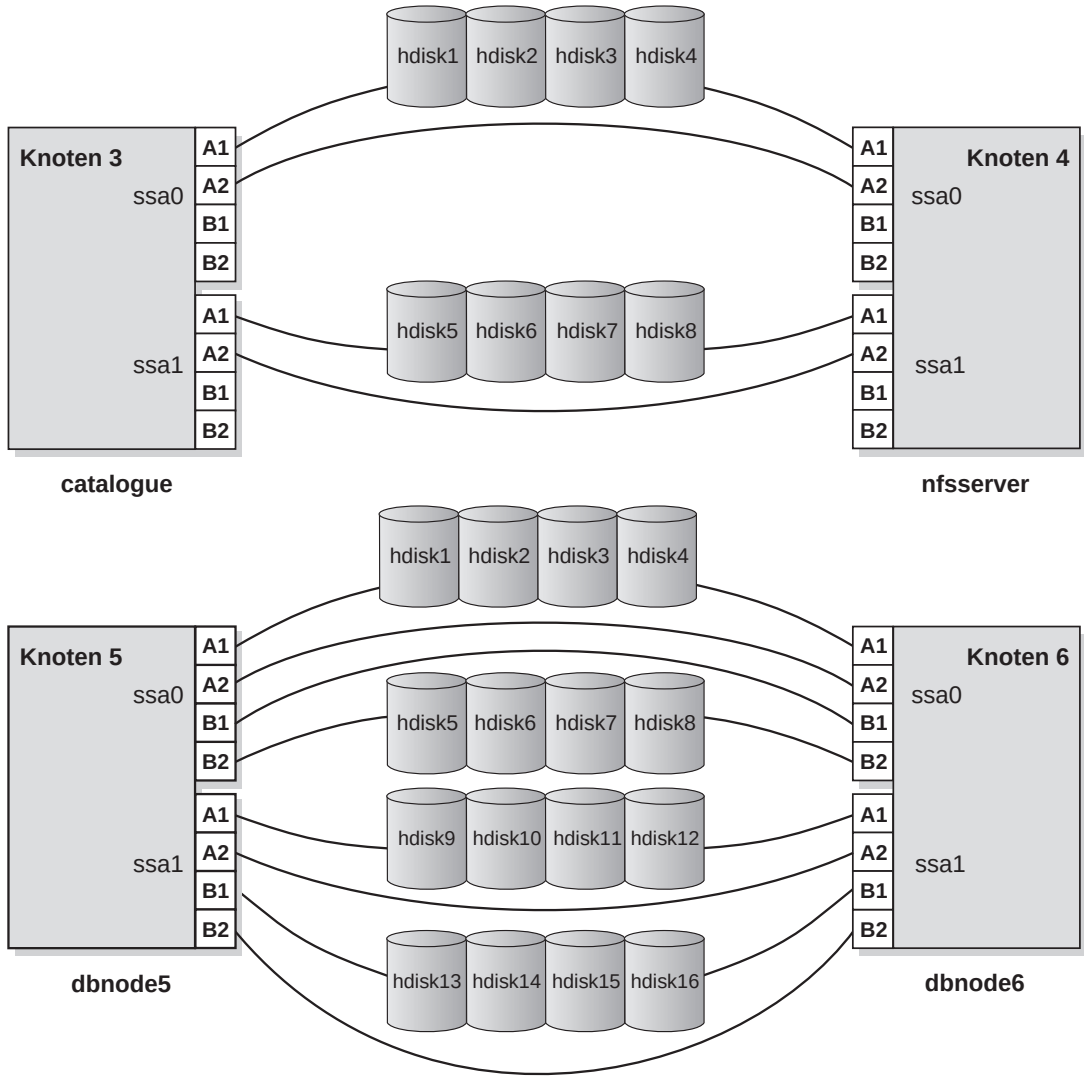


Abbildung 27. Beispielkonfiguration für externe Platten einer DB2-Datenbank mit 4 Knoten

Zum besseren Verständnis der folgenden Tabelle sollten Sie die HACMP-Dokumentation zu den Themen Quorumeinstellungen (Quorum Settings) für Datenträgergruppen und Konsistenzeinstellungen für gespiegelte Schreibvorgänge auf logischen Datenträgern (Mirrored Write Consistency Settings On Logical Volumes) gut durchlesen. Die Einstellungen beider Bereiche haben direkte Auswirkungen auf die Verfügbarkeit und Leistung Ihrer Konfiguration. Machen Sie sich unbedingt mit diesen Einstellungen und ihrer Bedeutung vertraut. Die typische Einstellung für ″quorum″ und ″mirrored write consistency″ ist ″off″.

Tabelle 12. Datenträgergruppen, logische Datenträger und Dateisysteme für HACMP

SP-Knoten	Name der Datenträgergruppe	PP-Größe (MB)	Name des logischen Datenträgers	# von PPs	Kopien	hdisk-Liste	Mount-Punkt für Dateisystem	Logischer Datenträger für Dateisystemprotokoll	Knotenbeschreibung und Ausweichknoten	Benut-.hr zer/ Eigner der logischen Einheit
3	havg3	8	hlv300	10	2	hdisk1 hdisk5	/newdata /pwq /NODE0003	hlog301	Katalog-knoten-Mount-Punkt; Knoten 4	root *
3	havg3	8	hlog301	1	2	hdisk1 hdisk5	N/V	N/V	Katalog-knoten jfslog; Knoten 4	root *
3	havg3	8	hlv301	10	2	hdisk2 hdisk6	N/V	N/V	Unformatierter temp. Bereich für Katalog-knoten; Knoten 4	pwq **
4	havg4	8	hlv400	10	2	hdisk3 hdisk7	/dbmnt	hlog401	nfsserver pwq Ausgangsverzeichnis; Knoten 3	root *
4	havg4	8	hlog401	1	2	hdisk3 hdisk7	N/V	N/V	nfsserver jfslog; Knoten 3	root *
5	havg5	8	hlv500	10	2	hdisk1 hdisk9	/newdata/ pwq/ NODE0005	HLOG501	Dbnode5-Mount-Punkt; Knoten 6	root *

Tabelle 12. Datenträgergruppen, logische Datenträger und Dateisysteme für HACMP (Forts.)

SP-Knoten	Name der Datenträgergruppe	PP-Größe (MB)	Name des logischen Datenträgers	# von PPs	Kopien	hdisk-Liste	Mount-Punkt für Dateisystem	Logischer Datenträger für Dateisystemprotokoll	Knotenbeschreibung und Ausweichknoten	Benut-.hr zer/ Eigner der logischen Einheit
5	havg5	8	hlog501	1	2	hdisk1 hdisk9	N/V	N/V	Dbnode5 jfslog; Knoten 6	root *
5	havg5	8	hlv501	10	2	hdisk2 hdisk10	N/V	N/V	Unformatierter temp. Bereich für Dbnode5; Knoten 6	pwq **
5	havg5	8	hlv502	100	2	hdisk2 hdisk10	N/V	N/V	Unformaierter Tabellen- bereich für Dbnode5; Knoten 6	pwq **
5	havg5	8	halv503	100	2	hdisk3 hdisk11	N/V	N/V	Unformatierter Tabellen- bereich für Dbnode5; Knoten 6	pwq **
5	havg5	8	halv504	100	2	hdisk3 hdisk11	N/V	N/V	Unformatierter Tabellen- bereich für Dbnode5; Knoten 6	pwq **
5	havg5	8	halv505	100	2	hdisk4 hdisk12	/dbdata5	hlog501	Dbnode6-Systemtabellenbereich; Knoten 6	root *
6	havg6	8	hlv600	10	2	hdisk5 hdisk13	/newdata/ pwq/ NODE0006	hlog601	Dbnode6-Mount-Punkt; Knoten 5	root *

Tabelle 12. Datenträgergruppen, logische Datenträger und Dateisysteme für HACMP (Forts.)

SP-Knoten	Name der Datenträgergruppe	PP-Größe (MB)	Name des logischen Datenträgers	# von PPs	Kopien	hdisk-Liste	Mount-Punkt für Dateisystem	Logischer Datenträger für Dateisystemprotokoll	Knotenbeschreibung und Ausweichknoten	Benut-.hr zer/ Eigner der logischen Einheit
6	havg6	8	hlog601	1	2	hdisk5 hdisk13	N/V	N/V	Dbnode6 jfslog; Knoten 5	root *
6	havg6	8	hlv601	10	2	hdisk6 hdisk14	N/V	N/V	Unformatierter temp. Bereich für Dbnode6; Knoten 5	pwq **
6	havg6	8	hlv602	100	2	hdisk6 hdisk14	N/V	N/V	Unformatierter Tabellenbereich für Dbnode6; Knoten 5	pwq **
6	havg6	8	hlv603	100	2	hdisk7 hdisk15	N/V	N/V	Unformatierter Tabellenbereich für Dbnode6; Knoten 5	pwq **
6	havg6	8	hlv604	100	2	hdisk7 hdisk15	N/V	N/V	Unformatierter Tabellenbereich für Dbnode6; Knoten 5	pwq **
6	havg6	8	hlv605	100	2	hdisk8 hdisk16	/dbdata6	hlog601	Systemtabellenbereich für Dbnode6; Knoten 5	root *

Anmerkungen:

1. * Logische Datenträger und Protokolle für JFS-Dateisysteme behalten die root-Berechtigungen bei.
2. ** Unformatierte (Raw) Datenbankbereiche erhalten die Datenbankbenutzerberechtigungen für die entsprechenden /dev-Dateieinträge (/dev/rxxxx).

Tabelle 13. Datenträgergruppen, logische Datenträger und Dateisysteme für HACMP (leer)

SP-Knoten	Name der Datenträgergruppe	PP-Größe (MB)	Name des logischen Datenträgers	# von PPs	Kopien	hdisk-Liste	Mount-Punkt für Dateisystem	Logischer Datenträger für Dateisystemprotokoll	Knotenbeschreibung und Ausweichknoten	Benutzer/ Eigner der logischen /dev-Einheit

Tabelle 13. Datenträgergruppen, logische Datenträger und Dateisysteme für HACMP (leer) (Forts.)

SP-Knoten	Name der Datenträgergruppe	PP-Größe (MB)	Name des logischen Datenträgers	# von PPs	Kopien	hdisk-Liste	Mount-Punkt für Dateisystem	Logischer Datenträger für Dateisystemprotokoll	Knotenbeschreibung und Ausweichknoten	Benutzer/ Eigner der logischen /dev-Einheit

Tabelle 13. Datenträgergruppen, logische Datenträger und Dateisysteme für HACMP (leer) (Forts.)

SP-Knoten	Name der Datenträgergruppe	PP-Größe (MB)	Name des logischen Datenträgers	# von PPs	Kopien	hdisk-Liste	Mount-Punkt für Dateisystem	Logischer Datenträger für Dateisystemprotokoll	Knotenbeschreibung und Ausweichknoten	Benutzer/ Eigner der logischen /dev-Einheit

Tabelle 14. Planung für HACMP-NFS-Server

SP-Knoten	Externes Dateisystem	Ausweichknoten	IP-Aliaspaare für SP-Switch-Boot- und Serviceadressen	Anzuhängendes Dateisystem (/etc/filesystems)	Als Ausgangsverzeichnis für Datenbank anzugebendes Dateisystem	Adressen zum Exportieren des Dateisystems (/etc/exports)
3	/dbmnt	4	nfs_boot_3 nfs_client_3	nfs_server:/ dbmnt als /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6

Tabelle 14. Planung für HACMP-NFS-Server (Forts.)

SP-Knoten	Externes Dateisystem	Ausweichknoten	IP-Aliaspaare für SP-Switch-Boot- und Serviceadressen	Anzuhängendes Dateisystem (/etc/filesystems)	Als Ausgangsverzeichnis für Datenbank anzugebendes Dateisystem	Adressen zum Exportieren des Dateisystems (/etc/exports)
4	/dbmnt	3	nfs_server_boot nfs_server	nfs_server:/ dbmnt als /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
5	N/V	N/V	nfs_boot_5 nfs_client_5	nfs_server:/ dbmnt als /dbi	/dbi/pwq	N/V
6	N/V	N/V	nfs_boot_6 nfs_client_6	nfs_server:/ dbmnt als /dbi	/dbi/pwq	N/V

Anmerkungen:

1. `/etc/passwd` muss auf allen Knoten identisch sein. Dies kann von der Steuer-Workstation aus synchronisiert werden.
2. Stellen Sie sicher, dass das externe Dateisystem die Berechtigung des Datenbankexemplareigners hat.
3. `/etc/filesystems` muss die Mount-Parameter hard, bg, intr und rw haben.
4. `/etc/exports` hat

   ```
   -root=ip1:ip2:ip3
   ```

   nur auf dem Server und dem zugehörigen Ausweichknoten.

Tabelle 15. Planung für HACMP-NFS-Server (leer)

SP-Knoten	Externes Dateisystem	Ausweichknoten	IP-Aliaspaare für SP-Switch-Boot- und Serviceadressen	Anzuhängendes Dateisystem (/etc/filesystems)	Als Ausgangsverzeichnis für Datenbank anzugebendes Dateisystem	Adressen zum Exportieren des Dateisystems (/etc/exports)

Tabelle 15. Planung für HACMP-NFS-Server (leer) (Forts.)

SP-Knoten	Externes Dateisystem	Ausweichknoten	IP-Aliaspaare für SP-Switch-Boot- und Serviceadressen	Anzuhängendes Dateisystem (/etc/filesystems)	Als Ausgangsverzeichnis für Datenbank anzugebendes Dateisystem	Adressen zum Exportieren des Dateisystems (/etc/exports)

Tabelle 15. Planung für HACMP-NFS-Server (leer) (Forts.)

SP-Knoten	Externes Dateisystem	Ausweichknoten	IP-Aliaspaare für SP-Switch-Boot- und Serviceadressen	Anzuhängendes Dateisystem (/etc/filesystems)	Als Ausgangsverzeichnis für Datenbank anzugebendes Dateisystem	Adressen zum Exportieren des Dateisystems (/etc/exports)

Kapitel 7. Hohe Verfügbarkeit unter dem Windows-Betriebssystem

Sie können Ihr Datenbanksystem so einrichten, dass bei Ausfall einer Maschine der Datenbankserver der ausgefallenen Maschine auf einer anderen Maschine ausgeführt werden kann. Unter Windows NT kann die Funktionsübernahme mit Microsoft Cluster Server (MSCS) implementiert werden. Zur Verwendung von MSCS wird Windows NT Version 4.0 Enterprise Edition mit installierter MSCS-Einrichtung benötigt.

MSCS kann sowohl die Fehlererkennung als auch das Neustarten von Ressourcen in einer Clusterumgebung ausführen, d. h., das Produkt übernimmt z. B. die Unterstützung für die Funktionsübernahme für physische Platten und IP-Adressen. (Wenn die ausgefallene Maschine wieder online ist, werden die Ressourcen nicht automatisch zurückübertragen, sofern sie dazu nicht vorher konfiguriert werden. Weitere Informationen finden Sie in „Überlegungen zur Rückübertragung" auf Seite 272.)

Bevor Sie DB2-Exemplare für die Unterstützung der Funktionsübernahme aktivieren können, müssen Sie die folgenden Planungsschritte ausführen:

1. Legen Sie fest, welche Platten für die Datenspeicherung verwendet werden sollen. Jedem Datenbankserver sollte zumindest eine Platte zur eigenen Verfügung zugeordnet werden. Die Platte, die Sie zum Speichern von Daten einsetzen, muss an ein gemeinsames Plattensubsystem angeschlossen und als MSCS-Plattenressource konfiguriert werden.
2. Stellen Sie sicher, dass Sie eine IP-Adresse für jeden Datenbankserver haben, der ferne Anforderungen unterstützen soll.

Wenn Sie die Unterstützung der Funktionsübernahme einrichten, kann dies für ein vorhandenes Exemplar geschehen, oder Sie können bei der Implementierung der Unterstützung der Funktionsübernahme ein neues Exemplar erstellen.

Zur Aktivierung der Unterstützung der Funktionsübernahme führen Sie die folgenden Schritte aus:

1. Erstellen Sie eine Eingabedatei für das Dienstprogramm DB2MSCS.
2. Rufen Sie den Befehl **db2mscs** auf.
3. Wenn Sie ein partitioniertes Datenbanksystem verwenden, registrieren Sie die Datenbanklaufwerkzuordnung zur Aktivierung der gegenseitigen

Übernahme. Siehe „Registrieren der Datenbanklaufwerkzuordnung für Konfigurationen zur gegenseitigen Übernahme in Umgebungen mit partitionierten Datenbanken" auf Seite 272.

Nach der Aktivierung des Exemplars für die Unterstützung der Funktionsübernahme sieht Ihre Konfiguration ungefähr wie in Abb. 28 aus.

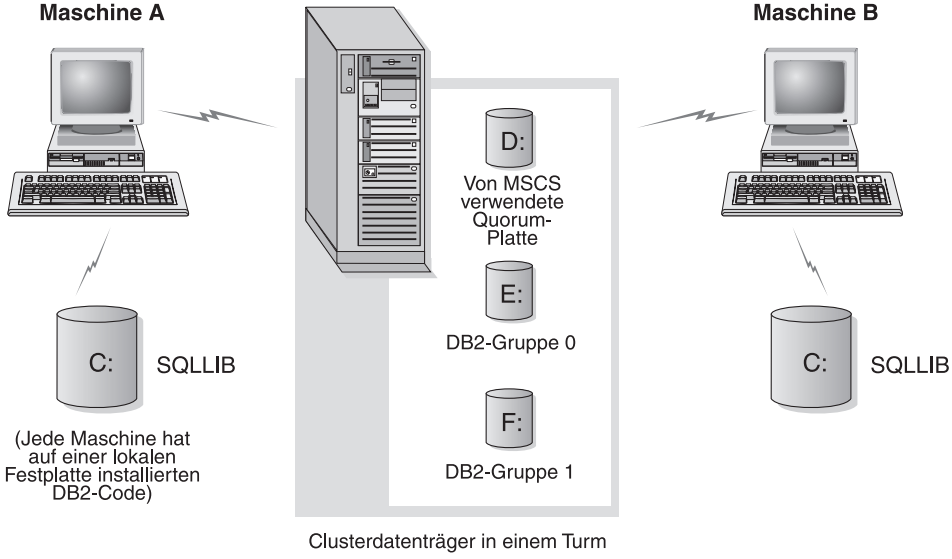


Abbildung 28. Beispiel für eine MSCS-Konfiguration

In den folgenden Abschnitten werden die verschiedenen Arten von Unterstützung für Funktionsübernahmen und ihre Implementierung beschrieben. Bevor Sie einen der im folgenden beschriebenen Schritte ausführen, müssen Sie die MSCS-Software bereits auf jeder Maschine installiert haben, die in einem MSCS-Cluster verwendet werden soll. Darüber hinaus muss auf jeder Maschine DB2 installiert sein.

Konfigurationen für Funktionsübernahme

Zwei Arten von Konfigurationen sind verfügbar:

- Bereitschaftsmodus
- Gegenseitige Übernahme

Zurzeit unterstützt MSCS Cluster mit zwei Maschinen.

In einer Umgebung mit partitionierten Datenbanken müssen nicht alle Cluster die gleiche Konfigurationsart haben. Sie können Cluster haben, die im Bereitschaftsmodus konfiguriert sind, und andere, die für gegenseitige Über-

nahme 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. 29 zeigt ein Beispiel einer Konfiguration für den Bereitschaftsmodus.

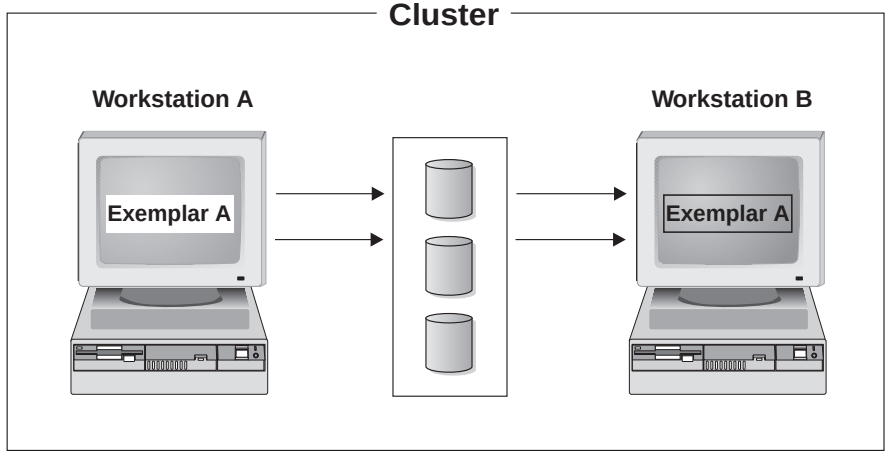


Abbildung 29. Konfiguration für Bereitschaftsmodus

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. 30 auf Seite 262 zeigt ein Beispiel für die Konfiguration zur gegenseitigen Übernahme.

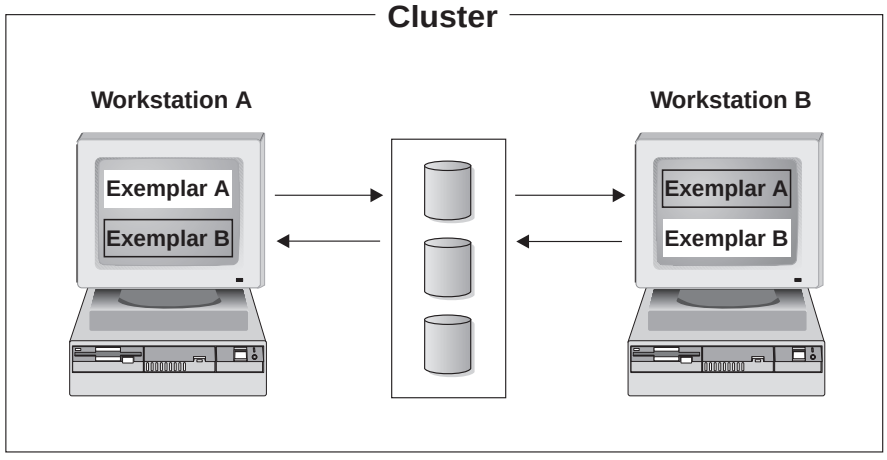


Abbildung 30. Konfiguration zur gegenseitigen Übernahme

Verwenden des Dienstprogramms DB2MSCS

Mit Hilfe des Dienstprogramms DB2MSCS können Sie die Infrastruktur für DB2 zur Unterstützung der Funktionsübernahme in der Windows NT-Umgebung durch die MSCS-Unterstützung (MSCS - Microsoft Cluster Service) erstellen. Sie können dieses Dienstprogramm zur Aktivierung der Funktionsübernahme in Datenbankumgebungen mit einer Partition und mit mehreren Partitionen verwenden.

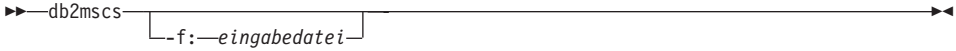
Damit Sie das Dienstprogramm DB2MSCS erfolgreich ausführen können, muss der Cluster Service die Ressourcen-DLL `db2wolf.dll` finden, die im Verzeichnis `%ProgramFiles%\SQLLIB\bin` gespeichert ist. Das Installationsprogramm von DB2 UDB Version 7 setzt die Systemumgebungsvariable PATH auf das Verzeichnis `%ProgramFiles%\SQLLIB\bin`. Wenn Sie das Betriebssystem Windows 2000 einsetzen, müssen Sie die Maschine nach der Installation jedoch nicht erneut starten. Wenn Sie das Dienstprogramm DB2MSCS ausführen möchten, müssen Sie die Maschine erneut starten, damit die Umgebungsvariable PATH für den Cluster Service aktualisiert wird.

Rufen Sie den Befehl **db2mscs** einmal für jedes Exemplar auf der Exemplareignermaschine auf. Wenn es auf einer Maschine im MSCS-Cluster nur ein aktives DB2-Exemplar gibt, richtet **db2mscs** eine Konfiguration im Bereitschaftsmodus ein. Wenn Sie auf jeder Maschine im MSCS-Cluster ein aktives Exemplar haben, führen Sie das Programm DB2MSCS einmal auf jeder Exemplareignermaschine aus, um eine Konfiguration zur gegenseitigen Übernahme einzurichten.

Das Dienstprogramm DB2MSCS führt folgende Schritte aus:

1. Es liest die erforderlichen MSCS- und DB2-Parameter aus einer Eingabedatei namens DB2MSCS.CFG. Informationen zur gesamten Gruppe der Eingabeparameter finden Sie in „Angeben der Datei DB2MSCS.CFG".
2. Es wertet die Parameter in der Eingabedatei aus.
3. Es registriert die DB2-Ressourcenart.
4. Es erstellt die MSCS-Gruppe (bzw. Gruppen) für die MSCS- und DB2-Ressourcen.
5. Es erstellt die IP-Ressource.
6. Es erstellt die Netzwerknamensressource.
7. Es versetzt MSCS-Platten in die Gruppe.
8. Es erstellt die DB2-Ressource (bzw. Ressourcen).
9. Es fügt alle erforderlichen Abhängigkeiten für die DB2-Ressource hinzu.
10. Es ändert das nicht in einem Cluster konfigurierte DB2-Exemplar in ein Clusterexemplar.
11. Es bringt alle Ressourcen online.

Der Befehl hat folgende Syntax:



Dabei gilt Folgendes:

-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.

Angeben der Datei DB2MSCS.CFG

Die Datei DB2MSCS.CFG ist eine ASCII-Textdatei, die Parameter enthält, die vom Dienstprogramm DB2MSCS gelesen werden. Die Eingabeparameter werden jeweils auf getrennten Zeilen in folgendem Format eingegeben: PARAMETER_SCHLÜSSELWORT=*parameterwert*. Beispiel:

```
CLUSTER_NAME=WOLFPACK
GROUP_NAME=DB2-Gruppe
IP_ADDRESS=9.21.22.89
```

Zwei Beispielkonfigurationsdateien befinden sich im Unterverzeichnis /CFG des Verzeichnisses /SQLLIB. Die erste Datei, DB2MSCS.EE, ist ein Beispiel für Datenbankumgebungen mit einer Partition. Die zweite Datei, DB2MSCS.EEE, ist ein Beispiel für Umgebungen mit partitionierten Datenbanken.

Parameter für die Datei DB2MSCS.CFG:

DB2_INSTANCE
Der Name des DB2-Exemplars. Wenn der Exemplarname *nicht* angegeben wird, wird das Standardexemplar (d. h. der Wert der Umgebungsvariablen DB2INSTANCE) verwendet.

Dieser Parameter besitzt einen globalen Gültigkeitsbereich und wird in der Datei DB2MSCS.CFG nur einmal angegeben.

Dieser Parameter ist optional.

Beispiel:

```
DB2_INSTANCE=DB2
```

Das Exemplar muss bereits vorhanden sein. Informationen zur Erstellung von Exemplaren finden Sie im Handbuch *DB2 Enterprise - Extended Edition für Windows Einstieg*.

DB2_LOGON_USERNAME
Der Name des Anmeldekontos für den DB2-Service.

Dieser Parameter besitzt einen globalen Gültigkeitsbereich und wird in der Datei DB2MSCS.CFG nur einmal angegeben.

Dieser Parameter ist nur für Exemplare von DB2 Enterprise - Extended Edition erforderlich.

Beispiel:

```
DB2_LOGON_USERNAME=db2user
```

DB2_LOGON_PASSWORD
Das Kennwort des Anmeldekontos für den DB2-Service. Wenn der Parameter DB2_LOGON_USERNAME angegeben ist, aber der Parameter DB2_LOGON_PASSWORD nicht, fordert das Dienstprogramm DB2MSCS die Eingabe des Kennworts an. Das Kennwort wird bei der Eingabe in die Befehlszeile nicht angezeigt.

Dieser Parameter besitzt einen globalen Gültigkeitsbereich und wird in der Datei DB2MSCS.CFG nur einmal angegeben.

Dieser Parameter ist nur für Exemplare von DB2 Enterprise - Extended Edition erforderlich.

Beispiel:

```
DB2_LOGON_PASSWORD=xxxxxx
```

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.

Geben Sie diesen Parameter einmal pro Cluster an.

Dieser Parameter ist optional. Wenn er nicht angegeben wird, wird der Name des MSCS-Clusters auf der lokalen Maschine verwendet.

Beispiel:

```
CLUSTER_NAME=WOLFPACK
```

GROUP_NAME

Der Name der MSCS-Gruppe. Wenn dieser Parameter angegeben wird, wird eine neue MSCS-Gruppe erstellt, wenn keine vorhanden ist. Wenn die Gruppe bereits vorhanden ist, wird sie als Zielgruppe verwendet. Jede MSCS-Ressource, die nach dieser Zeile erstellt wird, wird in dieser Gruppe erstellt, bis ein anderes Schlüsselwort GROUP_NAME angegeben wird.

Geben Sie diesen Parameter einmal für jede Gruppe an.

Dieser Parameter ist erforderlich.

Beispiel:

```
GROUP_NAME=DB2-Gruppe
```

DB2_NODE

Die Knotennummer des Datenbankpartitionsservers (Knotens), der in die aktuelle MSCS-Gruppe eingefügt werden soll. Wenn mehrere logische Knoten auf derselben Maschine vorhanden sind, benötigt jeder Knoten ein eigenes Schlüsselwort DB2_NODE.

Dieser Parameter wird nach dem Parameter GROUP_NAME angegeben, damit die DB2-Ressourcen in der richtigen MSCS-Gruppe erstellt werden.

Dieser Parameter ist nur für Exemplare von DB2 Enterprise - Extended Edition erforderlich.

Beispiel:

```
DB2_NODE=0
```

IP_NAME

Der Name der IP-Adressressource. Der Wert für IP_NAME kann willkürlich gewählt werden, muss jedoch 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. In Umgebungen mit partitionierten Datenbanken muss dieser Parameter für die Exemplareignermaschine angegeben werden. Dieser Parameter ist in Datenbankumgebungen mit einer Partition optional.

Beispiel:

```
IP_NAME=IP-Adresse für DB2
```

Anmerkung: DB2-Clients sollten die TCP/IP-Adresse dieser IP-Ressource zum Katalogisieren des TCP/IP-Knoteneintrags verwenden. Durch Verwenden der MSCS-IP-Adresse können DB2-Clients auch, nachdem die Funktion des Datenbankservers von der anderen Maschine übernommen wurde, noch eine Verbindung zum Datenbankserver herstellen, weil die IP-Adresse auf der Übernahmemaschine verfügbar ist.

Attribute der IP-Ressource:

IP_ADDRESS
Die TCP/IP-Adresse der IP-Ressource. Geben Sie dieses Schlüsselwort an, um die TCP/IP-Adresse der vorangehenden IP-Ressource zu definieren.

Dieser Parameter ist erforderlich, wenn der Parameter IP_NAME angegeben wird.

Beispiel:

```
IP_ADDRESS=9.21.22.34
```

IP_SUBNET
Die Subnet-Maske für die vorangehende IP-Ressource.

Dieser Parameter ist erforderlich, wenn der Parameter IP_NAME angegeben wird.

Beispiel:

```
IP_SUBNET=255.255.255.0
```

IP_NETWORK
Der Name des MSCS-Netzwerks, zu dem die vorangehende IP-Ressource gehört. Wenn dieser Parameter nicht angegeben wird, wird das erste vom System erkannte MSCS-Netzwerk verwendet.

Dieser Parameter ist optional.

Beispiel:

```
IP_NETWORK=Token Ring
```

NETNAME_NAME
Der Name der Netzwerknamensressource. Geben Sie diesen Parameter an, um die Netzwerknamensressource zu erstellen.

Dieser Parameter ist in Datenbankumgebungen mit einer Partition optional. Für Umgebungen mit partitionierten Datenbanken ist er erforderlich.

Beispiel:

```
NETNAME_NAME=Netzwerkname für DB2
```

Attribute der Netzwerknamensressource:

NETNAME_VALUE
Der Wert für den Netzwerknamen.

Dieser Parameter ist erforderlich, wenn der Parameter NETNAME_NAME angegeben wird.

Beispiel:

```
NETNAME_VALUE=DB2SRV
```

NETNAME_DEPENDENCY
Die Abhängigkeitsliste für die Netzwerknamensressource. Jede Netzwerknamensressource muss eine Abhängigkeit von einer IP-Adressressource haben. Wenn dieser Parameter nicht angegeben wird, besitzt die Netzwerknamensressource eine Abhängigkeit von der ersten IP-Ressource in der Gruppe. Dieser Parameter ist optional.

Beispiel:

```
NETNAME_DEPENDENCY=IP-Adresse für DB2
```

DISK_NAME
Der Name der physischen Plattenressourcen, die in die aktuellen Gruppen versetzt werden sollen. Geben Sie so viele Plattenressourcen wie nötig an.

Anmerkungen:

1. Die Plattenressourcen müssen bereits vorhanden sein.
2. Wenn das Dienstprogramm DB2MSCS das DB2-Exemplar für MSCS-Unterstützung konfiguriert, wird das Exemplarverzeichnis auf die *erste* MSCS-Platte in der Gruppe kopiert. Soll eine andere MSCS-Platte für das Exemplarverzeichnis angegeben werden, ist der Parameter INSTPROF_DISK zu verwenden.

Beispiel:

```
DISK_NAME=Disk E:
DISK_NAME=Disk F:
```

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* MSCS-Platte, die zur gleichen Gruppe wie das Exemplarverzeichnis gehört.

Das DB2-Exemplarverzeichnis wird auf der MSCS-Platte unter dem Verzeichnis X:\DB2PROFS erstellt. (Dabei ist X der Laufwerkbuchstabe der MSCS-Platte.)

Beispiel:

```
INSTPROF_DISK=Disk E:
```

TARGET_DRVMAP_DISK

Ein optionaler Parameter zur Angabe der MSCS-Zielplatte für die Zuordnung von Datenbanklaufwerken. Wenn eine Datenbank auf einer MSCS-Platte erstellt wird, die nicht zur gleichen Gruppe wie der Knoten gehört, wird die Zielplatte der Laufwerkzuordnung zur Aufnahme der Datenbankpartition verwendet. Wenn dieser Parameter nicht angegeben wird, muss die Zuordnung der Datenbanklaufwerke manuell mit Hilfe des Dienstprogramms DB2DRVMP registriert werden.

Beispiel:

```
TARGET_DRVMAP_DISK=Disk E:
```

Einrichten der Funktionsübernahme für ein Datenbanksystem mit einer Partition

Wenn Sie das Dienstprogramm DB2MSCS für ein Datenbanksystem mit einer Partition ausführen, enthält eine MSCS-Gruppe DB2 und sämtliche abhängigen MSCS-Ressourcen (die IP-Adresse, Netzwerkname und Platten). Zum Beispiel kann der Inhalt der Datei DB2MSCS.CFG für ein Datenbanksystem mit einer Partition wie folgt aussehen:

```
#
#  DB2MSCS.CFG für ein Datenbanksystem mit einer Partition
#
DB2_INSTANCE=DB2
CLUSTER_NAME=MSCS
GROUP_NAME=DB2-Gruppe
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

Einrichten einer Konfiguration zur gegenseitigen Übernahme für zwei Datenbanksysteme mit jeweils einer Partition

Sie können zwei Datenbanksysteme mit jeweils einer Partition, jede auf einer separaten Maschine einrichten, so dass ein Datenbanksystem bei einem Ausfall auf einer Maschine auf dem anderen MSCS-Knoten wieder gestartet wird.

Zur Einrichtung der Unterstützung der Funktionsübernahme für diese Konfiguration müssen Sie das Dienstprogramm DB2MSCS einmal auf jeder Exemplareignermaschine ausführen. Die Konfigurationsdatei muss für jedes Datenbanksystem angepasst werden.

Angenommen, die DB2-Exemplare hießen DB2A und DB2B. Die Datei DB2MSCS.CFG für das Exemplar DB2A sähe folgendermaßen aus:

```
#
#  DB2MSCS.CFG für das erste Datenbanksystem mit einer Partition
#
DB2_INSTANCE=DB2A
CLUSTER_NAME=MSCS
GROUP_NAME=DB2A-Gruppe
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

Die Datei DB2MSCS.CFG für das Exemplar DB2B sähe folgendermaßen aus:

```
#
#  DB2MSCS.CFG für das zweite Datenbanksystem mit einer Partition
#
DB2_INSTANCE=DB2B
CLUSTER_NAME=MSCS
GROUP_NAME=DB2B-Gruppe
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk F:
```

Ein vollständiges Beispiel finden Sie in „Beispiel - Einrichten zweier Exemplare mit einer Partition für gegenseitige Übernahme” auf Seite 275.

Einrichten mehrerer MSCS-Cluster für ein partitioniertes Datenbanksystem

Wenn Sie das Dienstprogramm DB2MSCS für ein Datenbanksystem mit mehreren Partitionen ausführen, wird eine MSCS-Gruppe für jede physische Maschine erstellt, die Teil des Systems ist. Die Datei DB2MSCS.CFG muss mehrere Abschnitte enthalten, und jeder Abschnitt muss einen anderen Wert für den Parameter GROUP_NAME und andere Werte für alle erforderlichen abhängigen Ressourcen für diese Gruppe enthalten.

Außerdem müssen Sie den Parameter DB2_NODE für jeden Datenbankpartitionsserver in jeder MSCS-Gruppe angeben. Wenn Sie mehrere logische Knoten haben, ist für jeden logischen Knoten ein eigenes Schlüsselwort DB2_NODE erforderlich.

Nehmen Sie beispielsweise an, dass Sie ein Datenbanksystem mit mehreren Partitionen haben, das aus vier Datenbankpartitionsservern auf vier Maschinen besteht, und dass Sie zwei MSCS-Cluster zur gegenseitigen Übernahme konfigurieren wollen. In diesem Fall würden Sie die Konfigurationsdatei DB2MSCS.CFG wie folgt einrichten:

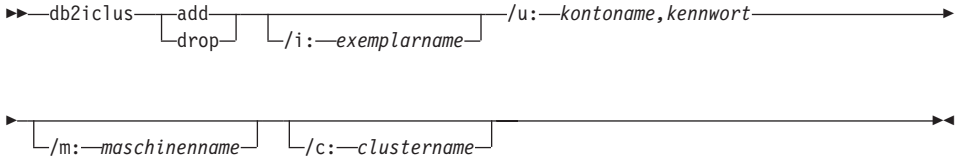
```
#
#  DB2MSCS.CFG für ein partitioniertes Datenbanksystem mit
#  mehreren Clustern
DB2_INSTANCE=DB2MPP
DB2_LOGON_USERNAME=db2user
DB2_LOGON_PASSWORD=xxxxxx
CLUSTER_NAME=MSCS1
# Gruppe 1
GROUP_NAME=DB2-Gruppe 1
DB2_NODE=0
IP_NAME=...

...
# Gruppe 2
GROUP_NAME=DB2-Gruppe 2
DB2_NODE=1
IP_NAME=...

...

CLUSTER_NAME=MSCS2
# Gruppe 3
GROUP_NAME=DB2-Gruppe 3
DB2_NODE=2
IP_NAME=...

...
# Gruppe 4
GROUP_NAME=DB2-Gruppe 4
DB2_NODE=3
IP_NAME=...

...
```

Ein vollständiges Beispiel finden Sie in „Beispiel - Einrichten eines in vier Knoten partitionierten Datenbanksystems zur gegenseitigen Übernahme” auf Seite 278.

Pflegen des MSCS-Systems

Bei der Ausführung des Dienstprogramms DB2MSCS wird die Infrastruktur für die Unterstützung der Funktionsübernahme für alle Maschinen im MSCS-Cluster erstellt. Zum Entfernen dieser Unterstützung von einer Maschine verwenden Sie den Befehl **db2iclus** mit der Option ″drop″. Soll die Unterstützung für eine Maschine wieder aktiviert werden, verwenden Sie die Option ″add″.

Der Befehl hat folgende Syntax:



Dabei gilt Folgendes:

add
Aktiviert die Unterstützung der Funktionsübernahme auf der Maschine, indem sie einem MSCS-Cluster hinzugefügt wird. Die DB2-Ressource (Datenbankserver) kann dann von dieser Maschine übernommen werden.

drop
Entfernt die Unterstützung der Funktionsübernahme von der Maschine, indem die Maschine aus einem MSCS-Cluster gelöscht wird.

/i: *exemplarname*
Der Name des Exemplars. (Dieser Parameter überschreibt die Einstellung der Umgebungsvariablen DB2INSTANCE.)

/u: *kontoname*, *kennwort*
Das Domänenkonto, das als Name des Anmeldekontos des DB2-Services verwendet wird. Beispiel:

```
/u:domäneA\db2nt,kennwort
```

Dieser Parameter ist nur im Zusammenhang mit der Option ″add″ erforderlich.

/m:*maschinenname*
Der Computername der Maschine, die einem MSCS-Cluster hinzugefügt oder aus ihm gelöscht werden soll. Sie müssen diese Option angeben, wenn Sie den Befehl von einer ande-

ren Maschine aus ausführen als der, für die Sie die Funktionsübernahmeunterstützung ändern.

/c: *clustername* — Der Name des MSCS-Clusters, wie er im LAN registriert ist. Dieser Name wird bei der Ersterstellung des MSCS-Clusters angegeben.

Überlegungen zur Rückübertragung

Standardmäßig sind Gruppen so eingerichtet, dass sie nicht auf die ursprüngliche (ausgefallene) Maschine zurückübertragen werden. Sofern Sie eine DB2-Gruppe nicht manuell für eine Rückübertragung (Fallback) nach einer Funktionsübernahme konfigurieren, wird sie auch nach Beseitigung der Ursache für die Funktionsübernahme weiterhin auf dem alternativen MSCS-Knoten ausgeführt.

Wenn Sie eine DB2-Gruppe zur automatischen Rückübertragung auf die ursprüngliche Maschine konfigurieren, werden alle Ressourcen in der DB2-Gruppe, einschließlich der DB2-Ressource, sofort nach erneuter Verfügbarkeit der ursprünglichen Maschine auf diese zurückübertragen. Wenn während der Rückübertragung eine Datenbankverbindung aktiv ist, kann die DB2-Ressource nicht offline genommen werden und der Rückübertragungsprozess schlägt fehl.

Wenn Sie die Beendigung aller Datenbankverbindungen während der Rückübertragung erzwingen wollen, setzen Sie den Wert der Registrierungsvariablen DB2_FALLBACK auf ON. Diese Variable muss folgendermaßen gesetzt werden:

```
db2set DB2_FALLBACK=ON
```

Nach dem Setzen dieser Registrierungsvariablen braucht das System nicht erneut gebootet und der Clusterservice nicht erneut gestartet zu werden.

Registrieren der Datenbanklaufwerkzuordnung für Konfigurationen zur gegenseitigen Übernahme in Umgebungen mit partitionierten Datenbanken

Bei der Erstellung einer Datenbank in einer Umgebung mit partitionierten Datenbanken können Sie einen Laufwerkbuchstaben angeben, um festzulegen, wo die Datenbank zu erstellen ist.

Anmerkung: Für Datenbankumgebungen mit einer Partition wird keine Datenbanklaufwerkzuordnung definiert.

Wenn der Befehl CREATE DATABASE ausgeführt wird, geht er davon aus, dass das Laufwerk, das Sie angeben, gleichzeitig für alle Maschinen verfügbar

ist, die an dem Exemplar beteiligt sind. Da dies nicht möglich ist, verwendet DB2 eine Datenbanklaufwerkzuordnung, um demselben Laufwerk für jede Maschine einen anderen Namen zuzuweisen.

Nehmen Sie beispielsweise an, dass ein DB2-Exemplar namens DB2 zwei Datenbankpartitionsserver enthält:

```
NODE0 ist auf Maschine WOLF_NODE_0 aktiv
NODE1 ist auf Maschine WOLF_NODE_1 aktiv
```

Nehmen Sie außerdem an, dass die freigegebene Platte E: zur selben Gruppe wie NODE0 und die freigegebene Platte F: zur selben Gruppe wie NODE1 gehören.

Der Befehl zur Erstellung einer Datenbank auf der freigegebenen Platte E:

```
db2 create database mppdb on e:
```

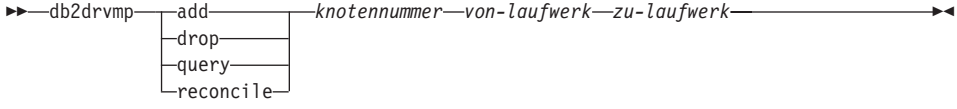
Damit der Befehl erfolgreich ausgeführt werden kann, muss Laufwerk E: auf beiden Maschinen verfügbar sein. In einer Konfiguration zur gegenseitigen Übernahme kann jeder Datenbankpartitionsserver auf einer anderen Maschine aktiv und die Clusterplatte E: nur für eine Maschine verfügbar sein. In einem solchen Fall schlägt der Befehl CREATE DATABASE immer fehl.

Zur Beseitigung dieses Problems sollte das Datenbanklaufwerk wie folgt zugeordnet werden:

```
Für NODE0 geht die Zuordnung von Laufwerk F: zu Laufwerk E:
Für NODE1 geht die Zuordnung von Laufwerk E: zu Laufwerk F:
```

Jeder Datenbankzugriff für NODE0 auf Laufwerk F: wird dann Laufwerk E: zugeordnet, und jeder Datenbankzugriff für NODE1 auf Laufwerk E: wird Laufwerk F: zugeordnet. Bei Verwendung der Laufwerkzuordnung erstellt der Befehl CREATE DATABASE Datenbankdateien auf Laufwerk E: für NODE0 und auf Laufwerk F: für NODE1.

Die Laufwerkzuordnung wird mit Hilfe des Befehls **db2drvmp** eingerichtet. Der Befehl hat folgende Syntax:



Der Befehl hat die folgenden Parameter:

add Fügt eine neue Datenbanklaufwerkzuordnung hinzu.

drop Entfernt eine vorhandene Datenbanklaufwerkzuordnung.

query Fragt eine Datenbankzuordnung ab.

reconcile Repariert eine Datenbanklaufwerkzuordnung, wenn der Inhalt der Registrierdatenbank beschädigt ist. Weitere Informationen finden Sie in „Abstimmen der Datenbanklaufwerkzuordnung".

knotennummer Die Knotennummer. Dieser Parameter ist für add- und drop-Operationen erforderlich.

von-laufwerk Der Laufwerkbuchstabe, der zugeordnet wird. Dieser Parameter ist für add- und drop-Operationen erforderlich.

zu-laufwerk Der Laufwerkbuchstabe, dem zugeordnet wird. Dieser Parameter ist für add-Operationen erforderlich. Für andere Operationen ist er nicht gültig.

Zum Beispiel würde eine Datenbanklaufwerkzuordnung von F: zu E: für NODE0 durch folgenden Befehl definiert:

```
db2drvmp add 0 F E
```

Anmerkung: Die Datenbanklaufwerkzuordnung bezieht sich nicht auf Tabellenbereiche, Behälter oder andere Datenbankspeicherobjekte.

Mit dem folgenden, ganz ähnlichen Befehl würde die Datenbanklaufwerkzuordnung von Laufwerk E: zu Laufwerk F: für NODE1 definiert:

```
db2drvmp add 1 E F
```

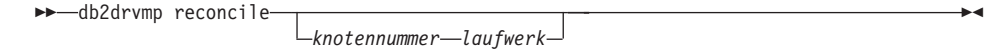
Anmerkung: Definitionen oder Änderungen der Datenbanklaufwerkzuordnung werden nicht sofort wirksam. Zur Aktivierung der Datenbanklaufwerkzuordnung muss die DB2-Ressource mit Hilfe des Clusterverwaltungsprogramms (Cluster Administrator) offline genommen und wieder online gebracht werden.

Mit Hilfe des Schlüsselworts TARGET_DRVMAP_DISK in der Datei `DB2MSCS.CFG` wird die Laufwerkzuordnung automatisch aktiviert.

Abstimmen der Datenbanklaufwerkzuordnung

Wenn eine Datenbank auf einer Maschine erstellt wird, für die eine Datenbanklaufwerkzuordnung in Kraft ist, wird die Zuordnung auf dem Laufwerk in einer verdeckten Datei gesichert. Dadurch wird verhindert, dass das Datenbanklaufwerk nach der Erstellung der Datenbank entfernt wird. Ein Fall, in dem ein *Abstimmen* der Datenbanklaufwerkzuordnung wünschenswert ist, liegt zum Beispiel vor, wenn die Datenbanklaufwerkzuordnung versehentlich gelöscht wurde. Zum Abstimmen der Zuordnung muss der Befehl

db2drvmp reconcile für jeden Datenbankpartitionsserver ausgeführt werden, der die Datenbank enthält. Der Befehl hat folgende Syntax:



Der Befehl hat die folgenden Parameter:

knotennummer Die Knotennummer des zu reparierenden Knotens. Wenn *knotennummer* nicht angegeben wird, stimmt der Befehl die Zuordnung für alle Knoten ab.

laufwerk Das abzustimmende Laufwerk. Wenn kein Laufwerk angegeben wird, stimmt der Befehl die Zuordnung für alle Laufwerke ab.

Der Befehl **db2drvmp** durchsucht alle Laufwerke auf der Maschine nach Datenbankpartitionen, die von dem Datenbankpartitionsserver verwaltet werden, und wendet die Datenbanklaufwerkzuordnung erneut in der erforderlichen Weise auf die Registrierdatenbank an.

Beispiel - Einrichten zweier Exemplare mit einer Partition für gegenseitige Übernahme

Das Ziel in diesem Beispiel besteht in der Einrichtung zweier Datenbankexemplare mit jeweils einer Partition in einer Konfiguration zur gegenseitigen Übernahme. In diesem Beispiel werden vier Server in zwei MSCS-Clustern konfiguriert. Durch die Konfiguration zur gegenseitigen Übernahme wird bei Ausfall einer der Maschinen der für diese Maschine konfigurierte Datenbankserver von der alternativen Maschine übernommen, wie es mit Hilfe der MSCS-Software konfiguriert wurde, und auf der alternativen Maschine ausgeführt.

In der entstehenden Konfiguration gibt es zwei MSCS-Cluster. Jeder Cluster verfügt über:

- Zwei Server, jeder mit 64 MB Speicher und einer lokalen SCSI-Platte von 2 GB
- Einen SCSI-Plattenturm, der drei gemeinsame, jeweils 2 GB große SCSI-Platten enthält

Darüber hinaus ist in jeder Maschine eine 100X-Ethernet-Adapterkarte installiert.

Auf jeder Maschine ist folgende Software installiert:

- Windows NT Version 4.0 Enterprise Edition mit installierter MSCS-Einrichtung
- DB2 Universal Database Enterprise Edition Version 7

Es ergibt sich folgende Netzwerkkonfiguration:

Server 1:	Server 2:
• Maschinenname: db2test1	• Maschinenname: db2test2
• TCP/IP-Host-Name: db2test1	• TCP/IP-Host-Name: db2test2
• IP-Adresse: 9.9.9.1 (Subnet-Maske: 255.255.255.0	• IP-Adresse: 9.9.9.2 (Subnet-Maske: 255.255.255.0
• MSCS-Clustername: ClusterA	• MSCS-Clustername: ClusterA

Beide Maschinen im Netzwerk sind mit TCP/IP konfiguriert und über einen Ethernet-100-T-base-Hub mit einem privaten LAN verbunden. Bei Abwesenheit eines DNS-Servers haben alle Maschinen eine lokale TCP/IP-Datei `hosts`, die folgende Einträge enthält:

```
9.9.9.1 db2test1   # für Server 1
9.9.9.2 db2test2   # für Server 2
9.9.9.3 ClusterA   # für MSCS-ClusterA
9.9.9.4 db2tcp1    # für ferne DB2-Client-Verbindung zu Server 1
9.9.9.5 db2tcp2    # für ferne DB2-Client-Verbindung zu Server 2
```

Vorbereitung

Bevor Sie die folgenden Tasks durchführen, wird angenommen, dass beide Maschinen zur gleichen Domäne namens `DB2NTD` gehören:

1. Erstellen Sie ein Domänenkonto für DB2, dass zur lokalen Administratorgruppe der Maschinen gehört, auf denen DB2 aktiv sein wird. Verwenden Sie das Konto zur Durchführung aller Tasks:
   - Setzen Sie den Benutzernamen auf `db2nt`.
   - Setzen Sie das Kennwort auf `db2nt`.
2. Installieren Sie die MSCS-Einrichtung auf den Maschinen `db2test1` und `db2test2`:
   - Nennen Sie den MSCS-Cluster `ClusterA`.
   - Die IP-Adresse des Clusters ist `9.9.9.3`.
   - Die freigegebene Platte D: wird von der MSCS-Software verwendet.
   - Die freigegebenen Platten E: und F: werden von DB2 verwendet.
3. Installieren Sie DB2 Universal Database Enterprise Edition Version 7 auf Maschine `db2test1`. Installieren Sie die Software auf dem lokalen Laufwerk unter `C:\SQLLIB`.

4. Installieren Sie DB2 Universal Database Enterprise Edition Version 7 auf Maschine db2test2. Installieren Sie die Software auf dem lokalen Laufwerk unter C:\SQLLIB.

Im nächsten Schritt wird die Datei DB2MSCS.CFG für jedes Exemplar eingerichtet und das Dienstprogramm DB2MSCS für jedes Exemplar ausgeführt.

Ausführen des Dienstprogramms DB2MSCS

Zum Einrichten der Maschine db2test1 führen Sie folgende Schritte aus:

1. Melden Sie sich auf Maschine db2test1 als Benutzer db2nt an. Das Kennwort ist db2nt.
2. Erstellen Sie das DB2-Exemplar DB2A, wenn es nicht bereits vorhanden ist. Der Befehl zur Erstellung des Exemplars lautet:

   ```
   db2icrt DB2A
   ```

3. Richten Sie die Datei DB2MSCS.CFG für das DB2-Exemplar auf Maschine db2test1 ein:

   ```
   #
   #  DB2MSCS.CFG für Datenbanksystem
   #  auf Maschine db2test1
   DB2_INSTANCE=DB2A
   CLUSTER_NAME=ClusterA
   #
   # Gruppe 1
   GROUP_NAME=DB2A-Gruppe
   IP_NAME=IP-Adresse für DB2A
   IP_ADDRESS=9.9.9.4
   IP_SUBNET=255.255.255.0
   IP_NETWORK=ClusterA
   NETNAME_NAME=Netzwerkname für DB2A
   NETNAME_VALUE=DB2SRV1
   NETNAME_DEPENDENCY=IP-Adresse für DB2A
   DISK_NAME=Disk E:
   INSTPROF_DISK=Disk E:
   ```

4. Führen Sie das Dienstprogramm DB2MSCS wie folgt aus:

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

5. Melden Sie sich vom Konto db2nt ab.
6. Melden Sie sich auf Maschine db2test2 als Benutzer db2nt an, der zur lokalen Administratorgruppe gehört. Das Kennwort ist db2nt.
7. Erstellen Sie das DB2-Exemplar DB2B, wenn es nicht bereits vorhanden ist. Der Befehl zur Erstellung des Exemplars lautet:

   ```
   db2icrt DB2B
   ```

8. Richten Sie die Datei DB2MSCS.CFG für das DB2-Exemplar auf Maschine db2test2 ein:

   ```
   #
   #  DB2MSCS.CFG für Datenbanksystem
   #  auf Maschine db2test2
   DB2_INSTANCE=DB2B
   ```

```
CLUSTER_NAME=ClusterA
#
# Gruppe 1
GROUP_NAME=DB2B-Gruppe
IP_NAME=IP-Adresse für DB2B
IP_ADDRESS=9.9.9.5
IP_SUBNET=255.255.255.0
IP_NETWORK=ClusterA
NETNAME_NAME=Netzwerkname für DB2B
NETNAME_VALUE=DB2SRV2
NETNAME_DEPENDENCY=IP-Adresse für DB2B
DISK_NAME=Disk F:
INSTPROF_DISK=Disk F:
```

9. Führen Sie das Dienstprogramm DB2MSCS wie folgt aus:

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

10. Melden Sie sich vom Konto db2nt ab.

Beispiel - Einrichten eines in vier Knoten partitionierten Datenbanksystems zur gegenseitigen Übernahme

Das Ziel in diesem Beispiel besteht in der Einrichtung eines in vier Knoten partitionierten Datenbanksystems in einer Konfiguration zur gegenseitigen Übernahme. In diesem Beispiel werden vier Server in zwei MSCS-Clustern konfiguriert. Durch die Konfiguration zur gegenseitigen Übernahme werden bei Ausfall einer der Maschinen die für diese Maschine konfigurierten Datenbankpartitionsserver von der alternativen Maschine übernommen, wie es mit Hilfe der MSCS-Software konfiguriert wurde, und als ein logischer Knoten auf der alternativen Maschine ausgeführt.

In der entstehenden Konfiguration gibt es zwei MSCS-Cluster. Jeder Cluster verfügt über:

- Zwei Server, jeder mit 64 MB Speicher und einer lokalen SCSI-Platte von 2 GB
- Einen SCSI-Plattenturm, der drei gemeinsame, jeweils 2 GB große SCSI-Platten enthält

Darüber hinaus ist in jeder Maschine eine 100X-Ethernet-Adapterkarte installiert.

Auf jeder Maschine ist folgende Software installiert:

- Windows NT Version 4.0 Enterprise Edition mit installierter MSCS-Einrichtung
- DB2 Universal Database Extended Enterprise Edition Version 7

Beispiel für für gegenseitige Übernahme (partitioniertes Datenbanksystem)

Es ergibt sich folgende Netzwerkkonfiguration:

Server 1: • Maschinenname: db2test1 • TCP/IP-Host-Name: db2test1 • IP-Adresse: 9.9.9.1 (Subnet-Maske: 255.255.255.0 • MSCS-Clustername: ClusterA	Server 2: • Maschinenname: db2test2 • TCP/IP-Host-Name: db2test2 • IP-Adresse: 9.9.9.2 (Subnet-Maske: 255.255.255.0 • MSCS-Clustername: ClusterA
Server 3: • Maschinenname: db2test3 • TCP/IP-Host-Name: db2test3 • IP-Adresse: 9.9.9.3 (Subnet-Maske: 255.255.255.0 • MSCS-Clustername: ClusterB	Server 4: • Maschinenname: db2test4 • TCP/IP-Host-Name: db2test4 • IP-Adresse: 9.9.9.4 (Subnet-Maske: 255.255.255.0 • MSCS-Clustername: ClusterB

Alle Maschinen im Netzwerk sind mit TCP/IP konfiguriert und über einen Ethernet-100-T-base-Hub mit einem privaten LAN verbunden. Bei Abwesenheit eines DNS-Servers haben alle Maschinen eine lokale TCP/IP-Datei `hosts`, die folgende Einträge enthält:

```
9.9.9.1 db2test1   # für Server 1
9.9.9.2 db2test2   # für Server 2
9.9.9.3 db2test3   # für Server 3
9.9.9.4 db2test4   # für Server 4
9.9.9.5 ClusterA   # für MSCS-Cluster 1
9.9.9.6 ClusterB   # für MSCS-Cluster 2
9.9.9.7 db2tcp     # für ferne DB2-Client-Verbindung
```

Vorbereitung

Bevor Sie die folgenden Tasks durchführen, wird angenommen, dass alle vier Maschinen zur gleichen Domäne namens `DB2NTD` gehören:

1. Erstellen Sie ein Domänenkonto für DB2, dass zur lokalen Administratorgruppe der Maschinen gehört, auf denen DB2 aktiv sein wird. Verwenden Sie das Konto zur Durchführung aller Tasks:
   - Setzen Sie den Benutzernamen auf `db2nt`.
   - Setzen Sie das Kennwort auf `db2nt`.
2. Erstellen Sie ein zweites Domänenkonto mit dem Merkmal, dass das Kennwort nie abläuft. Dieses Konto wird den DB2-Services zugeordnet:
   - Setzen Sie den Benutzernamen auf `db2mpp`.
   - Setzen Sie das Kennwort auf `db2mpp`.
3. Installieren Sie die MSCS-Einrichtung auf den Maschinen `db2test1` und `db2test2`:
   - Nennen Sie den MSCS-Cluster `ClusterA`.

- Die IP-Adresse des Clusters ist 9.9.9.5.
- Die freigegebene Platte D: wird von der MSCS-Software verwendet.
- Die freigegebenen Platten E: und F: werden von DB2 verwendet.

4. Installieren Sie die MSCS-Einrichtung auf den Maschinen db2test3 und db2test4:
   - Nennen Sie den MSCS-Cluster ClusterB.
   - Die IP-Adresse des Clusters ist 9.9.9.6.
   - Die freigegebene Platte D: wird von der MSCS-Software verwendet.
   - Die freigegebenen Platten E: und F: werden von DB2 verwendet.
5. Installieren Sie DB2 Enterprise - Extended Edition auf Maschine db2test1:
   - Wählen Sie die Option *"Diese Maschine wird zum Datenbankpartitionsserver, der Eigner des Exemplars ist"* aus.
   - Das Konto für den DB2-Service ist db2mpp. Das Kennwort ist db2mpp.
   - Installieren Sie die Software auf dem lokalen Laufwerk unter C:\SQLLIB.
6. Installieren Sie DB2 Enterprise - Extended Edition auf den Maschinen db2test2, db2test3 und db2test4:
   - Wählen Sie die Option *"Diese Maschine wird ein neuer Knoten auf einem vorhandenen partitionierten Datenbanksystem"* aus.
   - Wählen Sie db2test1 als die Exemplareignermaschine aus.
   - Das Konto für den DB2-Service ist db2mpp. Das Kennwort ist db2mpp.
   - Installieren Sie die Software auf dem lokalen Laufwerk unter C:\SQLLIB.

Im nächsten Schritt wird die Datei DB2MSCS.CFG eingerichtet und das Dienstprogramm DB2MSCS ausgeführt.

Ausführen des Dienstprogramms DB2MSCS

Zum Einrichten der Maschine db2test1 führen Sie folgende Tasks aus:

1. Melden Sie sich als Benutzer db2nt an, der zur lokalen Administratorgruppe gehört. Das Kennwort ist db2nt.
2. Richten Sie die Datei DB2MSCS.CFG ein:

```
#
#  DB2MSCS.CFG für ein partitioniertes Datenbanksystem mit
#  mehreren MSCS-Clustern
DB2_INSTANCE=DB2MPP
CLUSTER_NAME=ClusterA
DB2_LOGON_USERNAME=db2mpp
DB2_LOGON_PASSWORD=db2mpp
# Gruppe 1
# für DB2 Knoten 0
GROUP_NAME=DB2NODE0
DB2_NODE=0
IP_NAME=IP-Adresse für DB2
IP_ADDRESS=9.9.9.7
IP_SUBNET=255.255.255.0
```

```
IP_NETWORK=Ethernet
NETNAME_NAME=Netzwerkname für DB2
NETNAME_VALUE=DB2WOLF
NETNAME_DEPENDENCY=IP-Adresse für DB2
DISK_NAME=Disk E:
INSTPROF_DISK=Disk E:
#
# Gruppe 2
# für DB2 Knoten 1
GROUP_NAME=DB2NODE1
DB2_NODE=1
DISK_NAME=Disk F:
#
CLUSTER_NAME=ClusterB
# Gruppe 3
# für DB2 Knoten 2
GROUP_NAME=DB2NODE2
DB2_NODE=2
DISK_NAME=Disk E:
#
# Gruppe 4
# für DB2 Knoten 3
GROUP_NAME=DB2NODE3
DB2_NODE=3
DISK_NAME=Disk F:
```

3. Führen Sie das Dienstprogramm DB2MSCS wie folgt aus:

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

4. Melden Sie sich vom Konto db2nt ab.

Der letzte Schritt ist das Registrieren der Datenbanklaufwerkzuordnung für die beiden MSCS-Cluster.

Registrieren der Datenbanklaufwerkzuordnung für ClusterA

Zum Registrieren der Datenbanklaufwerkzuordnung für MSCS-Cluster ClusterA sind folgende Tasks auszuführen:

1. Melden Sie sich auf Maschine db2test1 als Benutzer db2mpp an, d. h. mit dem den DB2-Services zugeordneten Konto. Das Kennwort ist db2mpp.
2. Geben Sie zum Registrieren der Datenbanklaufwerkzuordnung die folgenden Befehle ein:

   ```
   db2drvmp add 0 F E

   db2drvmp add 1 E F
   ```

3. Nehmen Sie alle DB2-Ressourcen offline, und bringen Sie sie dann wieder online.

Registrieren der Datenbanklaufwerkzuordnung für ClusterB

Zum Registrieren der Datenbanklaufwerkzuordnung für MSCS-Cluster ClusterB sind folgende Schritte auszuführen:

Beispiel für für gegenseitige Übernahme (partitioniertes Datenbanksystem)

1. Melden Sie sich auf Maschine `db2test3` als Benutzer `db2mpp` an, d. h. mit dem den DB2-Services zugeordneten Konto. Das Kennwort ist `db2mpp`.
2. Geben Sie zum Registrieren der Datenbanklaufwerkzuordnung die folgenden Befehle ein:

   ```
   db2drvmp add 2 F E
   db2drvmp add 3 E F
   ```

3. Nehmen Sie alle DB2-Ressourcen offline, und bringen Sie sie dann wieder online.

Verwalten von DB2 in einer MSCS-Umgebung

Der Einsatz von MSCS-Clustern bringt für das DB2-Exemplar zusätzliche Planung im Hinblick auf den täglichen Betrieb, den Datenbankeinsatz und die Datenbankkonfiguration mit sich. Damit DB2 auf jedem MSCS-Knoten transparent ausgeführt werden kann, sind zusätzliche administrative Tasks erforderlich. Alle von DB2 abhängigen Betriebssystemressourcen müssen auf allen MSCS-Knoten verfügbar sein. Einige dieser Betriebssystemressourcen gehören nicht zum Bereich von MSCS. Das heißt, sie können nicht als eine MSCS-Ressource definiert werden. Sie müssen daher sicherstellen, dass jedes System so konfiguriert ist, dass die gleichen Betriebssystemressourcen auf allen MSCS-Knoten zur Verfügung stehen. In den folgenden Abschnitten werden die zusätzlichen Maßnahmen beschrieben, die durchgeführt werden müssen.

Starten und Stoppen von DB2-Ressourcen

Sie müssen DB2-Ressourcen vom Clusterverwaltungsprogramm (Cluster Administrator) aus starten und stoppen. Es sind verschiedene Möglichkeiten verfügbar, ein DB2-Exemplar zu starten, wie zum Beispiel der Befehl **db2start** und die Option **Dienste** in der Systemsteuerung. Wird DB2 jedoch nicht über Cluster Administrator gestartet, erkennt die MSCS-Software den Status des DB2-Exemplars nicht. Wenn ein DB2-Exemplar über Cluster Administrator gestartet und mit dem Befehl **db2stop** gestoppt wird, interpretiert die MSCS-Software den Befehl **db2stop** als Softwarefehler und versucht, DB2 erneut zu starten. (Die aktuellen MSCS-Schnittstellen ermöglichen keine Benachrichtigung über einen *Ressourcenstatus*.)

Analog gilt, dass, wenn Sie ein DB2-Exemplar mit dem Befehl **db2start** starten, MSCS nicht erkennen kann, dass die Ressource online ist. Wenn ein Datenbankserver ausfällt, bringt MSCS die DB2-Ressource in diesem Fall nicht auf der Übernahmemaschine im Cluster wieder online.

Drei Operationen können auf ein DB2-Exemplar angewendet werden:

Online

Diese Operation ist äquivalent zur Verwendung des Befehls **db2start**. Wenn DB2 bereits aktiv ist, kann diese Operation einfach dazu verwendet werden, MSCS darüber zu informieren, dass DB2 aktiv ist.

Fehler während der Ausführung dieser Operation werden in das Windows NT-Ereignisprotokoll geschrieben.

Offline
Diese Operation ist äquivalent zur Verwendung des Befehls **db2stop**. Wenn es aktive Verbindungen zu einem Exemplar gibt, schlägt diese Operation fehl. Dies stimmt mit der Funktionsweise des Befehls **db2stop** überein.

Ressource abbrechen (Fail Resource)
Diese Operation ist äquivalent zur Verwendung des Befehls **db2stop** unter Angabe der Option **force**. DB2 trennt alle Anwendungen vom DB2-System und stoppt alle Datenbankserver.

Ausführen von Scripts

Sie können Scripts ausführen, bevor eine DB2-Ressource online gebracht wird und nachdem sie online gebracht wurde. Diese Scripts *müssen* sich im Exemplarprofilverzeichnis befinden, das in der Umgebungsvariablen DB2INSTPROF angegeben ist. Dieses Verzeichnis ist der Verzeichnispfad, der durch den Parameter "-p" des Befehls **db2icrt** angegeben wird. Sie können diesen Wert durch folgenden Befehl abrufen:

```
db2set -i:exemplarname DB2INSTPROF
```

Dieser Dateipfad muss auf einer Platte im Cluster sein, so dass das Exemplarverzeichnis auf allen Clusterknoten verfügbar ist.

Diese Script-Dateien sind nicht erforderlich und werden nur ausgeführt, wenn Sie im Exemplarverzeichnis gefunden werden. Sie werden vom MSCS-Cluster-Service im Hintergrund gestartet. Die Script-Dateien müssen Standardausgaben umleiten, um alle Ausgaben als Ergebnis von Befehlen innerhalb der Script-Dateien aufzufangen. Die Ausgaben werden nicht auf dem Bildschirm angezeigt.

In einer Umgebung mit partitionierten Datenbanken wird standardmäßig das gleiche Script von jedem Datenbankpartitionsserver im Exemplar verwendet. Wenn Sie zwischen verschiedenen Datenbankpartitionsservern im Exemplar unterscheiden müssen, ordnen Sie der Umgebungsvariablen DB2NODE unterschiedliche Werte zu, um bestimmte Zielknotennummern anzugeben. (Verwenden Sie dazu zum Beispiel die Anweisung IF in den Dateien `db2cpre.bat` und `db2cpost.bat`.)

Ausführen von Scripts, bevor DB2-Ressourcen online gebracht werden

Wenn Sie ein Script ausführen wollen, *bevor* Sie eine DB2-Ressource online bringen, *muss* das Script den Namen `db2cpre.bat` erhalten. DB2 ruft Funktionen auf, die diese Stapeldatei über den Befehlszeilenprozessor von Windows NT starten und auf die Beendigung der Ausführung durch den Befehlszeilenprozessor warten, bevor die DB2-Ressource online gebracht wird. Sie können

diese Stapeldatei für Operationen wie das Ändern der DB2-Datenbankmanagerkonfiguration verwenden. Es ist zum Beispiel sinnvoll, einige Parameterwerte des Datenbankmanagers zu ändern, wenn das Übernahmesystem begrenzte Kapazitäten hat und die Systemressourcen, die von DB2 beansprucht werden, verringert werden müssen.

Die Befehle, die in das Script `db2cpre.bat` gestellt werden, sollten synchron ausgeführt werden. Ansonsten könnte die DB2-Ressource online gebracht werden, bevor alle Operationen im Script beendet sind, wodurch sich unvorhergesehene Resultate ergeben können. Insbesondere sollte der Befehl **db2cmd** nicht im Script `db2cpre.bat` ausgeführt werden, weil er seinerseits einen anderen Befehlsprozessor startet, der DB2-Befehle asynchron zum Programm **db2cmd** ausführt.

Wenn Sie DB2-CLP-Befehle im Script `db2cpre.bat` ausführen wollen, sollten die Befehle in eine Datei gestellt und als CLP-Stapeldatei von einem Programm aus ausgeführt werden, das die DB2-Umgebung für den DB2-Befehlszeilenprozessor initialisiert und anschließend auf die Beendigung des DB2-Befehlszeilenprozessors wartet. Beispiel:

```
#include <windows.h>

int WINAPI DB2SetCLPEnv_api(DWORD pid);

void main ( int argc, char *argv [ ] )
{
      STARTUPINFO         startInfo   = {0};
      PROCESS_INFORMATION pidInfo     = {0};
      char     title  [32]   = "Synchrone Ausführung";
      char     runCmd [64]  =
                             "DB2 -z c:\\run.out -tvf c:\\run.clp";
/* Aufrufen der API zum Einrichten einer CLP-Umgebung */
      if ( DB2SetCLPEnv_api (GetCurrentProcessId ()) == 0 ) 1
      {
         startInfo.cb          = sizeof(STARTUPINFO);
         startInfo.lpReserved  = NULL;
         startInfo.lpTitle     = title;
         startInfo.lpDesktop   = NULL;
         startInfo.dwX         = 0;
         startInfo.dwY         = 0;
         startInfo.dwXSize     = 0;
         startInfo.dwYSize     = 0;
         startInfo.dwFlags     = 0L;
         startInfo.wShowWindow = SW_HIDE;
         startInfo.lpReserved2 = NULL;
         startInfo.cbReserved2 = 0;
               if ( CreateProcessA( NULL,
                              runCmd, 2
                              NULL,
                              NULL,
                              FALSE,
                              NORMAL_PRIORITY_CLASS CREATE_NEW_CONSOLE,
```

```
                              NULL,
                              NULL,
                              &startInfo,
                              &pidInfo))
            {
               WaitForSingleObject (pidInfo.hProcess, INFINITE);
               CloseHandle (pidInfo.hProcess);
               CloseHandle (pidInfo.hThread);
            }
         }
         return;
}
```

1 Die API DB2SetCLPEnv_api wird durch die Importbibliothek DB2API.LIB aufgelöst. Diese API definiert eine Umgebung, die das Aufrufen von CLP-Befehlen ermöglicht. Wenn dieses Programm aus dem Script db2cpre.bat heraus aufgerufen wird, wird der Befehlsprozessor warten, bis die CLP-Befehle abgeschlossen sind.

2 runCmd ist der Name der Script-Datei, die die DB2-CLP-Befehle enthält.

Im Unterverzeichnis MISC des DB2-Installationspfads befindet sich ein Beispielprogramm namens db2clpex.exe. Diese ausführbare Datei ist dem angeführten Beispiel ähnlich, jedoch akzeptiert sie den DB2-CLP-Befehl als Befehlszeilenparameter. Wenn Sie dieses Beispielprogramm verwenden wollen, kopieren Sie es in das Unterverzeichnis BIN. Sie können diese ausführbare Datei im Script db2cpre.bat wie folgt verwenden (INSTHOME ist das Exemplarverzeichnis):

```
db2clpex "DB2 -Z INSTHOME\pre.log -tvf INSTHOME\pre.clp"
```

Alle DB2-Befehle ATTACH oder Anweisungen CONNECT sollten explizit einen Benutzer angeben. Ansonsten werden sie unter dem Benutzerkonto ausgeführt, das dem Clusterservice zugeordnet ist. CLP-Scripts sollten außerdem mit dem Befehl TERMINATE schließen, um den CLP-Hintergrundprozess zu beenden.

Es folgt ein Beispiel einer Datei db2cpre.bat:

```
db2cpre.bat : 1
-----------------------
db2clpex "db2 -z INSTHOME\pre-%DB2NODE%.log -tvf INSTHOME\pre.clp" 2 - 5
-----------------------

PRE.CLP 6
-----------------------
update dbm cfg using MAXAGENTS 200;
get dbm cfg;
terminate;
-----------------------
```

1 Das Script db2cpre.bat wird unter dem Benutzerkonto ausgeführt,

das dem Clusterservice zugeordnet ist. Wenn DB2-Aktionen erforderlich sind, muss das dem Clusterservice zugeordnete Benutzerkonto eine gültige SQL-Kennung sein, wie sie von DB2 definiert wird.

2 INSTHOME ist das Exemplarverzeichnis.

3 Der Name der Protokolldatei muss für jeden Knoten unterschiedlich sein, um Dateikonflikte zu vermeiden, wenn beide logischen Knoten gleichzeitig online gebracht werden.

4 `db2clpex.exe` ist ein Beispielprogramm, das ein Befehlszeilenargument zur Angabe des auszuführenden CLP-Befehls akzeptiert.

5 Das Beispielprogramm `db2clpex.exe` muss auf allen MSCS-Clusterknoten verfügbar gemacht werden.

6 Die CLP-Befehle in diesem Beispiel begrenzen die Anzahl von Agenten.

Ausführen von Scripts, nachdem DB2-Ressourcen online gebracht wurden

Wenn Sie ein Script ausführen wollen, *nachdem* Sie eine DB2-Ressource online gebracht haben, *muss* das Script den Namen `db2cpost.bat` erhalten. Das Script wird von MSCS asynchron ausgeführt, wenn die DB2-Ressource erfolgreich online gebracht wurde. Der Befehl **db2cmd** kann in diesem Script verwendet werden, um DB2-CLP-Script-Dateien auszuführen. Verwenden Sie den Parameter "-c" des Befehls **db2cmd**, um anzugeben, dass das Dienstprogramm alle Fenster nach Beendigung der Task schließen soll. Beispiel:

```
db2cmd -c db2 -tvf mycmds.clp
```

Der Parameter "-c" muss das erste Argument für den Befehl **db2cmd** sein, da er Befehlsprozessoren ohne Verbindung im Hintergrund verhindert.

Das Script `db2cpost.bat` ist nützlich, wenn Sie Datenbankaktivitäten unmittelbar, nachdem die DB2-Ressource von einer anderen Maschine übernommen wurde und wieder aktiv wird, ausführen wollen. Zum Beispiel können Sie Datenbanken im Exemplar erneut starten oder aktivieren, so dass sie für den Benutzerzugriff vorbereitet sind.

Es folgt ein Beispiel eines Scripts `db2cpost.bat`:

```
db2cpost.bat 1
------------------------
db2cmd -c db2 -z INSTHOME\post-%DB2NODE%.log -tvf INSTHOME\post.clp 2 - 4
------------------------

POST.CLP 5
------------------------
restart database SAMPLE;
```

```
connect reset;
activate database SAMPLE;
terminate;
------------------------
```

1 Das Script `db2cpost.bat` wird unter dem Benutzerkonto ausgeführt, das dem Clusterservice zugeordnet ist. Wenn DB2-Aktionen erforderlich sind, muss das dem Clusterservice zugeordnete Benutzerkonto eine gültige SQL-Kennung sein, wie sie von DB2 definiert wird.

2 INSTHOME ist das Exemplarverzeichnis.

3 Der Name der Protokolldatei muss für jeden Knoten unterschiedlich sein, um Dateikonflikte zu vermeiden, wenn beide logischen Knoten gleichzeitig online gebracht werden.

4 Der Befehl **db2cmd** kann verwendet werden, weil das Script `db2cpost.bat` asynchron ausgeführt werden kann. Der Parameter ″-c″ muss zur Beendigung des Befehlsprozessors verwendet werden.

5 Das CLP-Script in diesem Beispiel enthält Befehle zum Neustarten und Aktivieren der Datenbank. Dieses Script versetzt die Datenbank unmittelbar nach dem Starten des Datenbankmanagers zurück in einen aktiven Status. In einem partitionierten Datenbanksystem sollten Sie den Befehl ACTIVATE DATABASE entfernen, weil mehrere DB2-Ressourcen gleichzeitig online gebracht werden. Der Befehl RESTART DATABASE kann fehlschlagen, weil ein anderer Knoten die Datenbank gerade aktiviert. Wenn dies eintritt, führen Sie das Script erneut aus, um sicherzugehen, dass die Datenbank korrekt erneut gestartet wird.

Überlegungen zu Datenbanken

Stellen Sie bei der Erstellung einer Datenbank sicher, dass der Datenbankpfad eine freigegebene gemeinsame Platte angibt. Dadurch wird die Datenbank auf allen MSCS-Knoten sichtbar. Alle Protokolle und anderen Datenbankdateien müssen ebenfalls auf Platten im Cluster angelegt werden, damit die DB2-Funktionsübernahme erfolgreich ausgeführt werden kann. Wenn Sie diese Punkte nicht beachten, tritt ein DB2-Systemfehler auf, weil es für DB2 so aussieht, als ob Dateien gelöscht wurden oder nicht mehr verfügbar sind.

Stellen Sie darüber hinaus sicher, dass die Konfigurationsparameter des Datenbankmanagers und der Datenbank so eingestellt sind, dass die von DB2 beanspruchten Systemressourcen auf beiden MSCS-Knoten vorhanden sind. Der Datenbankkonfigurationsparameter *autorestart* sollte auf `ON` gesetzt sein, so dass die erste Datenbankverbindung nach der Funktionsübernahme die Datenbank in einen konsistenten Status versetzt. (Die Standardeinstellung für *autorestart* ist `ON`.) Die Datenbank kann auch mit Hilfe des Scripts `db2cpost.bat` zum erneuten Starten und Aktivieren der Datenbank in einen bereiten Status versetzt werden. Diese Methode ist vorzuziehen, weil sie von

autorestart unabhängig ist und die Datenbank unabhängig von einer Verbindungsanforderung eines Benutzers in einen bereiten Status versetzt wird.

Benutzer- und Gruppenunterstützung

DB2 ist für die Unterstützung der Benutzerauthentifizierung und Gruppen auf Windows NT angewiesen. Damit ein DB2-Exemplar eines MSCS-Knotens von einem anderen reibungslos übernommen werden kann, muss jeder MSCS-Knoten Zugriff auf die gleichen Sicherheitsdatenbanken von Windows NT haben. Sie können dies erreichen, indem Sie die Domänensicherheit von Windows NT (Domain Security) nutzen.

Definieren Sie alle DB2-Benutzer und DB2-Gruppen in einer Domänensicherheitsdatenbank. Die MSCS-Knoten müssen dieser Domäne angehören, oder die Domäne muss eine gesicherte Domäne (Trusted Domain) sein. DB2 verwendet dann unabhängig von dem MSCS-Knoten, auf dem DB2 aktiv ist, die Domänensicherheitsdatenbank zur Authentifizierung und für die Gruppenunterstützung.

Wenn Sie lokale Konten verwenden, müssen die Konten auf jedem MSCS-Knoten repliziert werden. Dieses Verfahren wird nicht empfohlen, weil es fehlerträchtig ist und eine doppelte Pflege verlangt.

DCE-Sicherheit ist ebenfalls ein unterstützter Authentifizierungsmodus, wenn alle MSCS-Knoten Clients in der gleichen DCE-Zelle sind.

Sie sollten dem MSCS-Service ein Benutzerkonto zuordnen, das der DB2-Namenskonvention entspricht. Dadurch kann der MSCS-Service Aktionen für DB2 ausführen, die in den Scripts `db2cpre.bat` und `db2cpost.bat` aufgerufen werden.

Weitere Informationen zur Unterstützung von Benutzern und Gruppen unter Windows NT finden Sie in ″Benutzerauthentifizierung mit DB2 für Windows NT″ im Handbuch *Systemverwaltung: Implementierung*.

Überlegungen zur Kommunikation

DB2 unterstützt zwei LAN-Protokolle in einer MSCS-Umgebung:

- TCP/IP
- NetBIOS

TCP/IP wird unterstützt, weil es ein unterstützter Clusterressourcentyp ist. Zur Aktivierung von DB2 zur Verwendung von TCP/IP als Kommunikationsprotokoll für ein partitioniertes Datenbanksystem erstellen Sie eine IP-Adressressource und platzieren sie in derselben Gruppe wie die DB2-Ressource, die den Datenbankpartitionsserver repräsentiert, den Sie als Koordinatorknoten für ferne Anwendungen verwenden wollen. Dann erstellen Sie eine Abhängig-

keit mit Hilfe von Cluster Administrator, um sicherzustellen, dass die IP-Ressource online ist, bevor die DB2-Ressource gestartet wird. DB2-Clients können dann TCP/IP-Knotenverzeichniseinträge katalogisieren, um diese TCP/IP-Adresse zu verwenden.

Der TCP/IP-Port, der dem Konfigurationsparameter *svcename* des Datenbankmanagers zugeordnet ist, muss zur Verwendung durch das DB2-Exemplar auf allen Maschinen reserviert werden, die am Exemplar beteiligt sind. Der Servicename, der der Portnummer zugeordnet ist, muss außerdem in der Datei `services` auf allen Maschinen identisch sein.

Obwohl NetBIOS keine unterstützte Clusterressource ist, können Sie NetBIOS als LAN-Protokoll verwenden, weil das Protokoll sicherstellt, dass NetBIOS-Namen im LAN eindeutig sind. Wenn DB2 einen NetBIOS-Namen registriert, stellt NetBIOS sicher, dass der Name im LAN nicht bereits verwendet wird. Wenn DB2 im Szenario einer Funktionsübernahme von einem System auf ein anderes übertragen wird, wird der von DB2 verwendete *nname* von einer Partnermaschine im MSCS-Cluster deregistriert und auf der anderen Maschine registriert.

Die DB2-Unterstützung für NetBIOS verwendet NetBIOS-Rahmen (NBF - NetBIOS Frames). Dieser Protokollstapelspeicher kann verschiedenen logischen Adapternummern (LANA) zugeordnet werden. Damit ein konsistenter NetBIOS-Zugriff auf den Server sichergestellt ist, sollte die dem NBF-Protokollstapel zugeordnete LANA auf allen Knoten im Cluster gleich sein. Sie können dies mit Hilfe der Option **Netzwerke** der Systemsteuerung konfigurieren. Sie sollten NBF der LANA 0 zuordnen, weil dies die von DB2 erwartete Standardeinstellung ist.

Überlegungen zur Systemzeit

DB2 verwendet die Systemzeit, um bestimmte Operationen mit einer Zeitmarke zu versehen. Alle MSCS-Knoten, die an einer DB2-Funktionsübernahme beteiligt sind, müssen eine synchronisierte Systemzeitzone und Systemzeit haben, um eine konsistente Funktionsweise von DB2 auf allen Maschinen sicherzustellen.

Stellen Sie die Systemzeitzone mit Hilfe der Option **Datum/Uhrzeit** der Systemsteuerung ein. MSCS hat einen Zeitservice, der das Datum und die Uhrzeit synchronisiert, wenn die MSCS-Knoten zu einem Cluster verbunden werden. Der Zeitservice synchronisiert die Zeit allerdings nur alle zwölf Stunden. Dadurch kann es zu Problemen kommen, wenn die Zeit auf einem System geändert und DB2 übernommen wird, bevor die Zeit synchronisiert wird.

Wenn die Zeit auf einem der MSCS-Clusterknoten geändert wird, sollte sie manuell auf den anderen Clusterknoten mit folgendem Befehl synchronisiert werden:

```
net time /set /y \\ferner-knoten
```

Dabei ist *ferner-knoten* der Maschinenname des Clusterknotens.

Überlegungen zu Verwaltungsserver und Steuerzentrale in Umgebungen mit partitionierten Datenbanken

Der DB2-Verwaltungsserver wird (optional) während der Installation von DB2 Universal Database erstellt. Er ist kein partitioniertes Datenbanksystem. Die Steuerzentrale verwendet vom Verwaltungsserver zur Verfügung gestellte Services, um DB2-Exemplare und DB2-Datenbanken zu verwalten.

In einem partitionierten Datenbanksystem kann sich ein DB2-Exemplar auf mehreren MSCS-Knoten befinden. Dies impliziert, dass ein DB2-Exemplar auf mehreren Systemen unter der Steuerzentrale katalogisiert sein muss, so dass der Zugriff auf das Exemplar erhalten bleibt, unabhängig davon, auf welchem MSCS-Knoten das DB2-Exemplar aktiv ist.

Das Exemplarverzeichnis des Verwaltungsservers wird nicht gemeinsam verwendet. Sie müssen alle benutzerdefinierten Dateien im Verzeichnis des Verwaltungsservers auf allen MSCS-Knoten spiegeln, um die gleiche Verwaltungsstufe auf allen MSCS-Knoten zur Verfügung zu stellen. Insbesondere müssen Benutzer-Scripts und geplante ausführbare Dateien auf allen Knoten verfügbar sein. Außerdem müssen Sie sicherstellen, dass geplante Aktivitäten auf allen Maschinen in einem MSCS-Cluster geplant sind.

Alternativ zum Duplizieren des Verwaltungsservers auf allen Maschinen kann es wünschenswert sein, eine Funktionsübernahme für den Verwaltungsserver zu ermöglichen. Nehmen Sie für das folgende Beispiel an, Sie hätten zwei MSCS-Knoten mit den Namen MACH0 und MACH1 im Cluster. MACH0 hat Zugriff auf eine Clusterplatte, die vom Verwaltungsserver verwendet wird. Nehmen Sie außerdem an, dass MACH0 und MACH1 einen Verwaltungsserver haben. Um den Verwaltungsserver hochverfügbar zu machen, würden Sie folgende Schritte ausführen:

1. Stoppen Sie den Verwaltungsserver auf beiden Maschinen, indem Sie den Befehl **db2admin stop** auf jeder Maschine ausführen.
2. Entfernen Sie auf allen Verwaltungs-Client-Maschinen alle Verweise auf die Verwaltungsserver von MACH0 und MACH1 mit dem Befehl UNCATALOG NODE aus dem Katalog. (Mit Hilfe des Befehls LIST NODE DIRECTORY auf der Client-Maschine können Sie feststellen, ob Verweise auf den Verwaltungsserver vorhanden sind.)
3. Löschen Sie den Verwaltungsserver von MACH1, indem Sie von MACH1 aus den Befehl **db2admin drop** ausführen. (Dieser Schritt ist nur erforderlich, wenn auf beiden Maschinen ein Verwaltungsserver vorhanden ist.)

4. Stellen Sie den Namen des Verwaltungsservers fest, indem Sie von MACH0 aus den Befehl **db2admin** ausführen. (Der Standardname ist DB2DAS00.)
5. Verwenden Sie das Dienstprogramm DB2MSCS, um die Unterstützung für die Funktionsübernahme des Verwaltungsservers einzurichten. Dies zieht die Erstellung einer DB2-Ressource auf MSCS namens DB2DAS00 nach sich, die Abhängigkeiten von der IP-Ressource und den Plattenressourcen (DISK) hat. (Wenn Sie eine Konfiguration zur gegenseitigen Übernahme haben, würden Sie die Ressource in die Gruppe stellen, die die DB2-Ressource für NODE0 enthält.) Diese Ressource wird als die MSCS-Ressource verwendet, die den Verwaltungsserver unterstützt. Die Datei DB2MSCS.ADMIN sähe wie folgt aus:

```
#
# db2mscs.admin für Verwaltungsserver
# run db2mscs -f:db2mscs.admin
#
DB2_INSTANCE=DB2DAS00
CLUSTER_NAME=CLUSTERA
DB2_LOGON_USERNAME=db2admin
DB2_LOGON_PASSWORD=db2admin
# den Verwaltungsserver in die gleiche Gruppe wie DB2 Knoten 0 stellen
GROUP_NAME=DB2NODE0 1
DISK_NAME=DISK E:
INSTPROF_DISK=DISK E:
IP_NAME= IP-Adresse für Verwaltungsserver
IP_ADDRESS=9.9.9.8
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
```

   1 Diese Gruppe kann die gleiche wie die vorhandene sein. Auf diese Weise benötigen Sie keine zusätzliche Platte für das Exemplarprofilverzeichnis.

6. Führen Sie auf MACH1 den folgenden Befehl aus, um DB2DAS00 als den Verwaltungsserver zu definieren:

```
db2set -g db2adminserver=DB2DAS00
```

7. Ändern Sie auf MACH0 die Startmerkmale von DB2DAS00 über das Programm **Dienste**, so dass er manuell und nicht automatisch gestartet wird, da DB2DAS00 nun von MSCS gesteuert wird.

Wenn der Verwaltungsserver für die Funktionsübernahme aktiviert ist, sollten alle fernen Zugriffe eine MSCS-IP-Ressource zur Kommunikation mit dem Verwaltungsserver verwenden. Der Verwaltungsserver hat nun die folgenden Merkmale:

- Das Exemplarverzeichnis des Verwaltungsservers wird zusammen mit dem Verwaltungsserver übernommen.

- Clients katalogisieren nur einen einzigen Knoten zur Kommunikation mit dem Verwaltungsserver, unabhängig davon, auf welchem MSCS-Knoten der Verwaltungsserver aktiv ist.
- Jobs müssen nur einmal für den Verwaltungsserver geplant werden.
- Lokale Exemplare können vom Verwaltungsserver nur gesteuert werden, wenn der Verwaltungsserver auf demselben Knoten aktiv ist wie das lokale Exemplar.
- Der Zugriff auf den Verwaltungsserver ist nicht möglich, wenn der Cluster-Service nicht aktiv ist.

Einschränkungen und Begrenzungen

Bei der Ausführung von DB2 in einer MSCS-Umgebung gibt es folgende Einschränkungen:

- Sie können auf den gemeinsamen Platten keine physische E/A verwenden, es sei denn, die gemeinsamen Platten haben auf beiden MSCS-Knoten dieselbe physische Plattennummer. Sie können logische E/A verwenden, weil auf die Platte mit Hilfe einer Partitionskennung zugegriffen wird.
- Sie müssen alle DB2-Ressourcen für MSCS-Unterstützung konfigurieren. Andernfalls treten Systemfehler während der Laufzeit von DB2 auf (DB2 kann nicht ordnungsgemäß funktionieren, wenn Systemressourcen fehlen). Wenn zum Beispiel die Datenbankprotokolle nicht auf einer gemeinsamen MSCS-Platte gespeichert werden, kann DB2 die Datenbank nicht erneut starten.
- Sie müssen das DB2-Exemplar über das Tool **Cluster Administrator** verwalten. MSCS betrachtet andere Methoden, die zum Starten und Stoppen des Datenbankmanagers verwendet werden, als Softwareunregelmäßigkeiten. Wenn Sie zum Beispiel DB2 über MSCS starten und den Befehl **db2stop** zum Stoppen von DB2 verwenden, erkennt MSCS dies als Softwarefehler und startet das Exemplar erneut. Dies heißt natürlich auch, dass Sie die Steuerzentrale nicht zum Starten und Stoppen von DB2 verwenden sollten.
- Zum Deinstallieren von DB2 müssen Sie MSCS zuerst stoppen.

Kapitel 8. Hohe Verfügbarkeit in der Solaris-Betriebsumgebung

Eine hohe Verfügbarkeit in der Solaris-Betriebsumgebung können Sie mit DB2 unter Einsatz von Sun Cluster 2.x (SC2.x), Sun Cluster 3.0 (SC3.0) oder Veritas Cluster Server (VCS) erreichen. Weitere Informationen zu Sun Cluster 3.0 finden Sie im White Paper mit dem Titel „DB2 and High Availability on Sun Cluster 3.0", das auf der Website „DB2 UDB and DB2 Connect Online Support" unter http://www.ibm.com/software/data/pubs/papers/ verfügbar ist. Weitere Informationen zu VERITAS Cluster Server finden Sie im White Paper mit dem Titel „DB2 and High Availability on VERITAS Cluster Server", das ebenfalls auf der Website „DB2 UDB and DB2 Connect Online Support" verfügbar ist.

Dieses Kapitel beschreibt detailliert, wie DB2 mit Sun Cluster 2.x (SC2.x) 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. 31).

Abbildung 31. DB2, Sun Cluster 2.x und hohe Verfügbarkeit

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, einer oder mehreren öffentlichen Netzwerkschnittstellen sowie einigen gemeinsam benutzten Platten. Diese spezielle Konfiguration ermöglicht das Versetzen eines Datenbankservice von einer Maschine auf eine andere. Durch das Versetzen des Datenbankservice auf eine andere Maschine im Cluster wird es möglich, den Zugriff auf die zugehörigen Daten weiterhin aufrecht zu erhalten. Das Versetzen eines Datenbankservice von einer Maschine auf eine andere wird als *Funktionsübernahme* bezeichnet (siehe Abb. 32 zur Illustration).

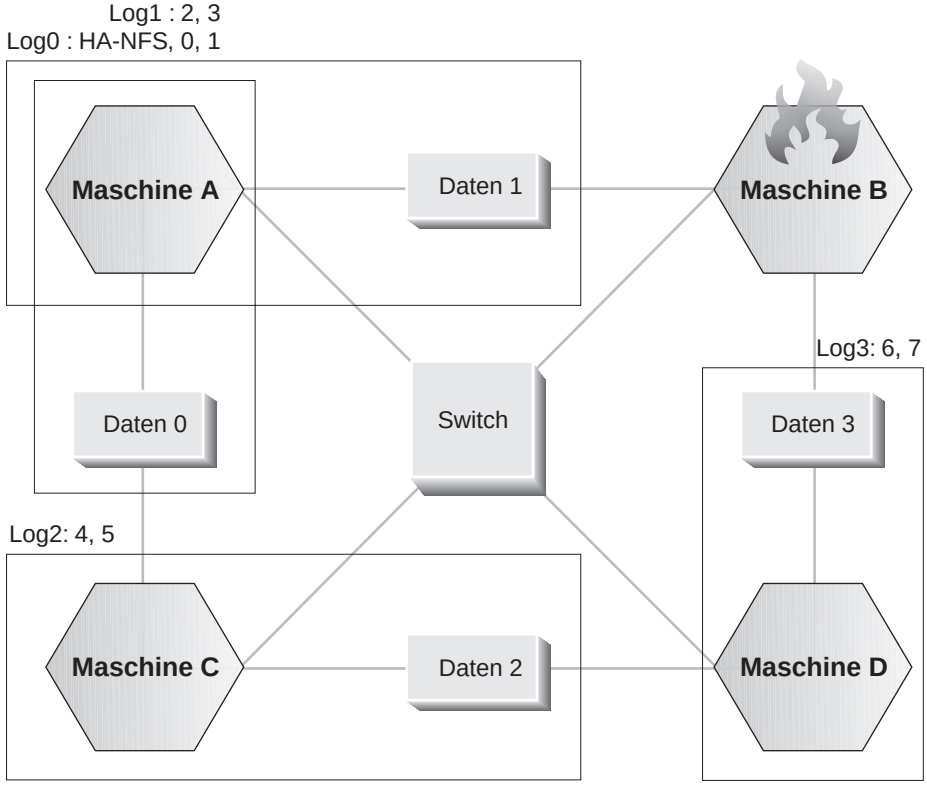


Abbildung 32. Funktionsübernahme

Die privaten Netzwerkschnittstellen werden zum Senden von *Heartbeat*-Nachrichten und von Steuernachrichten unter 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-Datenbankservice stellen eine Verbindung über TCP/IP nur zu den logischen Netzwerkschnittstellen des Datenbankservice 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 Datenbankservice erwarten, dass sich die Netzadresse und der Host-Name des Datenbankservice 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 Web-Browser, der auf einen hochverfügbaren Webserver zugreift. Die Anforderung wird mit einer URL-Adresse (Uniform Resource Locator) abgesetzt, die sowohl einen Host-Namen als auch den Pfad zu einer Datei auf dem Webserver enthält. Der Browser erwartet, dass sowohl der Host-Name als auch der Pfad auch 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 Datenbankservice wird in der Zeitdauer gemessen, die der Datenbankservice seinen Benutzern zur Verfügung steht. Die am häufigsten verwendete Maßeinheit für Verfügbarkeit ist der Prozentsatz der Betriebszeit (Up Time). Diese wird oft 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 jedesmal 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 fünfmal 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 und fortlaufende Verfügbarkeit

Eine andere Methode zur Erhöhung der Verfügbarkeit eines Datenbankservice 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.

Die *fortlaufende Verfügbarkeit* geht einen Schritt über die hohe Verfügbarkeit hinaus. Sie schützt die Clients sowohl gegen geplante als auch gegen ungeplante Ausfallzeiten. Bei einer Konfiguration für fortlaufende Verfügbarkeit bleibt der Client von Ausfällen einer der Maschinen, die den Datenbankservice enthalten, bzw. von Wartungszeiten für einzelne Maschinen völlig unberührt. Konfigurationen für fortlaufende Verfügbarkeit sind komplex und kostenintensiver in der Implementierung.

Ein HA-Cluster ist die gängigste Lösung zur Erhöhung der Verfügbarkeit, weil sie skalierbar, einfach zu handhaben und relativ kostengünstig in der Implementierung ist.

Sun Cluster 2.2

Sun Cluster 2.2 (SC2.2) ist das Produkt zum Clustering und zur Implementierung hoher Verfügbarkeit von Sun Microsystems. SC2.2 unterstützt bis zu vier Maschinen in einem einzigen Cluster. Bei Einsatz von vier Ultra Enterprise 10000s kann ein Cluster bis zu 256 CPUs und 256 GB RAM besitzen.

Unterstützte Systeme

System	UltraSPARC	Speicherkapazität	E/A
Ultra Enterprise 1	1	64 MB - 1 GB	3 SBus
Ultra Enterprise 2	1-2	64 MB - 2 GB	4 SBus
Ultra Enterprise 450	1-4	32 MB - 4 GB	10 PCI
Ultra Enterprise 3000	1-6	64 MB - 6 GB	9 SBus
Ultra Enterprise 4000	1-14	64 MB - 14 GB	21 SBus
Ultra Enterprise 5000	1-14	64 MB - 14 GB	21 SBus
Ultra Enterprise 6000	1-30	64 MB - 30 GB	45 SBus
Ultra Enterprise 10000	1-64	512 MB - 64 GB	64 SBus

Agenten

Die Sun Cluster-Software enthält eine Anzahl von Agenten für hohe Verfügbarkeit (HA-Agenten), die unterstützt und mit dem Produkt SC2.2 ausgeliefert werden. Andere HA-Agenten, wie zum Beispiel der für DB2, werden außerhalb von Sun entwickelt und nicht zusammen mit der Sun Cluster-Software ausgeliefert. Der HA-Agent für DB2 gehört zum Lieferumfang von DB2, wird von IBM unterstützt und mit DB2 kostenlos ausgeliefert.

Die Sun Cluster-Software arbeitet mit hochverfügbaren Datenbankservices, indem sie eine Möglichkeit gibt, Methoden (Scripts oder Programme) zu registrieren, die verschiedenen Komponenten der Sun Cluster-Software entsprechen. Mit Hilfe dieser Methoden kann die Software von SC2.2 einen Datenbankservice steuern, ohne über eingehende Kenntnisse über ihn zu verfügen. Zu diesen Methoden gehören Folgende:

START
: Dient zum Starten von Teilen des Datenbankservice, bevor die logischen Netzwerkschnittstellen online sind.

START_NET
Dient zum Starten von Teilen des Datenbankservice, nachdem die logischen Netzwerkschnittstellen online gebracht sind.

STOP Dient zum Stoppen von Teilen des Datenbankservice, nachdem die logischen Netzwerkschnittstellen offline genommen wurden.

STOP_NET
Dient zum Stoppen von Teilen des Datenbankservice, bevor die logischen Netzwerkschnittstellen offline genommen werden.

ABORT
Wie die Methode STOP, jedoch wird sie ausgeführt unmittelbar, bevor eine Maschine durch die Clustersoftware heruntergefahren wird. In diesem Fall ist der einwandfreie Zustand der Maschine fraglich, und ein Datenbankservice könnte noch einige „letzte" Anforderungen ausführen, bevor die Maschine heruntergefahren wird. Wird ausgeführt, nachdem die logischen Netzwerkschnittstellen offline genommen wurden.

ABORT_NET
Wie die Methode ABORT, allerdings wird sie ausgeführt, bevor die logischen Netzwerkschnittstellen offline genommen werden.

FM_INIT
Dient zur Initialisierung von Fehlermonitoren.

FM_START
Dient zum Starten von Fehlermonitoren.

FM_STOP
Dient zum Stoppen von Fehlermonitoren.

FM_CHECK
Wird durch den Befehl **hactl** aufgerufen. Liefert den aktuellen Status des entsprechenden Datenbankservice.

Der DB2-Agent besteht aus folgenden Scripts: START_NET, STOP_NET, FM_START und FM_STOP. Die folgenden Scripts werden bei der Neukonfiguration des Clusters nicht ausgeführt: ABORT, ABORT_NET und FM_CHECK.

Ein Agent für hohe Verfügbarkeit besteht aus mindestens einer dieser Methoden. Die Methoden werden bei SC2.2 durch den Befehl **hareg** registriert. Nach der Registrierung ruft die Sun Cluster-Software die entsprechende Methode auf, um den Datenbankservice zu steuern.

Es ist sehr wichtig, zu beachten, dass die Methoden ABORT und STOP eines Service möglicherweise nicht aufgerufen werden. Diese Methoden sind zum kontrollierten Herunterfahren eines Datenbankservice gedacht, und der

Datenbankservice muss wiederhergestellt werden können, wenn eine Maschine ausfällt, ohne diese Methoden aufzurufen.

Weitere Informationen finden Sie in der Sun Cluster-Dokumentation.

Logische Hosts

Die SC2.2-Software arbeitet mit dem Konzept eines logischen Hosts. Ein *logischer Host* besteht aus einer Gruppe von Platten und mindestens einer öffentlichen Netzwerkschnittstelle. Ein hochverfügbarer Datenbankservice wird einem logischen Host zugeordnet und benötigt die Platten, die in den Plattengruppen des logischen Hosts sind. Logische Hosts können auf verschiedenen Maschinen im Cluster enthalten sein und sich die CPUs und den Hauptspeicher der Maschine „entleihen", auf der sie aktiv sind.

Logische Netzwerkschnittstellen

Ebenso wie andere UNIX-Betriebssysteme verfügt Solaris über die Möglichkeit, neben der primären IP-Adresse für eine Netzwerkschnittstelle zusätzliche IP-Adressen zu besitzen. Die zusätzlichen IP-Adressen befinden sich auf einer logischen Schnittstelle in der gleichen Weise, wie sich die primäre IP-Adresse auf der physischen Netzwerkschnittstelle befindet. Das folgende Beispiel zeigt die logischen Schnittstellen auf zwei Maschinen in einem Cluster. Es gibt zwei logische Hosts, die sich zur Zeit beide auf der Maschine "thrash" befinden.

```
scadmin@crackle(202)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 289966 0 289966 0 0 0
hme0 1500 9.21.55.0 9.21.55.98 121657 6098 764122 0 0 0
scid0 16321 204.152.65.0 204.152.65.1 489307 0 476479 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.33 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.17 347317 0 348073 0 0 0
```

1. lo0 ist die loopback-Schnittstelle
2. hme0 ist die öffentliche Netzwerkschnittstelle (ethernet)
3. scid0 ist die erste private Netzwerkschnittstelle (SCI oder Scalable Coherent Interface)
4. scid0:1 ist eine logische Netzwerkschnittstelle, die von der Sun Cluster-Software intern verwendet wird
5. scid1 ist die zweite private Netzwerkschnittstelle

```
scadmin@thrash(203)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 1128780 0 118780 0 0 0
hme0 1500 9.21.55.0 9.21.55.92 1741422 5692 757127 0 0 0
hme0:1 1500 9.21.55.0 9.21.55.109 0 0 0 0 0 0
hme0:2 1500 9.21.55.0 9.21.55.110 0 0 0 0 0 0
scid0 16321 204.152.65.0 204.152.65.2 476641 0 489476 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.34 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.18 348199 0 347444 0 0 0
```

```
1. hme0:1 ist eine logische Netzwerkschnittstelle für einen
   logischen Host
2. hme0:2 ist eine log. Netzwerkschnittstelle für einen anderen
   logischen Host
```

Einem logischen Host können eine oder mehrere logische Schnittstellen zugeordnet sein. Diese logischen Schnittstellen werden mit dem logischen Host von einer Maschine auf eine andere versetzt und dienen dem Zugriff auf den Datenbankservice, der dem logischen Host zugeordnet ist. Da diese logischen Schnittstellen mit den logischen Hosts versetzt werden, können Clients auf den Datenbankservice unabhängig von der Maschine, auf der er sich befindet, zugreifen.

Ein hochverfügbarer Datenbankservice sollte an die TCP/IP-Adresse INADDR_ANY gebunden werden. Dadurch wird gewährleistet, dass jede IP-Adresse auf dem System Verbindungen für den Datenbankservice empfangen kann. Wenn ein Datenbankservice stattdessen an eine bestimmte IP-Adresse gebunden wird, muss er an die logische Schnittstelle gebunden werden, die dem logischen Host zugeordnet ist, auf dem der Datenbankservice aktiv ist. Das Binden an INADDR_ANY beseitigt die Notwendigkeit, den Service an eine neue IP-Adresse zu binden, wenn eine IP-Adresse auf dem System eintrifft, die vom Datenbankservice benötigt wird.

Anmerkung: Clients eines HA-Exemplars sollten die Datenbank mit dem Host-Namen für die logische IP-Adresse eines logischen Hosts katalogisieren. Sie sollten nie den primären Host-Namen für eine Maschine verwenden, weil es keine Garantie gibt, dass DB2 auf dieser Maschine ausgeführt wird.

Plattengruppen und Dateisysteme

Platten für einen Datenbankservice werden einem logischen Host in Gruppen (bzw. Sätzen) zugeordnet. Wenn der Cluster mit Sun StorEdge Volume Manager (Veritas) arbeitet, verwendet die Sun Cluster-Software das Veritas-Dienstprogramm ″vxdg″, um die Plattengruppen für jeden logischen Host zu importieren und zu deportieren. Das folgende Beispiel zeigt die Plattengruppen für zwei logische Hosts ″log0″ und ″log1″, die auf einer Maschine namens ″thrash″ untergebracht sind. Die Maschine mit dem Namen ″crackle″ enthält zurzeit keine logischen Hosts.

```
scadmin@crackle(206)# vxdg list
NAME STATE ID
rootdg enabled 899825206.1025.crackle

scadmin@thrash(205)# vxdg list
NAME STATE ID
rootdg enabled 924176206.1025.thrash
data0 enabled 925142028.1157.crackle=
data1 enabled 899826248.1108.crackle
```

Die Plattengruppen ″data0″ und ″data1″ entsprechen jeweils den logischen Hosts ″log0″ und ″log1″. Die Plattengruppe ″data0″ kann von ″thrash″ mit folgendem Befehl deportiert werden:

```
vxdg deport data0
```

Anschließend kann sie mit folgendem Befehl auf ″crackle″ importiert werden:

```
vxdg import data1
```

Dies wird durch die Sun Cluster-Software automatisch durchgeführt und sollte auf einem im Betrieb befindlichen Cluster nicht manuell vorgenommen werden.

Jede Plattengruppe enthält eine Anzahl von Platten, die von zwei oder mehr Maschinen im Cluster gemeinsam benutzt werden können. Ein logischer Host kann nur auf eine andere Maschine versetzt werden, die physischen Zugriff auf die Platten in den Plattengruppen hat, die ihm zugeordnet sind.

Zwei Dateien steuern die Dateisysteme für jeden logischen Host:

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logischer-host>
/etc/opt/SUNWcluster/conf/hanfs/dfstab.<logischer-host>
```

Dabei ist *logischer-host* der Name des zugeordneten logischen Hosts.

Die Datei `vfstab` ist der Datei `/etc/vfstab` ähnlich, abgesehen davon, dass sie Einträge für die Dateisysteme enthält, die angehängt werden sollen, nachdem die Plattengruppen für einen logischen Host importiert wurden. Die Datei `dfstab` ist der Datei `/etc/dfs/dfstab` ähnlich, jedoch enthält sie Einträge für die Dateisysteme, die über HA-NFS für einen logischen Host exportiert werden sollen. Jede Maschine verfügt über eine eigene Kopie dieser Datei, und es sollte sorgfältig darauf geachtet werden, dass sie auf jeder Maschine im Cluster den gleichen Inhalt haben.

Anmerkung: Die Pfade für die Dateien `vfstab` und `dfstab` eines logischen Hosts sind irreführend, weil sie das Verzeichnis `hanfs` enthalten. Nur die Datei `dfstab` für einen logischen Host wird für HA-NFS verwendet. Die Datei `vfstab` wird verwendet, selbst wenn HA-NFS nicht konfiguriert ist.

Die folgenden Beispiele stammen aus einem Cluster, auf dem DB2 Universal Database Enterprise - Extended Edition (EEE) in einer Konfiguration zur gegenseitigen Übernahme ausgeführt wird:

```
scadmin@thrash(217)# ls -l /etc/opt/SUNWcluster/conf/hanfs
total 8
-rw-r--r-- 1 root build 173 Apr 14 15:01 dfstab.log0
-rw-r--r-- 1 root build 316 Apr 26 12:07 vfstab.log0
-rw-r--r-- 1 root build 389 Apr 13 21:04 vfstab.log1
```

```
scadmin@thrash(218)# cat dfstab.log0
share -F nfs -o root=crackle:thrash:\
jolt:bump:crackle.torolab.ibm.com:thrash.torolab.ibm.com:\
jolt.torolab.ibm.com:bump.torolab.ibm.com /log0/home
```

Die Hosts, denen die Berechtigung zum Anhängen des Dateisystems `/log0/home` erteilt wird, stammen von allen Netzwerkschnittstellen (logischen und physischen) auf jeder Maschine im Cluster. Die Dateisysteme werden mit Root-Berechtigung exportiert.

```
scadmin@thrash(220)# cat vfstab.log0
#device to mount             device  to fsck               mount
#                                                          point

/dev/vx/dsk/data0/data1-stat /dev/vx/rdsk/data0/data1-stat /log0
/dev/vx/dsk/data0/vol01      /dev/vx/rdsk/data0/vol01      /log0/home
/dev/vx/dsk/data0/vol02      /dev/vx/rdsk/data0/vol02      /log0/data

scadmin@thrash(221)# cat vfstab.log1
#device to mount             device  to fsck               mount
#                                                          point
/dev/vx/dsk/data1/data1-stat /dev/vx/rdsk/data1/data1-stat /log1
/dev/vx/dsk/data1/vol01      /dev/vx/rdsk/data1/vol01      /log1/home
/dev/vx/dsk/data1/vol02      /dev/vx/rdsk/data1/vol02      /log1/data
/dev/vx/dsk/data1/vol03      /dev/vx/rdsk/data1/vol03      /log1/data1



FS   fsck mount   options
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -


FS   fsck mount   options
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -
ufs  2    no      -
```

Die Datei `vfstab.log0` enthält drei gültige Einträge für Dateisysteme unter dem Verzeichnis `/log0`. Beachten Sie, dass die Dateisysteme für den logischen Host `log0` logische Datenträgereinheiten verwenden, die Teil der Plattengruppe `data0` sind, die dem logischen Host zugeordnet ist.

Die Dateisysteme in den Dateien `vfstab` werden der Reihe nach von oben nach unten angehängt, so dass es wichtig ist, sicherzustellen, dass die Dateisysteme in der richtigen Reihenfolge aufgelistet sind. Dateisysteme, die unter-

halb eines bestimmten Dateisystems angehängt werden, müssen unter diesem Dateisystem aufgelistet werden. Die tatsächlich für einen logischen Host benötigten Dateisysteme hängen von den Anforderungen des Datenbankservice ab und werden sich von diesen Beispielen erheblich unterscheiden.

Während einer Funktionsübernahme ist die SC2.2-Software dafür zuständig, zu gewährleisten, dass die Plattengruppen und logischen Schnittstellen, die einem logischen Host zugeordnet sind, dem Host im Cluster von Maschine zu Maschine folgen. Der hochverfügbare Datenbankservice erwartet, dass ihm zumindest diese Ressourcen auf einem neuen System nach einer Funktionsübernahme zur Verfügung stehen. Tatsächlich ist es so, dass die meisten Datenbankservices selbst gar nicht erkennen, dass sie hochverfügbar sind, so dass diese Ressourcen auch nach einer Funktionsübernahme als exakt die gleichen erscheinen müssen.

Steuermethoden

Die Steuermethoden werden mit folgendem Befehl registriert:

```
hareg(1m)
```

Wenn ein HA-Service registriert ist, ist SC2.2 dafür zuständig, die Methoden, die für den HA-Service registriert wurden, zu gegebener Zeit während einer Änderung der Clusterkonfiguration oder einer Funktionsübernahme aufzurufen.

Während einer Änderung der Clusterkonfiguration (kontrollierten Funktionsübernahme) finden die folgenden Aktionen (in der angegebenen Reihenfolge) statt. Die Aktionen vor Schritt 5c werden nicht durchgeführt, falls die Maschine abstürzt. (Weitere Informationen zur Änderung der Clusterkonfiguration finden Sie in der SC2.2-Dokumentation.)

```
 1. Methode FM_STOP wird ausgeführt.
 2. Methode STOP_NET wird ausgeführt.
 3. Logische Schnittstellen für den logischen Host werden offline gebracht.
      - ifconfig hme0:1 0.0.0.0 down
 4. Methode STOP wird ausgeführt.
 5. Plattengruppen und Dateisysteme werden versetzt.
      a. Dateisysteme des logischen Hosts werden abgehängt.
      b. vxdg deportiert Plattengruppen auf einer Maschine.

 - Nur die folgenden Schritte werden ausgeführt, wenn eine Maschine abstürzt -

      c. vxdg importiert Plattengruppen auf die andere Maschine.
      d. fsck wird für Dateisysteme des logischen Hosts ausgeführt.
      e. Dateisysteme des logischen Hosts werden angehängt.
 6. Methode START wird ausgeführt.
 7. Logische Schnittstellen für den logischen Host werden online gebracht.
      - ifconfig hme0:1 <ip address> up
 8. Methode START_NET wird ausgeführt.
 9. Methode FM_INIT wird ausgeführt.
10. Methode FM_START wird ausgeführt.
```

Die Steuermethoden werden mit den folgenden Befehlszeilenparametern ausgeführt:

```
METHODE <untergebrachte logische Hosts> <nicht untergebrachte
logische Hosts> <Zeitlimit>
```

Der erste Parameter ist eine durch Kommas getrennte Liste logischer Hosts, die momentan auf der Maschine untergebracht sind, und der zweite Parameter eine durch Kommas getrennte Liste logischer Hosts, die nicht auf der Maschine untergebracht sind. Der letzte Parameter ist ein Zeitlimit für die Methode, d. h. die Zeit, die die Methode aktiv sein kann, bevor die SC2.2-Software sie abbricht.

Konfiguration von Platten und Dateisystemen

SC2.2 unterstützt zwei Datenträgermanager: Sun StorEdge Volume Manager (Veritas) und Solstice Disk Suite. Obwohl beide gut funktionieren, besitzt StorEdge Volume Manager einige Vorteile in einer Clusterumgebung. In einigen Clusterkonfigurationen kann die Controllernummer für eine Platteneinheit von Maschine zu Maschine im Cluster unterschiedlich sein. Wenn sich eine Controllernummer unterscheidet, unterscheiden sich auch die Pfade für die Platteneinheiten für den Controller. Da Disk Suite direkt mit den Pfaden der Plattengeräte arbeitet, funktioniert das Produkt in diesem Fall nicht gut. StorEdge Volume Manager arbeitet hingegen mit den Platten selbst, unabhängig von der Controllernummer, und wird durch unterschiedliche Controllernummern nicht beeinträchtigt. Da das Ziel von HA in einer Erhöhung der Verfügbarkeit für einen Datenbankservice liegt, spielt es eine wichtige Rolle, sicherzustellen, dass alle Dateisysteme und Platteneinheiten gespiegelt werden oder in einer RAID-Konfiguration angeordnet sind. Dadurch werden Funktionsübernahmen aufgrund einer ausgefallenen Platte vermieden, und die Stabilität des Clusters wird erhöht.

HA-NFS

DB2 UDB EEE erfordert ein gemeinsam benutztes Dateisystem, wenn ein Exemplar über mehrere Maschinen konfiguriert wird. Eine typische DB2 UDB EEE-Konfiguration besitzt ein Ausgangsverzeichnis, das aus einer Maschine über NFS exportiert wird und auf allen Maschinen, die an dem EEE-Exemplar beteiligt sind, angehängt wird. Bei einer Konfiguration zur gegenseitigen Übernahme ist DB2 UDB EEE davon abhängig, dass HA-NFS ein gemeinsam benutztes, hochverfügbares Dateisystem bereitstellt. Einer der logischen Hosts exportiert ein Dateisystem durch HA-NFS, und jede Maschine im Cluster hängt das Dateisystem als Ausgangsverzeichnis des EEE-Exemplars an. Weitere Informationen zu HA-NFS finden Sie in der Sun Cluster-Dokumentation.

Die Dienstprogramme cconsole und ctelnet

Zwei nützliche Dienstprogramme, die mit SC2.2 geliefert werden, sind `cconsole` und `ctelnet`. Diese Dienstprogramme können dazu verwendet werden, einen einzelnen Befehl an mehrere Maschinen in einem Cluster gleichzeitig abzusetzen. Wenn eine Konfigurationsdatei mit Hilfe dieser Dienst-

programme editiert wird, ist sichergestellt, dass sie auf allen Maschinen im Cluster gleich bleiben wird. Diese Dienstprogramme können außerdem dazu genutzt werden, Software in exakt der gleichen Weise auf jeder Maschine zu installieren. Weitere Informationen zu diesen Dienstprogrammen finden Sie in der Sun Cluster-Dokumentation.

Campus-Clustering und kontinentales Clustering

Ein Cluster wird als *Campus-Cluster* bezeichnet, wenn sich die zugehörigen Maschinen nicht im selben Gebäude befinden. Ein Campus-Cluster ist nützlich, um das Gebäude an sich als einzelnen Fehlerpunkt auszuschließen. Wenn sich zum Beispiel die Maschinen im Cluster alle im selben Gebäude befinden und das Gebäude abbrennt, ist der gesamte Cluster betroffen. Befinden sich die Maschinen hingegen in verschiedenen Gebäuden und ein Gebäude brennt ab, bleibt der Cluster bestehen.

Ein *kontinentaler Cluster* ist ein Cluster, dessen Maschinen über verschiedene Städte verteilt sind. In diesem Fall liegt das Ziel darin, die geographische Region als einzelnen Fehlerpunkt auszuschließen. Diese Art von Cluster bietet einen Schutz gegen Katastrophen wie Erdbeben und Flutwellen.

Zur Zeit kann Sun Cluster Maschinen unterstützen, die bis zu 10 km (ca. 6 Meilen) auseinander liegen. Dies macht ein Campus-Clustering zu einer funktionsfähigen Option für Benutzer, die Hochgeschwindigkeitsverbindungen zwischen zwei verschiedenen Standorten benötigen. Ein Cluster erfordert zwei private wechselseitige Verbindungen und eine Anzahl von Glasfaserkabeln für die gemeinsam genutzten Platten. Die Kosten von Hochgeschwindigkeitsverbindungen zwischen zwei Standorten könnten die Vorteile relativieren.

Häufige Probleme

Die SC2.2-Software verwendet Cluster Configuration Database bzw. CCD(4), um eine einziges clusterweites Repository für die Clusterkonfiguration bereitzustellen. CCD besitzt eine private API und ist unter dem Verzeichnis `/etc/opt/SUNWcluster/conf` gespeichert. In seltenen Fällen kann CCD die Synchronisierung verlieren und muss dann eventuell repariert werden. Die beste Methode zur Reparatur von CCD in diesem Fall ist die Wiederherstellung von CCD von einer Sicherungskopie.

Zur Sicherung von CCD fahren Sie die Clustersoftware auf allen Maschinen im Cluster herunter, komprimieren das Verzeichnis `/etc/opt/SUNWcluster/conf` mit "tar" und speichern die tar-Datei an einem sicheren Ort. Wenn die Clustersoftware bei der Erstellung der Sicherung nicht heruntergefahren ist, haben Sie vielleicht Schwierigkeiten bei der Wiederherstellung von CCD. Stellen Sie sicher, dass die Sicherungskopie auf aktuellem Stand gehalten wird, indem Sie die Sicherung nach jeder Änderung der Clusterkonfiguration aktualisieren. Zur Wiederherstellung von CCD fahren Sie die Clustersoftware auf allen Maschinen im Cluster herunter, versetzen das

Verzeichnis `conf` nach `conf.old` und dekomprimieren die Sicherungskopie. Der Cluster kann anschließend mit der wiederhergestellten CCD gestartet werden.

Überlegungen zu DB2

Dieser Abschnitt behandelt die folgenden Themen:

Anwendungen, die eine Verbindung zu einem HA-Exemplar herstellen

Anwendungen, die von einem hochverfügbaren DB2-Exemplar abhängig sind, müssen in der Lage sein, im Falle einer Funktionsübernahme eine neue Verbindung herzustellen. Da der Host-Name und die IP-Adresse eines logischen Host gleich bleiben, muss keine Verbindung zu einem anderen Host-Namen hergestellt oder die Datenbank erneut katalogisiert werden.

Betrachten Sie das Beispiel eines Clusters mit zwei Maschinen und einem Exemplar von DB2 Universal Database Enterprise Edition (EE). Das EE-Exemplar befindet sich im Normalfall auf einer der Maschinen im Cluster. Clients des HA-Exemplars stellen die Verbindung zur logischen IP-Adresse (oder zum Host-Namen) des logischen Hosts her, dem das HA-Exemplar zugeordnet ist.

Aus Sicht eines HA-Clients gibt es zwei Arten von Funktionsübernahme. Eine Art tritt auf, wenn die Maschine abstürzt, auf der das HA-Exemplar aktiv ist. Die andere Art tritt auf, wenn das HA-Exemplar eine Möglichkeit erhält, ordnungsgemäß herunterzufahren.

Wenn eine Maschine abstürzt und das HA-Exemplar außer Betrieb setzt, werden sowohl die vorhandenen Verbindungen als auch neue Verbindungen zur Datenbank blockiert. Die Verbindungen werden blockiert, weil es keine Maschinen im Netzwerk mit der IP-Adresse gibt, die die Clients für die Datenbank verwendeten. Wenn die Datenbank ordnungsgemäß heruntergefahren wird, unterbricht ein Befehl `db2stop force` vorhandene Verbindungen zur Datenbank und es wird eine Fehlernachricht zurückgegeben.

Während der Funktionsübernahme ist die der Datenbank zugeordnete logische IP-Adresse offline, entweder weil sie von der SC2.2-Software offline genommen wurde oder weil die Maschine, die den logischen Host enthielt, abgestürzt ist. Zu diesem Zeitpunkt werden alle neuen Verbindungen zur Datenbank für eine kurze Zeit blockiert.

Die der Datenbank zugeordnete logische IP-Adresse wird schließlich auf einer anderen Maschine wieder funktionsfähig gemacht, bevor DB2 gestartet wird. In diesem Stadium wird eine Verbindung zur Datenbank nicht blockiert, jedoch empfängt sie einen Kommunikationsfehler, weil DB2 im System noch nicht wieder gestartet wurde. DB2-Clients, die immer noch mit der Datenbank verbunden sind, erhalten nun ebenfalls Kommunikationsfehler. Obwohl die Clients weiterhin annehmen, dass sie verbunden sind, sind der Maschine, die die logische IP-Adresse übernommen hat, keine vorhandenen Verbindungen bekannt. Die Verbindungen werden einfach zurückgesetzt, und der DB2-Client empfängt einen Kommunikationsfehler. Nach einer kurzen Zeit wird DB2 auf der Maschine gestartet, so dass eine Verbindung zur Datenbank erfolgreich hergestellt werden kann. An diesem Punkt kann die Datenbank inkonsistent sein, und die Clients müssen möglicherweise warten, bis die Konsistenz wiederhergestellt ist.

Bei der Entwicklung einer Anwendung für eine HA-Umgebung ist es nicht nötig, speziellen Code für die Phasen zu schreiben, in denen die Datenbankverbindungen blockiert werden. Die Verbindungen sind nur kurze Zeit blockiert, während die Sun Cluster-Software die logische IP-Adresse versetzt. Jeder Datenbankservice, der unter Sun Cluster ausgeführt wird, erfährt die gleichen blockierten Verbindungen während dieser Phase. Unabhängig davon, wie die Datenbank außer Betrieb gesetzt wird, empfangen die Clients eine Fehlernachricht und müssen versuchen, die Verbindung wiederherzustellen, bis sie erfolgreich sind. Aus der Perspektive des Clients sieht es so aus, als wäre das HA-Exemplar ausgefallen und auf derselben Maschine wieder verfügbar gemacht worden. Bei einer kontrollierten Funktionsübernahme erscheint es dem Client so, als ob er zwangsweise getrennt wurde und später eine neue Verbindung zur Datenbank auf derselben Maschine herstellen könnte. Bei einer unkontrollierten Funktionsübernahme hat der Client den Eindruck, dass der Datenbankserver abgestürzt ist und auf derselben Maschine in kurzer Zeit wieder verfügbar gemacht wurde.

Plattenlayout für EE- und EEE-Exemplare

DB2 erwartet, dass die erforderlichen Platteneinheiten bzw. Dateisysteme auf jeder Maschine im Cluster gleich erscheinen. Um dies zu gewährleisten, sollten die erforderlichen Platten oder Dateisysteme so konfiguriert werden, dass sie dem logischen Host, dem das HA-Exemplar zugeordnet ist, folgen und auf jeder Maschine im Cluster die gleichen Pfadnamen besitzen.

Sowohl DMS- als auch SMS-Tabellenbereiche werden in einer HA-Umgebung unterstützt. Einheitenbehälter für DMS-Tabellenbereiche müssen vom Datenträgermanager erstellte unformatierte Einheiten (Raw Devices) verwenden, die entweder gespiegelt werden oder in einer RAID-Konfiguration eingerichtet sind. Reguläre Platteneinheiten, wie zum Beispiel `/dev/rdsk/c20t0d0s0`, sollten aus folgenden Gründen nicht eingesetzt werden:

- Sie erhöhen die Wahrscheinlichkeit, dass mehr als eine Maschine gleichzeitig auf die Einheiten schreiben.
- Die Controllernummer könnte auf einer anderen Maschine anders lauten.

Wenn DB2 in dieser Situation von einer anderen Maschine übernommen wird, sehen die erforderlichen Platteneinheiten nicht genauso aus, wie auf der anderen Maschine, und DB2 wird nicht gestartet. Dateibehälter für DMS-Tabellenbereiche und Behälter für SMS-Tabellenbereiche müssen sich auf angehängten Dateisystemen befinden. Die Dateisysteme für einen logischen Host werden automatisch angehängt, wenn sie in die Datei `vfstab` für den logischen Host aufgenommen wurden.

Die Datei `vfstab` für einen logischen Host befindet sich in folgendem Pfad:

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logischer-host>
```

Dabei ist *logischer-host* der Name des logischen Hosts, dem die Datei `vfstab` zugeordnet ist.

Jeder logische Host besitzt eine eigene Datei `vfstab`, die Dateisysteme enthält, die anzuhängen sind, nachdem die Plattengruppen für den logischen Host auf die aktuelle Maschine übertragen wurden, jedoch bevor die HA-Services gestartet werden. Die Sun Cluster-Software versucht, jedes ordnungsgemäß definierte Dateisystem nach der Ausführung von **fsck** (Dateisystemprüfung) anzuhängen, um den einwandfreien Zustand des Dateisystems sicherzustellen. Wenn **fsck** fehlschlägt, wird das Dateisystem nicht angehängt, und eine Fehlernachricht wird protokolliert.

Anmerkung: Wenn ein Prozess eine geöffnete Datei besitzt oder sich das zugehörige aktuelle Arbeitsverzeichnis unter einem Mount-Punkt befindet, schlägt das Anhängen fehl. Zur Vermeidung dieses Fehlers müssen Sie sicherstellen, dass keine Prozesse unter Mount-Punkten verbleiben, die in der Datei `vfstab` des logischen Hosts enthalten sind.

Für das Dateisystemlayout für ein EEE-Exemplar kann bei Verwendung von SMS-Tabellenbereichen eine beliebige Konvention verwendet werden. Das folgende Beispiel zeigt die Konvention, die vom Dienstprogramm hadb2_setup verwendet wird:

```
scadmin@crackle(190)# pwd
/export/ha_home/db2eee/db2eee
scadmin@crackle(191)# ls -l
total 18
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0000 -> /log0/disks/db2eee/NODE0000
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0001 -> /log0/disks/db2eee/NODE0001
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0002 -> /log0/disks/db2eee/NODE0002
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0003 -> /log0/disks/db2eee/NODE0003
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0004 -> /log0/disks/db2eee/NODE0004
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0005 -> /log1/disks/db2eee/NODE0005
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0006 -> /log1/disks/db2eee/NODE0006
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0007 -> /log1/disks/db2eee/NODE0007
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0008 -> /log1/disks/db2eee/NODE0008
scadmin@crackle(192)#
```

Der Exemplareigner ist `db2eee`, und das Standarddatenbankverzeichnis für das Exemplar `db2eee` ist `/export/ha_home/db2eee`. Der logische Host `log0` enthält die Datenbankpartitionen 0, 1, 2,3 und 4, während der logische Host `log1` die Datenbankpartitionen 5, 6, 7 und 8 enthält.

Für jede Datenbankpartition gibt es ein entsprechendes Verzeichnis `NODE`*xxxx*. Die Knotenverzeichnisse für die Datenbankpartitionen zeigen auf ein Verzeichnis unter dem Dateisystem des zugeordneten logischen Hosts.

Bei der Auswahl einer Pfadkonvention müssen Sie Folgendes sicherstellen:

1. Die Platten für das Dateisystem befinden sich in einer Plattengruppe des logischen Hosts, der für die Datenbankpartitionen zuständig ist, die sie benötigen.
2. Die Dateisysteme, die Behälter enthalten, werden über die Datei `vfstab` des logischen Hosts angehängt.

Ausgangsverzeichnis für EE- und EEE-Exemplare

Für ein EE-Exemplar sollte das Ausgangsverzeichnis ein Dateisystem sein, das in der Datei `vfstab` für einen logischen Host definiert ist. Dieses Verzeichnis wird verfügbar, bevor DB2 gestartet wird, und wird mit DB2 übertragen, wenn der logische Host im Cluster versetzt wird. Jede Maschine verfügt über eine eigene Kopie der Datei `vfstab`, und es sollte sorgfältig darauf geachtet werden, dass sie auf jeder Maschine im Cluster den gleichen Inhalt hat. Das folgende Beispiel zeigt das Ausgangsverzeichnis für ein EE-Exemplar:

```
/log0/home/db2ee
```

Dabei ist `/log0` das Dateisystem für den logischen Host `log0`, und `db2ee` ist der Name des DB2-Exemplars. Dieser Ausgangsverzeichnispfad sollte in die Datei `/etc/passwd` auf jeder Maschine im Cluster eingefügt werden, die als Host für das Exemplar "db2ee" fungieren kann.

Für ein EEE-Exemplar kann das Ausgangsverzeichnis auf zwei Arten eingerichtet werden. Bei einer Konfiguration im Bereitschaftsmodus kann das Aus-

gangsverzeichnis auf die gleiche Art wie für ein EE-Exemplar eingerichtet werden. Bei einer Konfiguration zur gegenseitigen Übernahme muss HA-NFS für das Ausgangsverzeichnis verwendet und *vor* dem Einrichten des EEE-Exemplars ordnungsgemäß konfiguriert werden.

Eine der Maschinen im Cluster muss mit Hilfe der Datei `dfstab` für einen ausgewählten logischen Host das Dateisystem für das EEE-Exemplar exportieren. Die Datei `dfstab` enthält Dateisysteme, die durch NFS exportiert werden sollten, wenn eine Maschine einen logischen Host enthält. Jede Maschine verfügt über eine eigene Kopie der Datei `dfstab`, und es sollte sorgfältig darauf geachtet werden, dass sie auf jeder Maschine im Cluster den gleichen Inhalt hat.

Informationen für das HA-NFS-Dateisystem werden in der Datei `hadb2tab` (durch das Programm `hadb2_setup`) abgelegt. Wenn ein HA-Agent die Informationen für das Exemplar liest, hängt er das HA-NFS-Dateisystem für das Exemplar automatisch an (siehe „Die Datei hadb2tab" auf Seite 314).

Der Mount-Punkt für das HA-NFS-Dateisystem ist in der Regel `/export/ha_home`. Auf jeder Maschine im Cluster würde dieses Verzeichnis vom logischen Host, der das HA-NFS-Verzeichnis exportiert, über NFS angehängt. Das Ausgangsverzeichnis des EEE-Exemplareigners wird unter diesem Verzeichnis angelegt und hat den folgenden Namen:

```
/export/ha_home/<exemplar>
```

Dabei ist *exemplar* der Name des Exemplareigners.

Es wäre möglich, ein Ausgangsverzeichnis für ein Exemplar auf jeder Maschine anzulegen, um ein Anhängen oder Abhängen dieses Verzeichnisses zu vermeiden. Dieses Verfahren erfordert einen zusätzlichen administrativen Aufwand, um sicherzustellen, dass die Ausgangsverzeichnisse auf allen Maschinen identisch bleiben. Wenn dies nicht gewährleistet wird, kann DB2 eventuell nicht ordnungsgemäß gestartet werden oder wird mit einer anderen Konfiguration gestartet. Dies ist *keine* unterstützte Konfiguration.

Logische Hosts und DB2 UDB EEE

Ein logischer Host wird in der Regel dazu ausgewählt, als Host für eine oder mehrere Datenbankpartitionen zu fungieren und das HA-NFS-Dateisystem zu exportieren. Wenn es zum Beispiel vier Datenbankpartitionen und zwei Maschinen im Cluster gäbe, sollte es einen logischen Host für jede Maschine geben (siehe Abb. 33 auf Seite 311). Ein logischer Host würde als Host zweier Datenbankpartitionen fungieren und das HA-NFS-Dateisystem exportieren, während der andere logische Host die verbleibenden zwei Datenbankpartitionen enthält.

Standardmäßig ordnet ein DB2 UDB EEE-Exemplar einer Maschine, die bereits eine oder mehrere im Betrieb befindliche Datenbankpartitionen für dieses Exemplar besitzt, genügend Ressourcen zum erfolgreichen Hinzufügen von bis zu zwei Datenbankpartitionen zu. Wenn es beispielsweise vier Datenbankpartitionen für ein einziges Exemplar in einem Cluster gibt, kann es nur dann zu Problemen kommen, wenn es eine Datenbankpartition pro logischen Host gibt oder ein logischer Host als Host von drei Datenbankenpartitionen fungiert. In beiden Fällen ist es möglich, dass drei Datenbankpartitionen von einer Maschine übernommen werden, die bereits für eine Datenbankpartition desselben Exemplars zuständig ist.

Die Registrierungsvariable DB2_NUM_FAILOVER_NODES kann dazu verwendet werden, die Ressourcenmenge zu erhöhen, die für zu übernehmende Datenbankpartitionen reserviert wird.

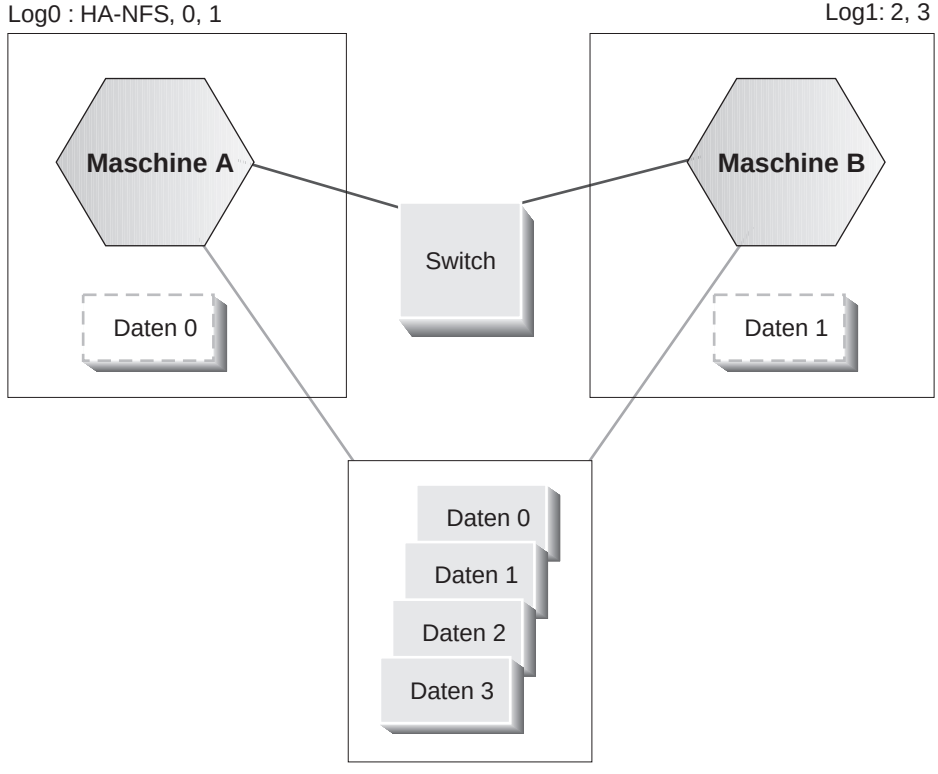


Abbildung 33. Ein logischer Host für jede Maschine

Speicherposition und Optionen der DB2-Installation

Das Dateisystem, in dem DB2 installiert wird, sollte gespiegelt werden oder zumindest in einer RAID-Konfiguration angelegt werden. Wenn DB2 auf normalen Platten installiert wird, ist ein Plattenausfall wahrscheinlicher. Die sich daraus ergebende Funktionsübernahme gilt als vermeidbar und verringert die Stabilität des Clusters.

DB2 kann nicht auf Platten in einer Plattengruppe für einen logischen Host installiert werden, weil der HA-Agent ständig Zugriff auf die DB2-Bibliotheken benötigt. Wenn die HA-Agenten keinen Zugriff auf die DB2-Bibliotheken haben, werden sie mit einem Fehler beendet. DB2 muss auf jeder Maschine im Cluster normal installiert werden.

Konfigurationsparameter für die Datenbank und den Datenbankmanager

Die Konfigurationsparameter des Datenbankmanagers können nach einer Funktionsübernahme und vor dem Starten von DB2 mit Hilfe des Scripts `pre_db2start` (siehe „Benutzer-Scripts" auf Seite 316) geändert werden. Dieses ausführbare Script wird (falls vorhanden) unter dem Verzeichnis `sqllib/ha` des Ausgangsverzeichnisses des Exemplareigners ausgeführt. Wie der Name verrät, wird es unmittelbar vor **db2start** ausgeführt. Die gleichen Parameter, die an die Steuermethoden übergeben werden, werden auch an das Script `pre_db2start` übergeben, wenn das Exemplar kein EEE-Exemplar ist. Für ein EEE-Exemplar wird dem Script `pre_db2start` außerdem die Knotennummer für den Befehl **db2start** übergeben.

Wiederherstellung nach einem Systemabsturz

Die Wiederherstellung nach einem Systemabsturz in einer HA-Umgebung erfolgt auf dieselbe Weise wie in einer regulären Umgebung. Selbst wenn das HA-Exemplar auf einer anderen Maschine als der Absturzmaschine wieder gestartet wird, sehen die Dateien und Platteneinheiten für das Exemplar gleich aus und die Aktionen, die zur Wiederherstellung der Datenbank durchzuführen sind, unterscheiden sich ebenfalls nicht. Weitere Informationen zur Wiederherstellung nach einem Systemabsturz und zu anderen Arten der Datenbankwiederherstellung finden Sie in „Kapitel 1. Entwickeln einer guten Sicherungs- und Wiederherstellungsstrategie" auf Seite 3.

Obwohl eine Datenbank manuell (oder durch eines der Benutzer-Scripts) erneut gestartet werden kann, wird empfohlen, den Datenbankkonfigurationsparameter *autorestart*, insbesondere für ein EEE-Exemplar, auf den Wert `ON` zu setzen. Dadurch wird der Zeitraum minimiert, den die Datenbank in einem inkonsistenten Zustand ist.

Hohe Verfügbarkeit durch Datenreplikation

Die Verfügbarkeit von Daten kann auch durch Replikation verbessert werden. Durch das Replizieren von Daten zwischen zwei Servern lässt sich eine Form

hoher Verfügbarkeit erreichen. Wenn einer der Server ausfällt, kann der andere Server die Funktion übernehmen und den Datenbankservice weiterhin bereitstellen.

Da die Replikation jedoch asynchron erfolgt, wurden von einem Server, der ausfällt, möglicherweise einige Änderungen noch nicht an den anderen Server weitergegeben.

Vorbedingungen zu DB2 Connect für Sun Cluster 2.2

DB2 Connect wird von Sun Cluster 2.2 unter den folgenden Bedingungen unterstützt:

- Das Protokoll für den Host ist TCP/IP (nicht SNA).
- Die zweiphasige Festschreibung wird nicht verwendet. Diese Rahmenbedingung gilt nicht, wenn der Benutzer das SPM-Protokoll so konfiguriert, dass es sich auf einem gemeinsam benutzten Datenträger befindet (dies erfolgt über den Konfigurationsparameter *spm_log_path* des Datenbankmanagers) und dass die Maschine für die Funktionsübernahme eine identische TCP/IP-Konfiguration aufweist (gleicher Host-Name, gleiche IP-Adresse usw.).

Der DB2-Agent für hohe Verfügbarkeit

Der DB2-Agent für hohe Verfügbarkeit (HA-Agent) fungiert als Mittler zwischen DB2 und SC2.x. Er bietet der Sun Cluster 2.2-Software eine Möglichkeit, DB2 in einer Clusterumgebung zu steuern, ohne sich mit den Einzelheiten von DB2 auskennen zu müssen. Es gibt einen Agenten für EE- und EEE-Exemplare. Der Agent unterstützt sowohl administrative Exemplare als auch Datenbankexemplare.

Registrieren des Service hadb2

Zur Arbeit mit SC2.2 muss der DB2-HA-Agent registriert werden. Durch die Registrierung eines Datenbankservice wird SC2.2 mitgeteilt, welche Steuermethoden verfügbar sind und in welchem Verzeichnis sie sich befinden. Mit dem HA-Agenten wird ein spezielles Script namens `hadb2_reg` geliefert, das den Service `hadb2` für EE- und EEE-Exemplare registrieren kann. Das Script `hadb2_reg` muss für den gesamten Cluster nur einmal ausgeführt werden.

Obwohl es nur einen Satz an Steuermethoden für den DB2-HA-Agenten gibt, hängt die Art, wie die Methoden registriert werden davon ab, ob ein EEE-Exemplar in einer Konfiguration zur gegenseitigen Übernahme verwendet wird oder nicht. Für ein EE-Exemplar bzw. ein EEE-Exemplar in einer Konfiguration im Bereitschaftsmodus wird HA-NFS nicht verwendet. Daher wird der Schalter `"-d nfs"` nicht benötigt, der der SC2.2-Software mitteilt, dass der Service `hadb2` von HA-NFS abhängig ist.

Der tatsächliche Befehl, der von hadb2_reg zur Registrierung der Steuermethoden für ein EEE-Exemplar der DB2 Version 7.1 verwendet wird, sieht wie folgt aus:

```
hareg -r hadb2 -b /opt/IBMdb2/V7.1/ha -m
START=hadb2_start,START_NET=hadb2_startnet,STOP_NET=hadb2_stopnet,
FM_START=hadb2_fmstart,FM_STOP=hadb2_fmstop
-t START_NET=$TIMEOUT,STOP_NET=$TIMEOUT -d nfs
```

Der Schalter -b weist SC2.x an, alle Steuermethoden im Verzeichnis opt/IBMdb2/V7.1/ha zu suchen. Der Schalter -m definiert die eigentlichen Steuermethoden für den Service hadb2. Der Schalter -t definiert das Zeitlimit für die Steuermethoden START_NET und STOP_NET. Eine detaillierte Beschreibung der einzelnen Steuermethoden finden Sie in der Sun Cluster-Dokumentation.

Das Script hadb2_unreg kann dazu genutzt werden, die Registrierung des Service hadb2 zurückzunehmen, und muss, wie hadb2_reg, nur einmal für den Cluster ausgeführt werden.

Die Datei hadb2tab

Die Datei hadb2tab ist die Hauptkonfigurationsdatei für den DB2-HA-Agenten. Jede Steuermethode greift auf diese Datei zurück, um zu ermitteln, welche Exemplare hochverfügbar sind. Die Datei hadb2tab befindet sich für DB2 UDB Version 7.1 im Verzeichnis /var/db2/v71/. Die Datei unterstützt mehrere Exemplare, und jede Nichtkommentarzeile stellt ein anderes HA-Exemplar dar. Das folgende Beispiel zeigt eine Datei hadb2tab:

```
<scadmin@thrash(203)# cat hadb2tab
EEE DATA db2eee jolt ON /export/ha_home /log0/home #Added by DB2 HA software
EE ADMIN db2ee log1 ON  -               -          #Added by DB2 HA software
```

Das erste Feld gibt dem DB2-HA-Agenten an, ob es sich bei dem Exemplar um ein EE- oder ein EEE-Exemplar handelt. Das zweite Feld gibt an, ob das Exemplar ein Datenexemplar oder ein administratives Exemplar ist. Das dritte Feld enthält den Benutzernamen für das HA-Exemplar. Das vierte Feld ist der logische Host oder der HA-NFS-Host für das Exemplar, je nachdem, ob es sich um ein EE- oder ein EEE-Exemplar handelt. Das fünfte Feld gibt an, ob die Fehlerüberwachung für das Exemplar aktiviert ist (ON) oder nicht (OFF). Die beiden letzten Felder geben den lokalen Mount-Punkt bzw. das ferne HA-NFS-Verzeichnis an. Diese Felder sollten mit dem Wert "-" (Bindestrich) gefüllt werden, wenn sie nicht verwendet werden, und sollten außerdem nur bei einer EEE-Konfiguration zur gegenseitigen Übernahme verwendet werden. Kommentare sind in der Datei hadb2tab zulässig, wenn die Informationen in der Zeile vor einem "#"-Zeichen entweder die Länge null haben oder eine gültige Definition eines Exemplars darstellen.

Steuermethoden

Steuermethoden für SC2.2-Agenten können eine Gruppe von Scripts oder Programmen sein. Der Agent für DB2 unter Solaris besteht aus einer Gruppe von Programmen, die folgende Methoden enthält:

START_NET
hadb2_startnet dient zum Starten von DB2.

STOP_NET
hadb2_stopnet dient zum Stoppen von DB2.

FM_START
hadb2_fmstart dient zum Starten des Fehlermonitors für DB2.

FM_STOP
hadb2_fmstop dient zum Stoppen des Fehlermonitors für DB2.

Weitere Informationen zu diesen Steuermethoden finden Sie in der Sun Cluster-Dokumentation.

Für EE-Exemplare wird der logische Host, der dem Exemplar zugeordnet ist, direkt in der Datei `hadb2tab` definiert. Für EEE-Exemplare muss die Steuermethode jedoch außerdem in folgende Datei schauen:

```
~<exemplar>/sqllib/ha/hadb2-eee.cfg
```

Dabei ist `~<exemplar>` das Ausgangsverzeichnis des Exemplareigners. Diese Datei enthält eine Zeile für jede Datenbankpartition und dient zur Zuordnung von Datenbankpartitionen zu logischen Hosts. Ein Beispiel für eine gültige Datei `hadb2-eee.cfg` sieht wie folgt aus:

```
crackle % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

Das Exemplar oder die Datenbankpartitionen folgen dem entsprechenden logischen Host von Maschine zu Maschine im Cluster. Der logische Host kann auf eine beliebige Maschine im Cluster versetzt werden, die von der zu Grunde liegenden Hardware und SC2.2 unterstützt wird. Wenn die Konfiguration ordnungsgemäß eingerichtet ist, unterstützt DB2 jede Topologie, die von der SC2.2-Software unterstützt wird.

Nach dem Lesen aller Informationen für ein Exemplar weiß die Steuermethode, welche logischen Hosts dem Exemplar zugeordnet sind. Nach der Syntaxanalyse der Befehlszeilenparameter weiß die Steuermethode außerdem, welche logischen Hosts sich auf der aktuellen Maschine befinden und welche nicht.

Die folgende Tabelle zeigt die Aktionen, die unternommen werden, je nachdem, welche Steuermethode ausgeführt wird, und abhängig davon, ob die der Datenbankpartition oder dem Exemplar zugeordneten logischen Hosts sich auf der aktuellen Maschine befinden.

Steuermethode	Zugeordnete logische Hosts auf Maschine	Zugeordnete logische Hosts nicht auf Maschine
START_NET	Starten des DB2-Exemplars oder der Datenbankpartitionen	Keine Aktion
STOP_NET	Keine Aktion	Stoppen des DB2-Exemplars oder der Datenbankpartitionen
FM_START	Starten des Fehlermonitors für das Exemplar	Keine Aktion
FM_STOP	Keine Aktion	Stoppen des Fehlermonitors für das Exemplar

Die Steuermethoden, die START-Aktionen ausführen, beziehen sich nur auf die logischen Hosts, die zur Zeit auf der Maschine aktiv sind, und die Steuermethoden, die STOP-Aktionen ausführen, beziehen sich nur auf logische Hosts, nicht zur Zeit nicht auf der Maschine aktiv sind.

Die Steuermethoden müssen darüber hinaus das HA-NFS-Verzeichnis auf spezielle Weise anhängen, wenn HA-NFS verwendet wird. Wenn der lokale Mount-Punkt und das Verzeichnis für HA-NFS nicht als "-" (Bindestrich) definiert sind, führt die Steuermethode einen Befehl `statvfs(2)` auf dem lokalen Mount-Punkt aus. Wenn der Dateisystemtyp für den lokalen Mount-Punkt nicht `nfs` ist, versucht der Agent, das Dateisystem mit Hilfe der Informationen aus der entsprechenden Zeile in der Datei `hadb2tab` anzuhängen. Wenn der Mount-Punkt und das Verzeichnis für HA-NFS als "-" (Bindestrich) definiert sind, ist die Datei `vfstab` für den entsprechenden logischen Host erforderlich, um das Dateisystem anzuhängen, das das Ausgangsverzeichnis für das Exemplar enthält. Der lokale Mount-Punkt und das ferne Verzeichnis für HA-NFS sollten nur für EE- und EEE-Konfigurationen im Bereitschaftsmodus als "-" (Bindestrich) definiert werden.

Benutzer-Scripts

Diese Scripts werden über die Steuermethoden ausgeführt, um zusätzliche Funktionalität bereitzustellen. Sie erhalten die gleichen Befehlszeilenparameter wie die Steuermethoden und werden vom System- bzw. vom Datenbankadministrator geschrieben.

Wenn ein Programm innerhalb eines Scripts ausgeführt werden muss, das nicht im Hintergrund ausgeführt wird, kann es sinnvoll sein, das Programm

mit nohup(1) in den Hintergrund zu verlegen. Das Programm nohup schützt das ausgeführte Programm vor dem Signal SIGHUP, d. h. vor einem nicht-programmierten Stopp. Ohne nohup kann ein Programm, das über ein Script im Hintergrund ausgeführt wird, infolge eines SIGHUP-Signals vorzeitig abgebrochen werden, wenn das Script die Verarbeitung beendet hat.

Die Steuermethoden führen die folgenden Scripts aus:

- /var/db2/v61/failover
- ~<exemplar>/sqllib/ha/pre_db2start
- ~<exemplar>/sqllib/ha/post_db2start
- ~<exemplar>%s/sqllib/ha/post_failover
- ~<exemplar>/sqllib/ha/pre_db2stop
- ~<exemplar>/sqllib/ha/fm_warning

Dabei ist *~exemplar* das Ausgangsverzeichnis des HA-Exemplars.

Mit Ausnahme des Scripts fm_warning wird jedes Benutzer-Script mit den gleichen Befehlszeilenparametern ausgeführt wie die Methode, die sie aufruft. Bei Verwendung von EEE-Exemplaren wird ferner die Nummer der Datenbankpartition (als letzter Parameter) an das Benutzer-Script übergeben.

Das Script /var/db2/v71/failover wird zu Beginn der Methode START_NET aufgerufen und im Hintergrund ausgeführt. Ein solches Script kann beispielsweise dazu genutzt werden, das Unterstützungspersonal per E-Mail über eine Funktionsübernahme in Kenntnis zu setzen. Das folgende Beispiel zeigt ein Failover-Script:

```
#!/bin/ksh

# Nachricht über Funktionsübernahme an Personal per E-mail oder Pager

echo "Funktionsübernahme auf Maschine 'hostname':Running $0!"
|/bin/mail admin@sphere.torolab.ibm.com
```

Zur erfolgreichen Benachrichtigung durch ein Script per E-Mail muss sendmail(1m) auf dem System ordnungsgemäß konfiguriert sein.

Wie der Name andeutet, wird das Script pre_db2start unmittelbar vor dem Aufrufen von **db2start** ausgeführt. Dieses Script kann zu solchen Zwecken wie dem Ändern von Konfigurationsparametern des Datenbankmanagers eingesetzt werden. Ihm werden maximal 20 Sekunden für die Ausführung eingeräumt. Für EEE-Exemplare wird dieses Script vor dem Aufrufen von **db2start** auf jeder Datenbankpartition ausgeführt. Dieses Script ist nur für Datenexemplare, nicht für administrative Exemplare von Bedeutung.

Analog wird das Script post_db2start unmittelbar *nach* dem Aufrufen von **db2start** ausgeführt. Dieses Script kann zu solchen Zwecken wie dem erneuten Starten von Datenbanken eingesetzt werden. Es wird im Hintergrund ausgeführt, um zu gewährleisten, dass seine Ausführungszeit keine störenden Auswirkungen auf andere Exemplare hat. Dieses Script ist nur für Datenexemplare, nicht für administrative Exemplare von Bedeutung.

Das Script post_failover im Ausgangsverzeichnis des Exemplareigners wird nach der Verarbeitung des Exemplars ausgeführt. Dieses Script kann dazu verwendet werden, Client-Anwendungen zu benachrichtigen, dass DB2 nun betriebsbereit ist, Datenbanken zu aktivieren oder Administratoren eine Statusdatei zu senden. Es wird im Hintergrund ausgeführt, um zu verhindern, dass seine Ausführungszeit Aktionen an anderen HA-Exemplaren verzögert. Das folgende Beispiel zeigt ein post_failover-Script:

```
#!/bin/ksh
#
# Statusdatei an Administrator senden.
mail admin@sphere.torolab.ibm.com </tmp/HA.info.db2eee
```

Sowohl die Methode START_NET als auch die Methode STOP_NET des DB2-HA-Agenten erstellen nach der Verarbeitung jedes Exemplars eine Statusdatei. Der Name der Statusdatei lautet:

```
/tmp/HA.info.<exemplar>
```

Dabei ist *exemplar* der Benutzername des Exemplareigners. Die Statusdatei enthält den Start- und Stoppbericht für das Exemplar sowie die Zeit, die zur Ausführung der Steuermethode benötigt wurde. Das folgende Beispiel zeigt eine Statusdatei:

```
scadmin@crackle(173)# cat /tmp/HA.info.db2eee
-----  Elapsed Time: 00:00:18           -----
-----  Elapsed Time: 00:00:00  (HA-NFS) -----

NODE     ACTION     RESULT     TRIES     RC
----     ------     ------     -----     --
   4     stop       success        3    1064
   5     stop       success        1    1064
   6     stop       success        2    1064
   7     stop       success        2    1064
   8     stop       success        1    1064
-----------------------------------------------
```

Das Script pre_db2stop wird unmittelbar vor dem Aufrufen von **db2stop** ausgeführt. Dieses Script kann dazu verwendet werden, Client-Anwendung darauf hinzuweisen, dass DB2 gestoppt wird. Ihm werden maximal 20 Sekunden für die Ausführung eingeräumt. Dieses Script ist nur für Datenexemplare, nicht für administrative Exemplare von Bedeutung.

Der Fehlermonitor (Fault Monitor) führt ebenfalls ein Benutzer-Script aus, wenn DB2 aufgrund eines unerwarteten Herunterfahrens erneut gestartet wurde. Dieses Script heißt:

```
~<exemplar>/sqllib/ha/fm_warning
```

Das Script `fm_warning` kann dazu genutzt werden, den Systemadministrator zu benachrichtigen, dass DB2 durch den Fehlermonitor erneut gestartet wurde. Der Systemadministrator sollte versuchen, herauszufinden, warum DB2 unerwartet heruntergefahren wurde, und die entsprechenden Maßnahmen ergreifen, um dieses in Zukunft zu vermeiden. Das Script `fm_warning` wird im Hintergrund ausgeführt.

Weitere Überlegungen

Wenn ein HA-Datenbankservice ausgeschaltet wird, werden während einer Funktionsübernahme oder einer Clusterrekonfiguration nur die STOP-Methoden ausgeführt. Die übrigen Methoden werden nur ausgeführt, wenn der HA-Datenbankservice ordnungsgemäß registriert und aktiviert ist.

Stellen Sie sicher, dass jede Maschine im Cluster über ausreichend Ressourcen verfügt, um alle Datenbankservices auszuführen, für die sie gegebenenfalls zuständig ist. Ressourcen wie CPU-Auslastung, Arbeitsspeicher, Auslagerungs- und Kernel-Parameter müssen bedacht werden, bevor der Cluster als Geschäftssystem eingesetzt wird. Wenn zum Beispiel eine Maschine im Cluster zwei DB2-Exemplare ausführen muss, sind die Kernel-Parameteranforderungen für diese Maschine die Summe der Ressourcen, die für jedes Exemplar erforderlich sind.

Fehlermonitor

Wenn die Fehlerüberwachung aktiviert ist, wird der Fehlermonitor bei einer Clusterrekonfiguration oder bei einer Funktionsübernahme gestartet. Falls DB2 nicht durch das Script START_NET gestartet wird, startet der Fehlermonitor DB2 selbst. Der Fehlermonitor kann erkennen, ob DB2 nicht gestartet wurde oder ob DB2 aus unbekannten Gründen heruntergefahren wurde. Daher ist es wichtig, DB2 nicht manuell herunterzufahren, wenn der Fehlermonitor aktiviert ist. Der Fehlermonitor sieht dies als unerwartetes Herunterfahren an und startet DB2 erneut. Wenn dies zu häufig geschieht, findet eine Funktionsübernahme des entsprechenden logischen Hosts durch eine andere Maschine statt.

Wenn die Fehlerüberwachung für ein Exemplar aktiviert ist, besteht die korrekte Art zum manuellen Starten oder Stoppen des Exemplars darin, zunächst die Fehlerüberwachung oder den Service `hadb2` zu inaktivieren. Beide Aktionen können durch den Befehl **hadb2_setup** mit den Schaltern `-f` und `-s` eingeleitet werden (siehe „Der Befehl hadb2_setup" auf Seite 325).

Anmerkung: Verwenden Sie nicht mehr als ein Exemplar für den gleichen logischen Host. Wenn einem logischen Host mehr als ein Exemplar zugeordnet wird, kann das einwandfreie Exemplar zusammen mit dem defekten Exemplar bei einer Funktionsübernahme versetzt werden.

Überlegungen zu EEE

Bei der Entscheidung, welche Datenbankpartitionen einem logischen Host zuzuordnen sind, spielt die Art und Weise, wie die Funktionsübernahme stattfindet, eine wichtige Rolle. Betrachten Sie zum Beispiel einen Cluster mit zwei Maschinen, der mit vier Datenbankpartitionen zwischen zwei Maschinen eingesetzt werden soll, wie in Abb. 34 zu sehen ist.

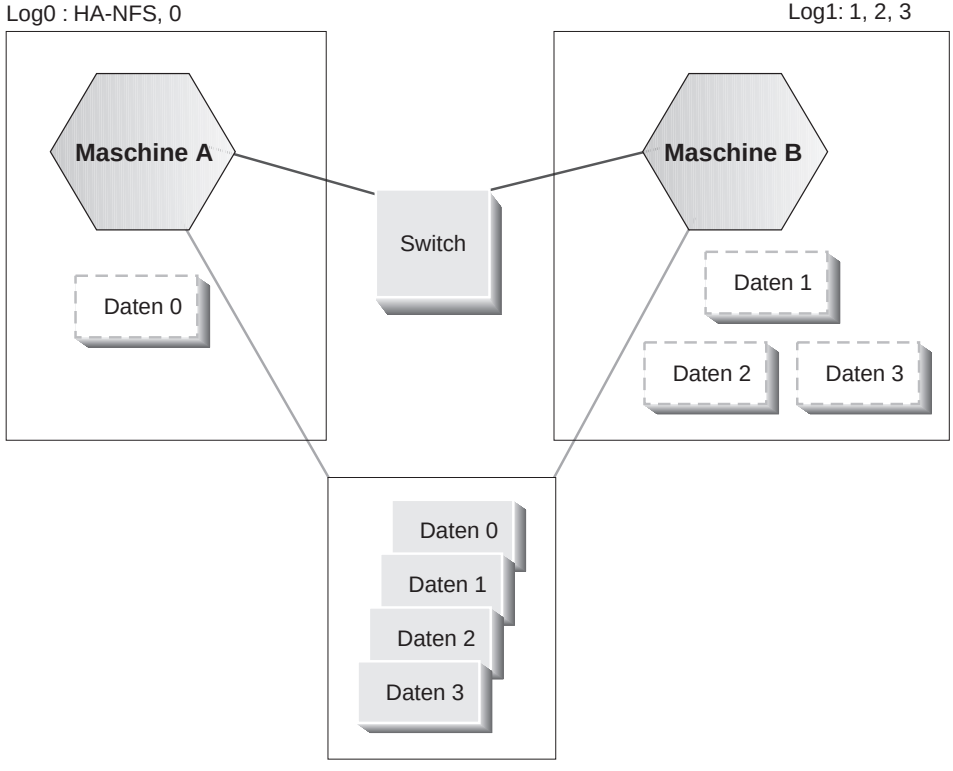


Abbildung 34. Cluster mit zwei Maschinen und vier Datenbankpartitionen

Sie könnten jede Datenbankpartition je einem logischen Host und HA-NFS ebenfalls einem logischen Host zuordnen. In diesem Fall kann ein Problem entstehen, falls alle logischen Hosts in einem System enthalten sind. Wenn ein System ausfällt, müssen alle logischen Hosts gleichzeitig von diesem System versetzt werden. Leider versetzt die Sun Cluster-Software die logischen Hosts in keiner vorhersagbaren Reihenfolge, so dass es möglich ist, dass ein logi-

scher Host für eine Datenbankpartition versetzt wird, bevor der logische Host mit HA-NFS versetzt wird. In der Regel ist es ein gutes Verfahren, Datenbankpartitionen je nachdem, was gewöhnlich auf einem einzelnen System untergebracht würde, zusammen zu gruppieren. Das bedeutet, dass zwei Datenbankpartitionen, die unter normalen Bedingungen auf einer Maschine untergebracht sind, einem einzigen logischen Host zugeordnet werden sollten.

Die Datei `db2nodes.cfg`, die von einem EEE-Exemplar verwendet wird, wird aktualisiert, um die Maschine anzugeben, auf der sich die Datenbankpartitionen befinden. Wenn zum Beispiel alle Datenbankpartitionen auf einer Maschine namens "crackle" untergebracht sind, sieht die Datei `db2nodes.cfg` in etwa wie folgt aus:

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 crackle 4 204.152.65.33
5 crackle 5 204.152.65.33
6 crackle 6 204.152.65.33
7 crackle 7 204.152.65.33
8 crackle 8 204.152.65.33
```

Wenn einige dieser Datenbankpartitionen auf eine Maschine namens "thrash" versetzt werden, wird die Datei `db2nodes.cfg` wie folgt aktualisiert:

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 thrash 0 204.152.65.34
5 thrash 1 204.152.65.34
6 thrash 2 204.152.65.34
7 thrash 3 204.152.65.34
8 thrash 4 204.152.65.34
```

Beachten Sie, dass sowohl der Host-Name als auch der Switch-Name geändert werden, um den Maschinennamen "thrash" anzugeben, und dass die Port-Nummern ebenfalls unterschiedlich sind.

Die Datei HA.config

Wenn sie vorhanden ist, kann die Datei `/etc/HA.config` eine Reihe von Konfigurationsoptionen, einschließlich der folgenden, enthalten:

```
scadmin@thrash(204)# cat /etc/HA.config
SYSLOG_FACILITY=LOG_LOCAL3
SYSLOG_LPRIORITY=LOG_INFO
SYSLOG_EPRIORITY=LOG_ERR
USE_INTERCONNECT=auto
SWITCH_NAME=204.152.65.18
DEBUG_LEVEL=2
```

```
FAILS_PER_HOUR=2
FAILS_PER_DAY=4
FAILS_PER_WEEK=10
FM_FAIL_SEV=soft
DB2START_TIMEOUT=60
DB2STOP_TIMEOUT=500
SCRIPT_USER=bin
```

Anmerkung: Wenn die Datei `HA.config` nicht vorhanden ist, werden Standardwerte verwendet.

Die Variable SYSLOG_FACILITY stellt die SYSLOG-Einrichtung auf das Protokollieren von Nachrichten und Fehlern ein. Die Variablen SYSLOG_LPRIORITY und SYSLOG_EPRIORITY stellen die SYSLOG-Priorität auf das Protokollieren von Informationsnachrichten bzw. Fehlernachrichten ein.

Einige Änderungen sind zur Aktivierung des SYSLOG-Dämons zur Protokollierung von Informationen aus dem DB2-HA-Agenten eventuell erforderlich. Zum Beispiel wird durch Einfügen einer der folgenden beiden Zeilen in der Datei `/etc/syslog.conf` der SYSLOG-Dämon angewiesen, Informationen in eine Protokolldatei zu schreiben.

```
*.notice                                    /var/adm/SC.x
local3.info                                 /var/adm/SC.LOG_LOCAL3
```

Ein Sun Cluster verfügt in der Regel über eine wechselseitige Hochgeschwindigkeitsverbindung. Um die Hochgeschwindigkeitsverbindung mit DB2 zu nutzen, muss USE_INTERCONNECT auf den Wert `auto` oder `override` gesetzt werden. Die Einstellung `auto` (Standardeinstellung) arbeitet mit der internen logischen Sun-Netzwerkschnittstelle. Diese Schnittstelle wird auf eine andere physische Schnittstelle übertragen, wenn die erste Schnittstelle ausfällt. Wenn USE_INTERCONNECT auf den Wert `override` gesetzt ist, wird der Switch-Name der Variablen SWITCH_NAME entnommen. Eine weitere Option besteht darin, USE_INTERCONNECT auf `no` zu setzen, wodurch definiert wird, dass die Hochgeschwindigkeitsverbindung nicht zu verwenden ist.

Die Variable DEBUG_LEVEL gibt an, wie viele Informationen bei einer Funktionsübernahme zu protokollieren sind. Der Wert ist eine Zahl zwischen 0 und 10, wobei 10 die höchste Debug-Stufe darstellt. Die Informationen werden mit der angegebenen SYSLOG-Priorität und der angegebenen SYSLOG-Einrichtung protokolliert. Wenn Probleme auftreten, setzen Sie die Debug-Stufe auf den höchsten Wert, konfigurieren SYSLOG so, dass die Ausgabe der HA-Agenten protokolliert wird und senden die SYSLOG-Ausgabe an den IBM Kundendienst.

Drei der Variablen unterstützen den DB2-Fehlermonitor bei der Entscheidung, wann eine Funktionsübernahme für einen logischen Host durchzuführen ist: FAILS_PER_HOUR, FAILS_PER_DAY und FAILS_PER_WEEK. Jede HA-Um-

gebung ist unterschiedlich. Sie müssen entscheiden, wie viele DB2-Ausfälle akzeptabel sind. Nach jedem akzeptablen Ausfall wird DB2 auf der gleichen Maschine wieder gestartet. Wenn einer dieser drei Schwellenwerte für Ausfälle überschritten wird, wird die Funktion des logische Hosts, dem das Exemplar oder die Datenbankpartition zugeordnet ist, von einer anderen Maschine übernommen.

Die Variable FM_FAIL_SEV gibt an, ob die Funktionsübernahme ″soft″ oder ″hard″ ist. Weitere Informationen finden Sie in der Sun Cluster-Dokumentation über `hactl(1m)`.

Die Variablen DB2START_TIMEOUT und DB2STOP_TIMEOUT geben die maximale Anzahl von Sekunden an, die **db2start** und **db2stop** ausgeführt werden dürfen. Wenn das angegebene Intervall abgelaufen ist, betrachtet der HA-Agent die Operation als fehlgeschlagen und versucht, das Exemplar erneut zu starten.

Es gibt einige Benutzer-Scripts, die nicht einem bestimmten Exemplar zugeordnet sind. Normalerweise werden diese Scripts mit Root-Berechtigung ausgeführt. Dies kann mit Hilfe der Variablen SCRIPT_USER, mit der die Benutzer-ID angegeben wird, die diese Scripts ausführen kann, überschrieben werden.

Ausführen von DB2-Befehlen durch Steuermethoden

Der DB2-HA-Agent verwendet den Befehl **su**, um Befehle als Exemplareigner auszuführen. Der tatsächliche Befehl könnte in etwa wie folgt aussehen:

```
su - <exemplar> -c "db2stop"
```

Dabei ist *exemplar* der Benutzername des Exemplars.

Es ist wichtig, sicherzustellen, dass die Datei `.profile` des Exemplareigners **su**-″freundlich″ ist. Wenn dies nicht der Fall ist, funktioniert der Befehl **su** eventuell nicht richtig. Rufen Sie den Befehl **su** manuell oder aus einem Script auf, um zu prüfen, ob der Befehl erfolgreich ausgeführt wird.

Installation und Konfiguration

Bevor Sie diesen Abschnitt lesen, machen Sie sich mit der SC2.2-Software vertraut. Dieser Abschnitt setzt voraus, dass Sie wissen, wie SC2.2 und HA-NFS eingerichtet wird, und dass Sie wissen, wie der Datenträgermanager (Volume Manager) verwendet wird. Neben den anderen erforderlichen Programmkorrekturen für DB2 sind für den HA-Agenten die folgenden Programmkorrekturen erforderlich:

```
Solaris 2.6:
   105210-17 (oder höher)
   105786-05 (oder höher)
```

Anmerkung: Für Solaris 7 (Solaris 2.7) sind keine Programmkorrekturen erforderlich.

Allgemeine Installationsschritte

1. Installieren Sie SC2.2 auf allen Maschinen im Cluster. Während der Installation fragt SC2.2, welche Agenten zu installieren sind. Da DB2 nicht mit SC2.2 ausgeliefert wird, ist DB2 nicht in der Agentenliste enthalten. Der Agent für DB2 wird mit DB2 installiert und über den Befehl **hadb2_reg** registriert.
2. Konfigurieren Sie die logischen Hosts mit Plattengruppen und logischen IP-Adressen.

Installation und Konfiguration von DB2 UDB Enterprise Edition

1. Erstellen Sie das Ausgangsverzeichnis für das Exemplar unter dem Dateisystem eines logischen Hosts.
2. Installieren Sie DB2 auf allen Maschinen im Cluster.
3. Erstellen Sie das Exemplar auf der Maschine im Cluster, auf dem sich zurzeit das Ausgangsverzeichnis für das Exemplar befindet.
4. Fügen Sie den Benutzer für das Exemplar den anderen Maschinen im Cluster hinzu, wobei sicherzustellen ist, dass die numerische Benutzer-ID identisch ist.
5. Registrieren Sie den Service `hadb2` mit dem Befehl **hadb2_reg**.
6. Führen Sie den Befehl **hadb2_setup** aus, um HA für das Exemplar einzurichten.

Installation und Konfiguration von DB2 UDB Enterprise - Extended Edition

1. Erstellen Sie das Ausgangsverzeichnis für den HA-Exemplareigner:
   a. Für den Bereitschaftsmodus erstellen Sie das Ausgangsverzeichnis für das Exemplar unter dem Dateisystem eines logischen Hosts.
   b. Für gegenseitige Übernahme konfigurieren Sie HA-NFS und exportieren das Ausgangsverzeichnis von einem der logischen Hosts. Hängen Sie auf einer der Maschinen das HA-NFS-Verzeichnis unter dem gewählten Mount-Punkt an.
2. Installieren Sie DB2 auf allen Maschinen im Cluster.
3. Erstellen Sie das Exemplar auf der Maschine, auf der das HA-NFS-Dateisystem angehängt ist.
4. Fügen Sie den Benutzer für das Exemplar den anderen Maschinen im Cluster hinzu, wobei sicherzustellen ist, dass die numerische Benutzer-ID identisch ist.
5. Registrieren Sie den Service `hadb2` mit dem Befehl **hadb2_reg**.
6. Führen Sie den Befehl **hadb2_setup** aus, um HA für das Exemplar einzurichten.

Anmerkung: Die Verwendung von NIS zur Definition von Informationen für das HA-Exemplar ist nicht zu empfehlen, weil NIS einen einzelnen Fehlerpunkt bilden kann.

Der Befehl hadb2_setup

Der Befehl **hadb2_setup** ist der zentrale Punkt der Programme, die mit dem DB2-HA-Agenten geliefert werden. Er kann zur Einrichtung, Änderung und Löschung eines Exemplars verwendet werden. Ferner kann er dazu dienen, den Service hadb2_setup zu aktivieren und zu inaktivieren. Bei Verwendung dieses Befehls ist es nicht erforderlich, die Datei `hadb2tab` manuell zu editieren.

Anmerkung: Der Befehl **hadb2_setup** führt Aktionen nur auf der Maschine aus, auf der er aufgerufen wird. Änderungen, die auf einer Maschine vorgenommen werden, sollten auch auf den anderen Maschinen im Cluster durchgeführt werden.

Folgende Argumente werden unterstützt:

```
Zum Hinzufügen eines EE-Exemplars:
----------------------------------
   hadb2_setup -a -i <exemplar> -f [on|off] -h <logischer-host>
   -p [DATA|ADMIN] -t EE

   Beispiel:
      hadb2_setup -a -i db2ee -f off -h log1 -p DATA -t EE


Zum Hinzufügen eines EEE-Exemplars:
-----------------------------------
   hadb2_setup -a -i <exemplar> -f [on|off] -h <nfs-host> -l <mount-punkt> \
      -r <ha-nfs-verz> -p [DATA|ADMIN] -t EEE -n "<knoteninfo>"

   Beispiel:
      hadb2_setup -a -i db2eee -f off -h ha-sun1 -l /export/ha_home \
         -r /log0/home -p DATA -t EEE -n "log0[0,10,20],log1[30,40,50]"


Zum Löschen eines Exemplars:
----------------------------
   hadb2_setup -d -i <exemplar>


Zum Ändern eines Exemplars:
---------------------------
   hadb2_setup -m -i <exemplar> [-f [on|off] | -l <mount-punkt> | \
      -h <host> | -p [DATA|ADMIN] | -r <ha-nfs_verz> | -t [EE|EEE] ]


Weitere Optionen:
-----------------
   -s   <on|off>       Aktivieren/Inaktivieren von hadb2 (für alle HA-Exemplare)
   -y                  Annehmen von YES für Sicherheitsprüfungen
```

Zur Aktivierung oder Inaktivierung des Service hadb2 geben Sie den Schalter -s an. Dies ist äquivalent zur Verwendung des Befehls **hareg** mit den Schaltern -n und -y und der Angabe des Service hadb2. Weitere Informationen zum Befehl **hareg(1m)** finden Sie in der Sun Cluster-Dokumentation.

Der Fehlermonitor für das Exemplar kann mit dem Schalter -f inaktiviert werden. Dies bewirkt, dass der Fehlermonitor für das Exemplar auf der lokalen Maschine gestoppt und die Datei hadb2tab geändert wird, um die Tatsache zu vermerken, dass die Fehlerüberwachung ausgeschaltet wurde.

Für EE-Exemplare wird das Ausschalten der Fehlerüberwachung auf allen Maschinen empfohlen, wenn das Exemplar von einer anderen Maschine übernommen wird. Für EEE-Exemplare muss die Fehlerüberwachung auf allen Maschinen inaktiviert werden, die Datenbankpartitionen für das Exemplar enthalten, bevor es manuell heruntergefahren wird.

Zum Löschen eines Exemplars dient der Schalter -d. Durch diesen Schalter wird das Exemplar nur aus der Datei hadb2tab entfernt. Andere Dateien oder Verzeichnisse werden nicht gelöscht oder geändert. Da die Datei hadb2tab die Hauptkonfigurationsdatei für den HA-DB2-Agenten ist, bewirkt das Löschen eines Exemplars aus dieser Datei, dass die Steuermethoden das Vorhandensein des Exemplars nicht mehr erkennen.

Zum Ändern eines Exemplars verwenden Sie den Schalter -m. Durch diesen Schalter werden Informationen in der Datei hadb2tab geändert. Andere Dateien oder Verzeichnisse werden nicht gelöscht oder geändert. Der Schalter -m kann mit jedem Schalter verwendet werden, der sich auf Informationen in der Datei hadb2tab bezieht. Die Datei db2nodes.cfg und die Datei hadb2-eee.cfg müssen im Anschluss an die Erstkonfiguration manuell geändert werden, weil der Befehl **hadb2_setup** eine Änderung dieser Dateien nicht unterstützt.

Das Hinzufügen eines Exemplars ist etwas komplizierter.

Für EE-Exemplare sind die folgenden Parameter erforderlich:

```
hadb2_setup -a -i <exemplar> -f <fü> -h <logischer-host> -t
<EEE-oder-EE> -p <zweck>
```

Dabei gilt: *exemplar* ist der Name des hinzuzufügenden Exemplars, *fü* gibt an, ob die Fehlerüberwachung anfangs aktiv (ON) oder inaktiv (OFF) ist, *logischer-host* ist der zugeordnete logische Host, *EEE-oder-EE* wird auf EE gesetzt und *zweck* kann entweder DATA oder ADMIN sein.

Für EEE-Exemplare sind die folgenden Parameter erforderlich:

```
hadb2_setup -a -i <exemplar> -f <fü> -h <nfs-host> -t <EEE-oder-EE> -p
   <zweck> -l <mount-punkt> -r <HA-NFS-verzeichnis> -n <knoteninfo>
```

Dabei gilt: *exemplar* ist der Name des hinzuzufügenden Exemplars, *fü* gibt an, ob die Fehlerüberwachung anfangs aktiv (ON) oder inaktiv (OFF) ist, *nfs-host* ist der Host-Name für den logischen Host, der das HA-NFS-Dateisystem exportiert, *EEE-oder-EE* wird auf `EEE` gesetzt, *zweck* kann entweder `DATA` oder `ADMIN` sein, *mount-punkt* ist der lokale Mount-Punkt für das HA-NFS-Verzeichnis, *HA-NFS-verzeichnis* ist das HA-NFS-Verzeichnis und *knoteninfo* sind die Informationen, durch die Datenbankpartitionen einem logischen Host zugeordnet werden. Beispiel:

```
hadb2_setup -a -i db2eee -f on -h jolt -l /export/ha_home -p DATA -t EEE -r
   /log1/home -n "log0[0,1],log1[2,3]"
```

Wenn ein EEE-Exemplar hinzugefügt wird, müssen die Knoteninformationen in Anführungszeichen gesetzt werden. In diesem Beispiel wird das Exemplar ″db2eee″ zwei logischen Hosts ″log0″ und ″log1″ zugeordnet. Die Datenbankpartitionen ″0″ und ″1″ des Exemplars ″db2eee″ werden dem logischen Host ″log0″ und die Datenbankpartitionen ″2″ und ″3″ dem logischen Host ″log1″ zugeordnet. Verwenden Sie den Befehl **hadb2_setup**, um ein Exemplar allen Maschinen im Cluster hinzuzufügen. Das Exemplar kann anschließend gestartet werden, indem eine Änderung der Clusterkonfiguration erzwungen wird oder indem der Service `hadb2` inaktiviert und wieder aktiviert wird. Dies kann entweder mit Hilfe des Befehls **hareg** oder mit Hilfe des Schalters `-s` des Befehls **hadb2_setup** geschehen. Wenn das Exemplar nicht startet, lesen Sie die Informationen in „Fehlerbehebung” auf Seite 331.

Wenn der Befehl **hadb2_setup** ein EEE-Exemplar hinzufügt, werden die folgenden Aktionen transparent ausgeführt:

- Überprüfen der angegebenen Informationen. Dies umfasst auch eine Überprüfung, ob der Benutzer im System vorhanden ist und ob HA-NFS aktiv ist.
- Erstellen einer Datei `db2nodes.cfg`.
- Erstellen einer Datei `hadb2-eee.cfg`.
- Erstellen einer Datei `.rhosts` für das EEE-Exemplar.
- Erstellen symbolischer Verbindungen vom Standarddatenbankpfad zu Datenverzeichnissen der zugeordneten logischen Hosts.
- Einfügen einer Zeile in die Datei `hadb2tab`.

Zur Vermeidung von Konfigurationsfehlern und zur Sicherstellung, dass das HA-Exemplar nach der Ausführung des Befehls **hadb2_setup** gestartet werden kann, führt der Befehl eine beträchtliche Menge an Tests aus, bevor ein neues Exemplar hinzugefügt wird. Die Datei `db2nodes.cfg` wird erstellt und mit Startinformationen zum aktuellen Clusterstatus gefüllt. Wenn zum Beispiel der logische Host ″log0″ in der Maschine ″crackle″ enthalten ist, enthalten die Einträge für die dem Host ″log0″ zugeordneten Datenbankpartitionen den Maschinennamen ″crackle″ und die Hochgeschwindigkeitsverbindung für ″crackle″:

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 thrash 0 204.152.65.34
3 thrash 1 204.152.65.34
```

Die Datei hadb2-eee.cfg wird nur auf der Basis der im Befehl angegebenen Knoteninformationen erstellt. Für jede Datenbankpartition ist eine Zeile vorhanden:

```
sphere % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

Die Datei .rhost ist für DB2 UDB EEE erforderlich und sollte alle Host-Namen (oder IP-Adressen) für jede Maschine im Cluster enthalten. Beispiel:

```
crackle db2eee
204.152.65.1 db2eee
204.152.65.17 db2eee
thrash db2eee
204.152.65.2 db2eee
204.152.65.18 db2eee
crackle db2eee
jolt db2eee
bump db2eee
thrash.torolab.ibm.com db2eee
crackle.torolab.ibm.com db2eee
```

In Übereinstimmung mit einem Dateisystemlayout für SMS-Tabellenbereiche richtet der Befehl **hadb2_setup** eine Reihe von Verzeichnissen und symbolischen Verbindungen ein. Dazu gehören:

- Ein Verzeichnis namens ″data″ unter dem Dateisystem des logischen Hosts für jeden logischen Host.
- Ein Knotenverzeichnis (unter diesem Verzeichnis ″data″) für jede Datenbankpartition, die dem logischen Knoten zugeordnet ist.
- Symbolische Verbindungen im Standarddatenbankpfad unter ~<exemplar>, wobei ~<exemplar> das Ausgangsverzeichnis des Exemplars ist. Für jede Datenbankpartition, die auf das entsprechende Knotenverzeichnis zeigt, ist eine symbolische Verbindung vorhanden. Weitere Informationen finden Sie in „Plattenlayout für EE- und EEE-Exemplare” auf Seite 307.

Dauer der Funktionsübernahme

Die Dauer einer Funktionsübernahme wird von dem Zeitpunkt an gemessen, zu dem die Daten zuerst nicht mehr verfügbar sind, bis zu dem Zeitpunkt, zu dem sie wieder verfügbar werden. Eine Reihe von Ereignissen, die während einer Funktionsübernahme auftreten, können einen erheblichen Beitrag zur Dauer der Funktionsübernahme leisten:

- Deportieren und Importieren von Platten

  Das Deportieren und Importieren von Platten benötigt in der Regel nicht sehr lange Zeit im Vergleich zu anderen Ereignissen, jedoch kann sich dadurch die allgemeine Ausfallzeit verlängern. Je mehr Platten von einer Maschine auf eine andere bei der Funktionsübernahme versetzt werden müssen, um so länger dauert der Prozess. Wenn es fehlerhafte Platten gibt, kann der Prozess noch länger dauern.
- `Fsck`-Prüfung der Dateisysteme, die für einen logischen Host angehängt sind

  Bevor die Dateisysteme des logischen Hosts angehängt werden können, müssen sie eine `fsck`-Prüfung durchlaufen, die den einwandfreien Zustand des Dateisystems sicherstellt. Je größer das Dateisystem ist, desto länger dauert dieser Prozess. Durch Verwendung eines Journaled File System kann diese Zeit erheblich verringert werden. Da in einer HA-Umgebung in der Regel Journaled File Systems verwendet werden, spielt die `fsck`-Zeit gewöhnlich keine große Rolle.
- Benutzer-Scripts, die von dem HA-Agenten aufgerufen werden

  Der HA-Agent ruft Benutzer-Scripts auf, wenn sie vorhanden und ausführbar sind. Einige dieser Scripts werden synchron ausgeführt und können die Zeit verlängern, die zum Starten von HA-Exemplaren benötigt wird. Stellen Sie sicher, dass sie so schnell wie möglich ausgeführt werden. Ziehen Sie in Betracht, externe Programme, die von diesen Scripts aufgerufen werden, im Hintergrund auszuführen.
- HA-NFS

  Für ein einzelnes EEE-Exemplar in einer Konfiguration zur gegenseitigen Übernahme muss HA-NFS für das Ausgangsverzeichnis des Exemplareigners verwendet werden. HA-NFS verlängert die Funktionsübernahmedauer wegen der Karenzzeit für *lockd* (definiert im HA-Agenten für HA-NFS), die 90 Sekunden bei Ausführung von HA-NFS beträgt. Die Funktionsübernahmedauer ist betroffen, weil jeder Prozess, der eine Datei im HA-NFS-Dateisystem nach einer Funktionsübernahme sperrt, warten muss, bis die Karenzzeit abgelaufen ist. Der HA-Agent für DB2 ist der erste Prozess, der eine Datei unter dem Ausgangsverzeichnis des Exemplareigners nach einer Funktionsübernahme sperrt, und er zeichnet die Zeit auf, die er benötigt, um die erste Sperre zu erhalten. Diese Zeit wird im Statusbericht nach einer Funktionsübernahme angegeben.
- Starten von DB2

  Das Starten von DB2 trägt nur einen kleinen Teil zur Dauer der Funktionsübernahme bei. Für ein EE-Exemplar sind dies nur 5 bis 15 Sekunden im Durchschnitt. Für ein EEE-Exemplar trägt es ungefähr 10 Sekunden plus je fünf Sekunden pro Datenbankpartition bei, die übernommen wird. Wenn zum Beispiel drei Datenbankpartitionen übernommen werden, beträgt die Übernahmezeit, die durch das Starten dieser drei Datenbankpartitionen hin-

zugefügt wird, ungefähr 25 Sekunden. Dies umfasst *nicht* die Zeit, die zur Wiederherstellung nach einem Systemabsturz des Exemplars benötigt wird.

- Wiederherstellung nach einem Systemabsturz der Datenbank

  Die Wiederherstellung nach einem Systemabsturz ist häufig der größte Einzelposten in der Ausfallzeit bei einer Funktionsübernahme. Wie lange es dauert, eine Datenbank wiederherzustellen, hängt von einer Reihe Faktoren ab:
  - Client-Auslastung. Nur Änderungen an der Datenbank werden in den Transaktionsprotokollen aufgezeichnet. Wenn die Client-Auslastung überwiegend aus Leseoperationen besteht, müssen während der Wiederherstellung nach einem Systemabsturz relativ wenige Transaktionen auf die Datenbank angewendet werden.
  - Platten- und Maschinengeschwindigkeit. Die Geschwindigkeit der Platten und der Maschine, auf der das HA-Exemplar ausgeführt wird, spielt ebenfalls eine Rolle für die Zeit, die eine Wiederherstellung der Datenbank benötigt. Je schneller das System arbeitet, desto kürzer ist die für eine Wiederherstellung nach einem Systemabsturz benötigte Zeit.
  - Der Wert des Datenbankkonfigurationsparameters *softmax*. Der Wert des Parameters *softmax* definiert den Prozentsatz der Protokolldateigröße, bei dem ein bedingter Prüfpunkt vorzunehmen und eine Protokollsteuerdatei zu schreiben ist. Die Protokollsteuerdatei dient während der Wiederherstellung nach einem Systemabsturz dazu, zu bestimmen, welche Protokollsätze tatsächlich zur Wiederherstellung der Datenbank in einem konsistenten Zustand erforderlich sind. Eine Verringerung dieses Werts bewirkt, dass der Datenbankmanager die Seitenlöschfunktionen häufiger auslöst und häufigere bedingte Prüfpunkte ansetzt. Obwohl dadurch die Leistung herabgesetzt wird, erfolgt die Wiederherstellung der Datenbank schneller.
  - Unterschied zwischen EE- oder EEE-Exemplar. Wenn es sich um ein EEE-Exemplar handelt, werden die Operationen zum Neustart der Datenbank parallel ausgeführt. Jede Datenbankpartition ist für den Neustart des eigenen Teils der Datenbank zuständig. Bei einer Datenmenge von 50 GB für eine Datenbank kann ein Exemplar mit vier Datenbankpartitionen die Datenbank ungefähr viermal schneller als ein EE-Exemplar wiederherstellen.

Fehlerbehebung

In der folgenden Tabelle werden Probleme aufgeführt, die auftreten können, ihre wahrscheinlichen Ursachen beschrieben und Maßnahmen zu ihrer Behebung erläutert.

Tabelle 16. Fehlerbehebung bei hoher Verfügbarkeit unter Sun Cluster 2.2

Symptom	Mögliche Ursache	Maßnahme
Dateisystem von logischem Host lässt sich nicht anhängen.	Das Dateisystem des logischen Hosts wird normalerweise während der Funktionsübernahme eines logischen Hosts angehängt und abgehängt. Während der Funktionsübernahme sollten keine aktiven Prozesse oder geöffneten Dateien unter dem Dateisystem des logischen Hosts vorhanden sein. In seltenen Fällen haben Prozesse, die nicht zwangsweise beendet werden können, ihr aktuelles Arbeitsverzeichnis unter dem Dateisystem des logischen Hosts. Verwenden Sie `fuser(1m)` oder ein GNU-Dienstprogramm namens `lsof`, um herauszufinden, ob ein Prozess unter dem Mount-Punkt vorhanden ist. Fehlernachrichten werden generiert, wenn das Dateisystem des logischen Hosts nicht angehängt werden kann.[a]	Starten Sie das System erneut, oder versetzen Sie das Dateisystem des logischen Hosts in einen anderen Namen, und erstellen Sie es erneut. Dadurch kann der eingefrorene Prozess unter dem Verzeichnis bleiben (da er nicht abgebrochen werden kann), und das Anhängen kann stattfinden.[b]
Das Zeitlimit für db2start oder db2stop funktioniert nicht.	Ein Signal SIGALRM kann eventuell nicht aus einem blockierenden Systemaufruf ausbrechen. Statt dessen wird der Systemaufruf erneut gestartet so, als wäre die Markierung SA_RESTART mit `sigaction()` gesetzt. Dies bewirkt, dass Zeitlimits für die DB2-HA-Agenten ignoriert werden und die Agentenmethode blockiert, anstatt einen blockierten Befehl **db2start** oder **db2stop** zu beheben.	Wenden Sie die erforderliche Programmkorrektur 105210-17 (oder höher) für Solaris 2.6 an.

Tabelle 16. Fehlerbehebung bei hoher Verfügbarkeit unter Sun Cluster 2.2 (Forts.)

Symptom	Mögliche Ursache	Maßnahme
Anmelden an Exemplar blockiert.	Obwohl es zahlreiche Ursachen hierfür geben kann, gehören zu den häufigsten Ursachen NFS-Probleme und das Programm `/usr/sbin/quota`.	Überprüfen Sie die angehängten NFS, um sicherzustellen, dass sie ordnungsgemäß funktionieren, und suchen Sie nach Quota-Prozessen, die dem Exemplareigner gehören. Das Problem kann eventuell durch eine Änderung des Quota-Programms in eine symbolische Verbindung zu `/bin/true` behoben werden, was jedoch im Ermessen des Systemadministrators liegt. Dies ist keine empfohlene Lösung, aber sie funktioniert vielleicht.
Nach der Einrichtung eines EEE-Exemplars lässt sich dieses nicht starten.	Der Befehl **hadb2_setup** fügt keine Ports in die Datei `/etc/services` ein. Es wird erwartet, dass der Administrator diese manuell hinzufügt. Es wird eine Fehlernachricht zurückgegeben.[c]	Stellen Sie sicher, dass Sie die richtigen Ports in die Datei `/etc/services` eingetragen haben.

Tabelle 16. Fehlerbehebung bei hoher Verfügbarkeit unter Sun Cluster 2.2 (Forts.)

Symptom	Mögliche Ursache	Maßnahme
Methode START_NET kann DB2 nicht starten.		Schalten Sie die Fehlerüberwachung aus, um sicherzustellen, dass das Exemplar nicht übernommen wird. Melden Sie sich als Exemplareigner an, und versuchen Sie, DB2 manuell zu starten. 1. Stellen Sie sicher, dass in der Konfigurationsdatei `hadb2tab` die korrekte Exemplarart angegeben ist. Wenn zum Beispiel eine Datei `db2nodes.cfg` für ein administratives EE-Exemplar vorhanden ist, verursacht dies Probleme. Die HA-Agentenmethoden können diese Probleme nicht überwinden. 2. Stellen Sie sicher, dass die Datei `.rhosts` vorhanden ist und gültige Einträge enthält. 3. Stellen Sie sicher, dass das HA-NFS-Dateisystem mit Root-Berechtigung von allen Maschinen im Cluster gemeinsam benutzt werden kann. 4. Überprüfen Sie die Kernel-Parameter, und stellen Sie sicher, dass sie korrekt sind. 5. Stellen Sie sicher, dass die Datei `/etc/services` Einträge für das Exemplar enthält.

Tabelle 16. Fehlerbehebung bei hoher Verfügbarkeit unter Sun Cluster 2.2 (Forts.)

Symptom	Mögliche Ursache	Maßnahme
Das Exemplar funktioniert nur auf einer Maschine.	• Die numerische *uid* für das Exemplar ist vielleicht nicht auf allen Maschinen im Cluster gleich. • Die Kernel-Parameter sind vielleicht nicht auf jeder Maschine im Cluster gültig. • Die Datei `hadb2tab` ist vielleicht nicht auf jeder Maschine im Cluster gleich. • Andere Konfigurationsdateien, wie zum Beispiel die Datei `vfstab` für logische Hosts, sind vielleicht nicht auf jeder Maschine im Cluster gleich.	Wenn keine dieser Ursachen zuzutreffen scheint, versuchen Sie, sich als Exemplareigner anzumelden und DB2 manuell zu starten. Für EE-Exemplare sollte dies funktionieren, wenn der logische Host, der dem Exemplar zugeordnet ist, auf der aktuellen Maschine aktiv ist. Für EEE-Exemplare sollte dies von jeder Maschine im Cluster aus funktionieren, die die Datenbankpartitionen übernehmen kann.
su - <exemplar> -c "db2start" funktioniert nicht.	• Die Datei `.profile` für das Exemplar ist eventuell nicht **su**-"freundlich". • Es gibt ein bekanntes Problem mit der Bourne-Shell (`/bin/sh`), in der der Befehl **su** manuell, jedoch nicht über den HA-Agenten funktioniert.	• Versuchen Sie mit Root-Berechtigung, diesen Befehl manuell auszuführen, und stellen Sie sicher, dass er funktioniert, bevor Sie den Versuch über den HA-Agenten wiederholen. • Wechseln Sie zur Korn-Shell (`/bin/ksh`), falls erforderlich.
Das EEE-Exemplar kann nicht starten, aber das Ausgangsverzeichnis wird angehängt.	Das HA-NFS-Verzeichnis wurde eventuell nicht mit der Root-Berechtigung auf die Maschinen im Cluster exportiert. Sowohl für DB2 als auch für die HA-Agenten muss dieses Verzeichnis ordnungsgemäß arbeiten.	Versuchen Sie testweise, eine Datei (als Root) unter dem Ausgangsverzeichnis des Exemplareigners zu erstellen.
Der Versuch eines Zugriffs auf das EEE-Exemplarverzeichnis liefert eine Fehlernachricht "Stale NFS file handle".	Es gibt eventuell noch Prozesse unter dem Ausgangsverzeichnis des Exemplareigners.	Hängen Sie das Ausgangsverzeichnis des Exemplareigners ab, und lassen Sie den HA-Agenten es erneut anhängen. Der HA-Agent hängt es wieder an, wenn der Service `hadb2` inaktiviert und wieder aktiviert wird (siehe Beschreibung des Schalters `-s` in „Der Befehl hadb2_setup" auf Seite 325).

Tabelle 16. Fehlerbehebung bei hoher Verfügbarkeit unter Sun Cluster 2.2 (Forts.)

Symptom	Mögliche Ursache	Maßnahme
Steuermethoden werden durch SC2.2 nicht erfolgreich ausgeführt.	Der Service `hadb2` ist eventuell nicht bei der Sun Cluster-Software registriert, oder er ist inaktiviert.	Wenn es scheint, als könnten die Steuermethoden über die Befehlszeile normal ausgeführt werden, prüfen Sie die SYSLOG-Dateien auf Fehlernachrichten, die helfen, das Problem zu erklären. Stellen Sie sicher, dass der Service `hadb2` bei der Sun Cluster-Software registriert ist und dass er aktiviert ist. Die manuelle Ausführung der Methoden ist für das Debugging eines Problems hilfreich.[d] Die Methoden sollten mit Root-Berechtigung und den entsprechenden Befehlszeilenparametern ausgeführt werden. Wenn die Liste logischer Hosts leer ist, sollte der Parameter als `""` angegeben werden. Doppelte Anführungszeichen ohne Leerzeichen als Trenner bezeichnen ein leeres Argument. Beispiel: `hadb2_startnet log0,log1 "" 600` Das erste Argument `log0,log1` teilt der Methode `hadb2_startnet` mit, dass die logischen Hosts `log0` und `log1` auf der aktuellen Maschine aktiv sind. Das zweite Argument ist leer, was der Methode `hadb2_startnet` anzeigt, dass es keine anderen logischen Hosts gibt, die auf anderen Maschinen im Cluster aktiv sind (alle sind auf der aktuellen Maschine aktiv). Das dritte Argument teilt der Methode mit, dass für SC2.2 ein Zeitlimit von 600 Sekunden gesetzt ist.

Tabelle 16. Fehlerbehebung bei hoher Verfügbarkeit unter Sun Cluster 2.2 (Forts.)

Symptom	Mögliche Ursache	Maßnahme
Benutzer-Scripts lassen sich nicht ausführen.	Die Benutzer-Scripts können nur ausgeführt werden, wenn sie in den richtigen Verzeichnissen vorhanden und ausführbar sind.	Überprüfen Sie die Eigentumsrechte und Attribute der Dateien. Wenn ein Script sich nicht ausführen lässt, wenden Sie sich an den IBM Kundendienst. Leiten Sie eine Verzeichnisliste des Scripts, das sich nicht ausführen lässt, und die SYSLOG-Ausgabe für eine Funktionsübernahme bzw. einer Clusterrekonfiguration weiter, durch die das Script hätte ausgeführt werden müssen.
Informationen werden nicht in der in /etc/syslog.conf angegebenen Datei protokolliert.		Verwenden Sie `touch(1)` zur Erstellung der Datei, die in der Datei `/etc/syslog.conf` angegeben ist, und starten Sie den SYSLOG-Dämon erneut.

Tabelle 16. Fehlerbehebung bei hoher Verfügbarkeit unter Sun Cluster 2.2 (Forts.)

Symptom	Mögliche Ursache	Maßnahme

[a] Fehlernachrichten, die generiert werden, wenn das Dateisystem des logischen Hosts nicht angehängt werden kann, können etwa wie folgt aussehen:

```
Aug 17 11:14:01 rash ID[SUNWcluster.loghost.1170]: importing data1
Aug 17 11:14:06 rash ID[SUNWcluster.scnfs.3040]: mount -F ufs -o ""
     /dev/vx/dsk/data1/data1-stat /log1 failed.
Aug 17 11:14:07 rash ID[SUNWcluster.ccd.ccdd.5304]: error freeze cmd =
     /opt/SUNWcluster/bin/loghost_sync
CCDSYNC_POST_ADDU LOGHOST_CM:log1:rash /etc/opt/SUNWcluster/conf/ccd.database
     2 "0 1" 1 error code = 1
```

[b] Beispiel:

```
scadmin@rash(218)# ps -fe | egrep db2
db2ee 1984 1 0 0:01 <defunct>

Lösung:

   scadmin@rash(229)# cd /
   scadmin@rash(230)# mv /log1 /log1.bkp
   scadmin@rash(231)# mkdir /log1
```

[c] Die Fehlernachricht kann etwa wie folgt aussehen:

```
SQL6030N Der Befehl START DATABASE MANAGER oder STOP DATABASE MANAGER
         ist fehlgeschlagen. Ursachencode: "13".
```

[d] Wenn die Methode `hadb2_startnet` zum Beispiel `libdb2.so.1` nicht finden kann, aber über die Sun Cluster-Software normal ausgeführt wird, werden keine Fehlernachrichten gemeldet. Eine manuelle Ausführung der Methode führt zu folgendem Ergebnis:

```
scadmin@crackle(213)# hadb2_startnet '''log0,log1' 600
ld.so.1: hadb2_startnet: fatal: libdb2.so.1: open failed:
   No such file or directory
Killed
```

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.

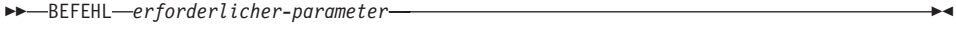
Bei der Eingabe von Daten eines Syntaxdiagramms müssen Sie auch Zeichen wie Anführungszeichen und Gleichheitszeichen eingeben.

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.

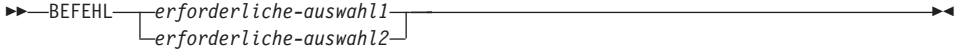
Erforderliche Parameter werden im Hauptpfad angegeben:



Optionale Parameter werden unterhalb des Hauptpfads angegeben:

Der Standardwert eines Parameters wird oberhalb des Pfads angegeben:

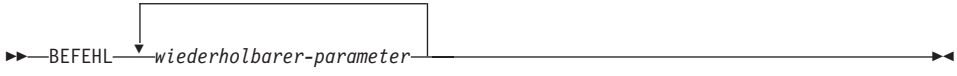
Ein Stapel von Parametern, bei dem der erste Parameter im Hauptpfad angegeben ist, gibt an, dass einer der Parameter gewählt werden muss:



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:

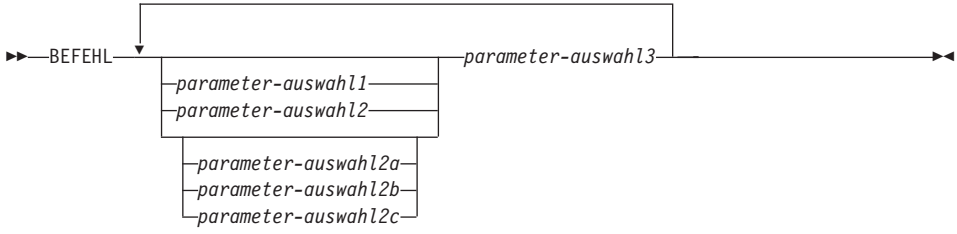
- Wenn der Pfeil durchgängig ist, kann das Element in einer Liste wiederholt werden, wobei die Elemente durch Leerstellen zu trennen sind:



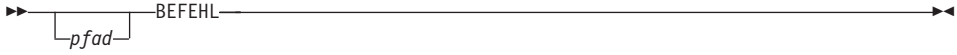
- Wenn der Pfeil ein Komma enthält, kann das Element in einer Liste wiederholt werden, wobei die Elemente durch Kommas zu trennen sind:

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:

Lesen von Syntaxdiagrammen

Anhang B. Warnungen, Fehler- und Beendigungsnachrichten

Die von den verschiedenen Sicherungs- und Wiederherstellungsdienstprogrammen 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.

Weitere Informationen zu allen DB2-Nachrichten finden Sie im Handbuch *Fehlernachrichten*. Sie können die Informationen in diesem 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 UNIX-Systemen wird die Verwendung von doppelten Anführungszeichen als Begrenzer empfohlen. Dadurch werden Probleme vermieden, wenn in dem Verzeichnis aus einem Einzelzeichen bestehende Dateinamen vorhanden sind:

```
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-Basisbetriebssystemen und anderen, die die Pipe „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

db2adutl - Mit von TSM archivierten Images arbeiten

Ermöglicht Benutzern das Abfragen, Extrahieren, Prüfen und Löschen von Sicherungsimages und Ladekopieimages, die mit Hilfe von Tivoli Storage Manager (früher ADSM) gespeichert wurden.

Auf UNIX-Systemen befindet sich dieses Dienstprogramm im Verzeichnis INSTHOME/sqllib/misc. Auf Windows- und OS/2-Systemen befindet es sich im Verzeichnis \sqllib\misc.

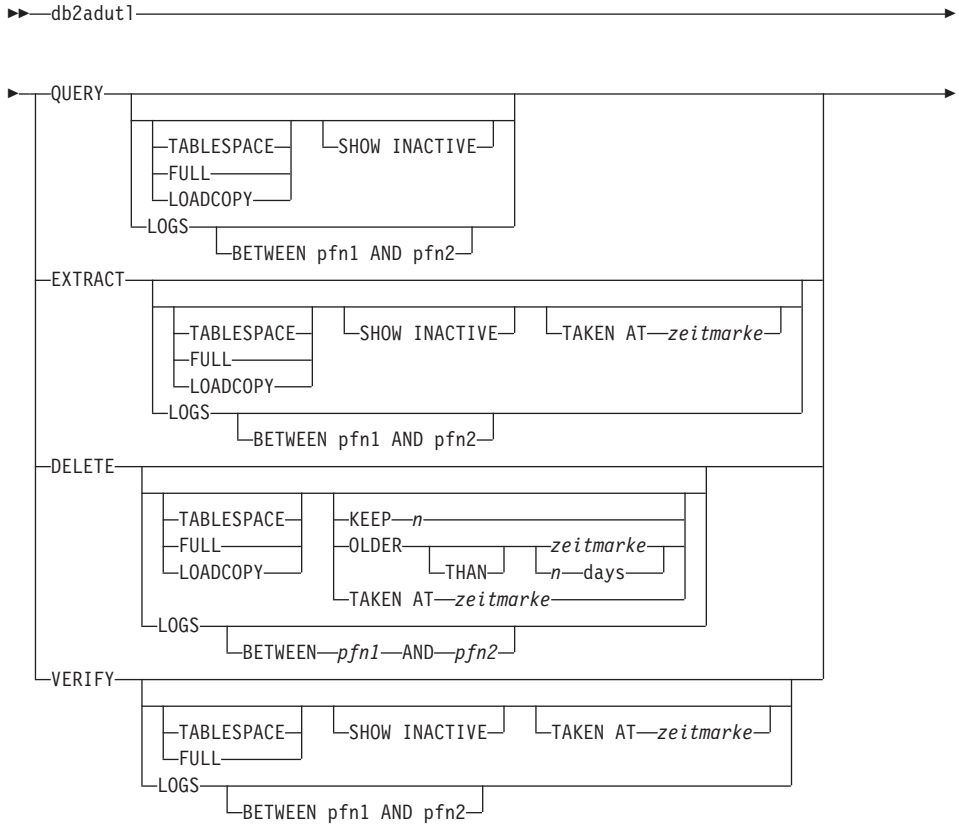
Berechtigung

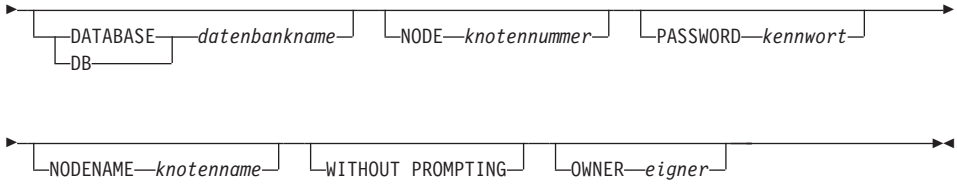
Keine

Erforderliche Verbindung

Keine

Befehlssyntax





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.

TABLESPACE
Erfasst nur Sicherungsimages von Tabellenbereichen.

FULL Erfasst nur vollständige Sicherungsimages von Datenbanken.

LOADCOPY
Erfasst nur Ladekopieimages.

LOGS Umfasst nur Protokollarchivimages.

BETWEEN *pfn1* **AND** *pfn2*
Gibt an, dass die Protokolle zwischen der Protokollfolgenummer 1 und der Protokollfolgenummer 2 verwendet werden sollen.

SHOW INACTIVE
Erfasst Sicherungsobjekte, die inaktiviert wurden.

TAKEN AT *zeitmarke*
Gibt ein Sicherungsimage über seine Zeitmarke an.

KEEP *n*
Inaktiviert alle Objekte des angegebenen Typs außer den *n* nach Zeitmarke letzten Objekten.

OLDER THAN *zeitmarke* **oder** *n* **days**
Gibt an, dass Objekte mit einer Zeitmarke älter als *zeitmarke* oder *n* Tage inaktiviert werden.

DATABASE *datenbankname*
Betrifft nur Objekte, die dem angegebenen Datenbanknamen zugeordnet sind.

NODE *knotennummer*
Betrifft nur Objekte, die auf dem Knoten mit der angegebenen Nummer erstellt worden sind.

PASSWORD *kennwort*
Gibt das TSM-Client-Kennwort 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.

WITHOUT PROMPTING
Der Benutzer wird vor dem Löschen von Objekten nicht zur Bestätigung aufgefordert.

OWNER *eigner*
Betrifft nur Objekte, die der angegebene Eigner erstellt hat.

Beispiele

Es folgt eine Beispielausgabe des folgenden Befehls: `db2 backup database rawsampl use tsm`

```
Sicherung erfolgreich. Die Zeitmarke für diese Sicherungsimagedatei ist:
19970929130942
db2adutl query

Query for database RAWSAMPL
Retrieving full database backup information.
   full database backup image: 1, Time: 19970929130942,
                                  Oldest log: S0000053.LOG, Sessions used: 1
   full database backup image: 2, Time: 19970929142241,
                                  Oldest log: S0000054.LOG, Sessions used: 1

Retrieving table space backup information.
   table space backup image: 1, Time: 19970929094003,
                                Oldest log: S0000051.LOG, Sessions used: 1
   table space backup image: 2, Time: 19970929093043,
                                Oldest log: S0000050.LOG, Sessions used: 1
   table space backup image: 3, Time: 19970929105905,
                                Oldest log: S0000052.LOG, Sessions used: 1
```

```
Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

Es folgt eine Beispielausgabe des folgenden Befehls: `db2adutl delete full taken at 19950929130942 db rawsampl`

```
Query for database RAWSAMPL

Retrieving full database backup information. Please wait.

  full database backup image: RAWSAMPL.0.db26000.0.19970929130942.001

  Do you want to deactivate this backup image (Y/N)? y

    Are you sure (Y/N)? y

db2adutl query

Query for database RAWSAMPL

Retrieving full database backup information.
   full database backup image: 2, Time: 19950929142241,
                           Oldest log: S0000054.LOG, Sessions used: 1

Retrieving table space backup information.
   tablespace backup image: 1, Time: 19950929094003,
                         Oldest log: S0000051.LOG, Sessions used: 1
   tablespace backup image: 2, Time: 19950929093043,
                         Oldest log: S0000050.LOG, Sessions used: 1
   tablespace backup image: 3, Time: 19950929105905,
                         Oldest log: S0000052.LOG, Sessions used: 1

Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

db2ckbkp - Sicherung überprüfen

Mit diesem Dienstprogramm können Sie die Integrität des Sicherungsimage testen und bestimmen, ob Sie das Image wiederherstellen können. Außerdem können Sie damit die Metadaten anzeigen, die im Sicherungs-Header gespeichert sind.

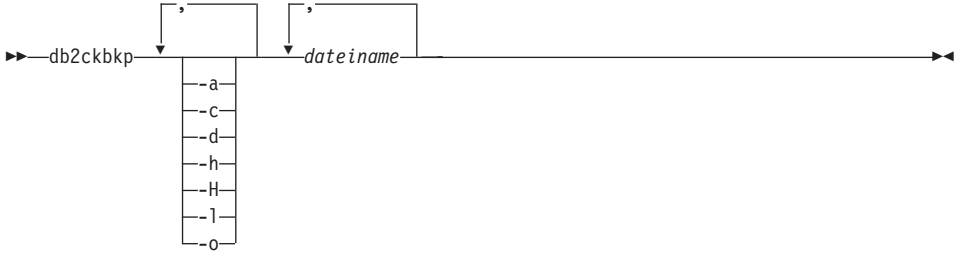
Berechtigung

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.

-d Zeigt Informationen aus den Headern von Datenseiten von DMS-Tabellenbereichen an.

-h Zeigt Datenträger-Header einschließlich des Imagenamens oder -pfades an, der vom Wiederherstellungsdienstprogramm erwartet wird.

-H Zeigt nur die Datenträger-Header-Daten an.

Anmerkungen:

1. Diese Option prüft das Image nicht auf Gültigkeit. Wenn diese Option nicht angegeben ist, erfolgt die Gültigkeitsprüfung für das gesamte Image.
2. Diese Option ist in Kombination mit einer beliebigen anderen Option ungültig.

-l Zeigt die LFH-Daten an (Protokoll-Header-Daten).

-o Zeigt Details aus den Objekt-Headern an.

dateiname
Der Name der Sicherungsimagedatei. Mindestens eine Datei kann 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 Image müssen Sie das erste Sicherungsimageobjekt (.001) als erstes angeben.

Beispiele

```
db2ckbkp SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.*
[1] Buffers processed:  ##
[2] Buffers processed:  ##
[3] Buffers processed:  ##
Image Verification Complete - successful.

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
        Node                           -- 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
        Backup Type                    -- 0
        Backup Gran.                   -- 0
        Status Flags                   -- 11
        System Cats inc                -- 1
        Catalog Node Number            -- 0
        DB Codeset                     -- ISO8859-1
        DB Territory                   --
        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

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.

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 mit der Angabe des Bandeinheitennamens auf. Auf UNIX-System geben Sie z. B. Folgendes an:

```
db2ckbkp -h /dev/rmt0
```

Unter Windows NT:

```
db2ckbkp -d \\.\tape1
```

Sofern das Sicherungsimage sich auf TSM befindet, finden Sie weitere Informationen in „db2adutl - Mit von TSM archivierten Images arbeiten” auf Seite 348.

db2ckrst - Reihenfolge der Images für 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.

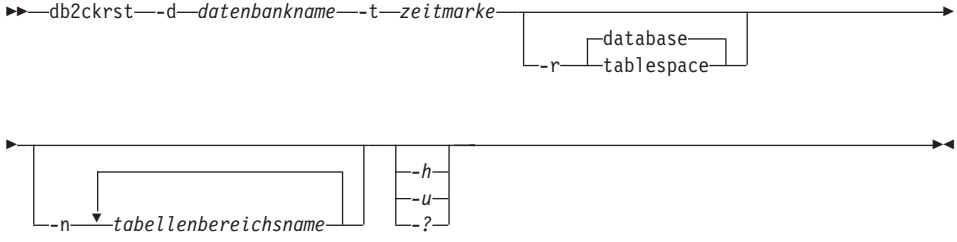
Berechtigung

Keine

Erforderliche Verbindung

Keine

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 einen Tabellenbereich ausgewählt haben und keine Tabellenbereichsnamen angegeben sein, sucht das Dienstprogramm im Protokolleintrag des angegebenen Image und verwendet für die Wiederherstellung die aufgelisteten Tabellenbereichsnamen.

-n tabellenbereichsname
Gibt den Namen mindestens eines Tabellenbereichs an, der wiederhergestellt wird.

Anmerkung: Wenn Sie die Datenbankwiederherstellung auswählen und eine Liste von Tabellenbereichsnamen angeben, fährt das Dienstprogramm mit einer 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

Backup successful. The timestamp for this backup image is : 20001016001426

> db2 backup db mr incremental

Backup successful. The timestamp for this backup image is : 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 die Option HISTORY FILE für den Befehl RESTORE DATABASE an, bevor Sie dieses Dienstprogramm verwenden.

db2ckrst - Reihenfolge der Images für Teilwiederherstellung überprüfen

Wenn Sie die Option FORCE des Befehls PRUNE HISTORY verwenden, können Sie Einträge löschen, die erforderlich sind, um Daten aus dem neuesten Gesamtsicherungsimage der Datenbank wiederherzustellen. Die Standardoperation des Befehls PRUNE HISTORY verhindert, dass erforderliche Einträge gelöscht werden. Sie sollten die Option FORCE des Befehls PRUNE HISTORY nicht verwenden.

Es empfiehlt sich, alle Sicherungsoperationen zu notieren und sich an diesem Dienstprogramm zu orientieren.

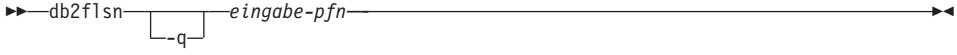
db2flsn - Protokollfolgenummer suchen

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 zwölf Byte lange Zeichenfolge, 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 Protokoll-Header-Steuerdatei `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 *logretain* auf `RECOVERY` oder *userexit* auf `ON` gesetzt ist.

db2inidb - Spiegeldatenbank initialisieren

In einer Umgebung mit Spiegeldatenbanken können Sie mit diesem Befehl eine gespiegelte Datenbank zu unterschiedlichen Zwecken initialisieren.

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. Diese Datenbank ist nur für den Lesezugriff.

STANDBY
Gibt an, dass die Datenbank in den Status *Aktualisierende Wiederherstellung anstehend* versetzt wird. 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 primären Datenbank verwendet werden kann.

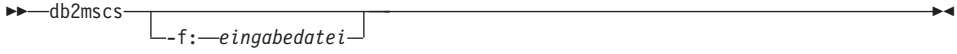
db2mscs - Windows NT-Dienstprogramm zur Funktionsübernahme (Failover Utility) einrichten

Erstellt die Infrastruktur für die Unterstützung der DB2-Funktionsübernahme unter Windows NT/2000 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.

ARCHIVE LOG

Schließt für eine wiederherstellbare Datenbank die aktive Protokolldatei und schneidet sie ab. Wenn der Benutzer-Exit aktiviert ist, wird eine Archivierungsanforderung abgesetzt.

Bereich

In einer MPP-Umgebung schließt dieser Befehl auf allen Knoten die aktiven Protokolle und schneidet diese Protokolle ab. Sie können jedoch eine Untergruppe von Knoten angeben.

Berechtigung

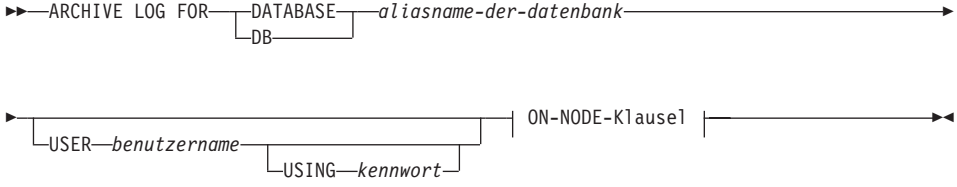
Eine der folgenden Berechtigungen:

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

Erforderliche Verbindung

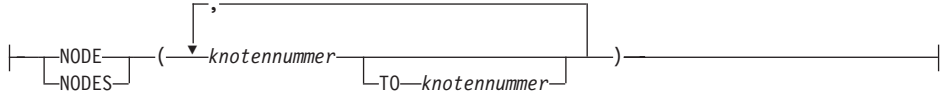
Mit diesem Befehl stellen Sie automatisch eine Verbindung zur angegebenen Datenbank her. Sollte bereits eine Verbindung vorhanden sein, wird ein Fehler zurückgegeben.

Befehlssyntax



Knotenlistenklausel:

Knotenlistenklausel:



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 NODES
Gibt an, dass der Befehl auf allen Knoten abgesetzt werden soll, die sich in der Datei `db2nodes.cfg` befinden. Dies ist die Standardeinstellung, falls keine Klausel für den Knoten angegeben ist.

EXCEPT
Gibt an, dass der Befehl auf allen Knoten abgesetzt werden soll, die sich in der Datei `db2nodes.cfg` befinden, mit Ausnahme der Knoten, die in der Knotenliste stehen.

ON NODE/ON NODES
Gibt an, dass die Protokolle für die angegebene Datenbank auf einer Gruppe von Knoten archiviert werden sollen.

knotennummer
Gibt eine Knotennummer in der Knotenliste an.

TO knotennummer
Sie können dies verwenden, wenn Sie einen Bereich von Knoten angeben, für die die Protokolle archiviert werden müssen. Alle Knoten ab er ersten in der Knotenliste angegebenen Knotennummer bis einschließlich der zweiten dort angegebenen Knotennummer.

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.

Sollten Transaktionen anderer Anwendungen laufen, während Sie diesen Befehl aufrufen, vermindert sich die Leistung geringfügig, während der Protokollpuffer durch Schreiben auf die Platte geleert wird; andere Transaktionen, die versuchen, Protokollsätze in den Puffer zu schreiben, müssen warten, bis dieser Vorgang beendet ist.

Dieser Befehl bewirkt, dass die Datenbank einen Teil ihres Speicherbereichs für Protokollfolgenummern verliert. Dadurch werden gültige Protokollfolgenummern schneller knapp.

INITIALIZE TAPE

DB2 für Windows NT/2000 unterstützt 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 über den Befehl BACKUP DATABASE (siehe „BACKUP DATABASE, Befehl" auf Seite 104) und über den Befehl RESTORE DATABASE (siehe „RESTORE DATABASE, Befehl" auf Seite 133) angegeben ist, 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.

Siehe auch

„REWIND TAPE" auf Seite 371

„SET TAPE POSITION" auf Seite 372

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 oder Umbenennen eines Tabellenbereichs, die Ausführung der Statistikfunktion, die Reorganisation der Tabelle und das Laden.

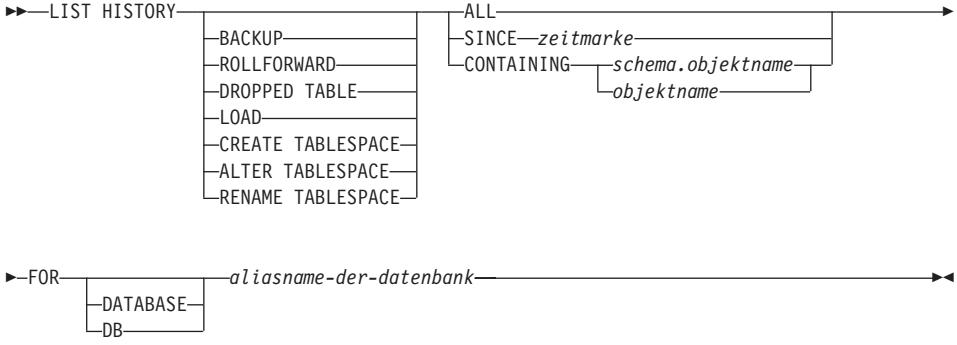
Berechtigung

Keine

Erforderliche Verbindung

Exemplar. Eine explizite Verbindung ist nicht erforderlich. Sollte die Datenbank als fern aufgelistet sein, wird für die Dauer des Befehls eine Exemplarverbindung zum fernen Knoten hergestellt.

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.

LOAD
Listet Ladeoperationen auf.

CREATE TABLESPACE
Listet auf den Tabellenbereich bezogene Erstellungs- und Löschoperationen auf.

RENAME TABLESPACE
Listet auf den Tabellenbereich bezogene Umbenennungsoperationen auf.

ALTER TABLESPACE
Listet Änderungsoperationen des Tabellenbereichs auf.

ALL Listet alle Einträge des angegebenen Typs in der Protokolldatei auf.

SINCE zeitmarke
Sie können eine vollständige Zeitmarke (Format *jjjjmmtthhnnss*) oder ein Präfix (mindestens *jjjj*) angeben. Alle Einträge mit Zeitmarken gleich der angegebenen Zeitmarke oder größer als diese 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 - Aktualisierende wiederherstellen
   G - Tabelle reorganisieren
   L - Laden
   N - Tabellenbereich umbenennen
   O - Tabellenbereich löschen
   Q - In den Wartemodus versetzen
   R - Wiederherstellen
   S - Statistik ausführen
   T - Tabellenbereich ändern
   U - Aus Tabelle laden
```

```
Art
Arten der Sicherung:

   F - Offline
   N - Online
   I - Inkrementell offline
   O - Inkrementell online
   D - Delta offline
   E - Delta online

Arten der aktualisierenden Wiederherstellung:

   E - Ende der Protokolle
   P - Zeitpunkt

Ladearten:

   I - Einfügen
   R - Ersetzen

Tabellenbereichsarten:

   C - Behälter hinzufügen
   R - Neu ausgleichen

Arten für das Versetzen in den Wartemodus:

   S - Freigabe in den Wartemodus versetzen
   U - Aktualisierung in den Wartemodus versetzen
   X - Exclusiv in den Wartemodus versetzen
   Z - Grundstellung in den Wartemodus versetzen
```

Wenn eine aktualisierende Wiederherstellung bis zum Ende aller Protokolle ausgeführt worden ist, ist die Sicherungs-ID das Ende der Zeitspanne; d. h.,die Sicherungs-ID weist den Wert 99991231235959 auf.

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. Das Löschen von Protokolldateien aus dem Pfad für aktive Protokolldateien ist möglicherweise erforderlich, wenn Sie Protokolle manuell archivieren (statt mit einem Benutzer-Exit-Programm).

Berechtigung

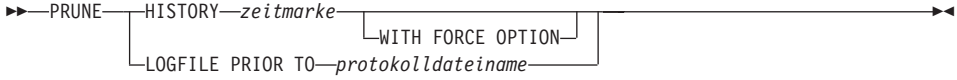
Eine der folgenden Berechtigungen:

- *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 gleich der angegebenen Zeitmarke oder kleiner als diese 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 Sicherungsimage. Wenn Sie diesen Parameter nicht angeben, bleiben alle Einträge aus der Weiterleitung des Sicherungsimage im Protokoll.

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) erstellt haben, aus der Datei des Wiederherstellungsprotokolls, geben Sie Folgendes ein:

```
db2 prune history 199412
```

Anmerkung: 199412 wird als 19941201000000 interpretiert.

Hinweise

Das Löschen von Sicherungseinträgen aus der Protokolldatei bewirkt, dass zugehörige Dateisicherungen auf DB2 Data Links Manager-Servern gelöscht werden.

REWIND TAPE

DB2 für Windows NT/2000 unterstützt Sicherungs- und Wiederherstellungsoperationen auf Datenstrombandeinheiten. Verwenden Sie diesen Befehl, um ein Band zurückzuspulen.

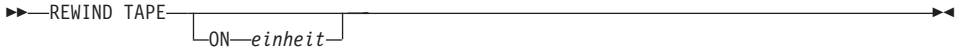
Berechtigung

Keine

Erforderliche Verbindung

Keine

Befehlssyntax



Befehlsparameter

ON einheit
Gibt einen gültigen Bandeinheitennamen an. Der Standardwert ist `\\.\TAPE0`.

Siehe auch

„INITIALIZE TAPE" auf Seite 365

„SET TAPE POSITION" auf Seite 372

SET TAPE POSITION

DB2 für Windows NT/2000 unterstützt Sicherungs- und Wiederherstellungsoperationen auf Datenstrombandeinheiten. Verwenden Sie diesen Befehl, um ein Band zu positionieren.

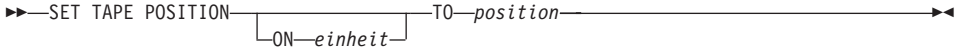
Berechtigung

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 NT/2000 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.

Siehe auch

„INITIALIZE TAPE" auf Seite 365

„REWIND TAPE" auf Seite 371

UPDATE HISTORY FILE

Aktualisiert die Position, den Einheitentyp oder den Kommentar in einem Protokolldateieintrag.

Berechtigung

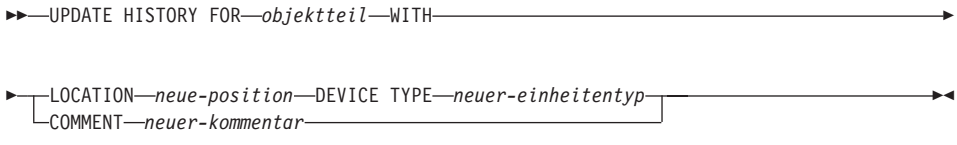
Eine der folgenden Berechtigungen:

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

Erforderliche Verbindung

Datenbank

Befehlssyntax



Befehlsparameter

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 Sicherungsimage an. Die Interpretation dieses Parameters hängt vom Einheitentyp ab.

DEVICE TYPE neuer-einheitentyp
Gibt einen neuen Einheitentyp zur Speicherung des Sicherungsimage an. Gültige Einheitentypen:

D Platte

K Diskette

T Band

A TSM

U Benutzer-Exit

O Sonstiges

COMMENT neuer-kommentar
Gibt einen neuen Kommentar mit der Beschreibung des Eintrags an.

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

Datenbankadministratoren verwenden die Protokolldatei zur Aufzeichnung. DB2 verwendet sie intern zur automatischen Wiederherstellung von Teilsicherungen.

Siehe auch

„PRUNE HISTORY/LOGFILE" auf Seite 369

Anhang D. Weitere APIs und zugehörige Datenstrukturen

db2ArchiveLog - Aktive Protokolldatei archivieren (API)

Schließt für eine wiederherstellbare Datenbank die aktive Protokolldatei und schneidet sie ab. Wenn der Benutzer-Exit aktiviert ist, wird eine Archivierungsanforderung abgesetzt.

Bereich

In einer MPP-Umgebung schließt diese API auf allen Knoten die aktiven Protokolle und schneidet diese Protokolle ab.

Berechtigung

Eine der folgenden Berechtigungen:

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

Erforderliche Verbindung

Mit dieser API stellen Sie automatisch eine Verbindung zur angegebenen Datenbank her. Sollte bereits eine Verbindung vorhanden sein, wird ein Fehler zurückgegeben.

API-Include-Datei

db2ApiDf.h

C-API-Syntax

```
/* File: db2ApiDf.h */
/* API: Archive Active Log */
/* ... */
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;
/* ... */
```

Generische API-Syntax

```
/* File: db2ApiDf.h */
/* API: Archive Active Log */
/* ... */
SQL_API_RC SQL_API_FN
   db2gArchiveLog (
    db2Uint32 version,
      void * pDB2gArchiveLogStruct,
   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;
} db2gArchiveLogStruct;
/* ... */
```

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*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

iAliasLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Aliasnamens der Datenbank in Byte darstellt.

iUserNameLen
Eingabe. Eine vier Byte lange ganze Zahl ohne Vorzeichen, die die Länge des Benutzernamens in Byte darstellt. 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 darstellt. Wenn kein Kennwort verwendet wird, ist dieser Parameter auf null gesetzt.

piDatabaseAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der Datenbank (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
Eingabe. Nur MPP. Diese Markierung zeigt an, ob die Operation für alle Knoten, die in der Datei `db2nodes.cfg` aufgelistet sind, gelten soll. Gültige Werte:

DB2ARCHIVELOG_ALL_NODES
Auf alle Knoten anwenden. (piNodeList muss NULL sein.) Dies ist der Standardwert.

DB2ARCHIVELOG_NODE_LIST
Auf alle Knoten anwenden, die in einer Knotenliste angegeben sind, welche in *piNodeList* übergeben wird.

DB2ARCHIVELOG_ALL_EXCEPT
Auf alle Knoten anwenden, *außer* denjenigen, die in einer Knotenliste angegeben sind, welche in *piNodeList* übergeben wird.

iNumNodes
Eingabe. Nur MPP. Gibt die Anzahl der Knoten in der Matrix *piNodeList* an.

piNodeList
Eingabe. Nur MPP. Ein Zeiger auf einen Bereich von Knotennummern, für die das Archivierungsprotokollieren angewendet werden soll.

iOptions
Eingabe. Zur zukünftigen Verwendung reserviert.

Hinweise

Diese API 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.

Sollten Transaktionen anderer Anwendungen in Bearbeitung sein, während Sie diese API aufrufen, vermindert sich die Leistung geringfügig, während der Protokollpuffer durch Schreiben auf die Platte geleert wird; andere Transaktionen, die versuchen, Protokollsätze in den Puffer zu schreiben, müssen warten, bis dieser Vorgang beendet ist.

Diese API bewirkt, dass die Datenbank einen Teil ihres Speicherbereichs für Protokollfolgenummern verliert. Dadurch werden gültige Protokollfolgenummern schneller knapp.

db2HistoryCloseScan - Suche in Datei des Wiederherstellungsprotokolls beenden (API)

Beendet das Durchsuchen einer Datei des Wiederherstellungsprotokolls und gibt DB2-Ressourcen frei, die für die Suche benötigt wurden.

Dieser API muss ein erfolgreicher Aufruf der API db2HistoryOpenScan (siehe „db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386) vorausgehen.

Berechtigung

Keine

Erforderliche Verbindung

Exemplar. Vor dem Aufruf dieser API müssen Sie nicht **sqleatin** aufrufen.

API-Include-Datei

db2ApiDf.h

C-API-Syntax

```
/* File: db2ApiDf.h */
/* API: Close Recovery History File Scan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2HistoryCloseScan (
    db2Uint32 version,
    void * piHandle,
    struct sqlca * pSqlca);
/* ... */
```

Generische API-Syntax

```
/* File: db2ApiDf.h */
/* API: Close Recovery History File Scan */
/* ... */
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, die von der API db2HistoryOpenScan zurückgegeben wurde (siehe „db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386).

pSqlca
Ausgabe. Ein Zeiger auf die Struktur *sqlca*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

REXX-API-Syntax

```
CLOSE RECOVERY HISTORY FILE :scanid
```

REXX-API-Parameter

scanid Host-Variable mit der Such-ID, die von OPEN RECOVERY HISTORY FILE SCAN zurückgegeben wurde.

Hinweise

Eine detaillierte Beschreibung der Verwendung der APIs für die Datei des Wiederherstellungsprotokolls finden Sie in „db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386.

Siehe auch

„db2HistoryGetEntry - Nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls abrufen (API)" auf Seite 382

„db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386

„db2Prune (API)" auf Seite 396

„db2HistoryUpdate - Datei des Wiederherstellungsprotokolls aktualisieren (API)" auf Seite 392

db2HistoryGetEntry - Nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls abrufen (API)

Ruft den nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls ab. Dieser API muss ein erfolgreicher Aufruf der API db2HistoryOpenScan (siehe „db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386) vorausgehen.

Berechtigung

Keine

Erforderliche Verbindung

Exemplar. Vor dem Aufruf dieser API müssen Sie nicht **sqleatin** aufrufen.

API-Include-Datei

db2ApiDf.h

C-API-Syntax

```
/* File: db2ApiDf.h */
/* API: Get Next Recovery History File Entry */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryGetEntry (
    db2Uint32 version,
    void * pDB2HistoryGetEntryStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2HistoryGetEntryStruct;
/* ... */
```

Generische API-Syntax

```
/* File: db2ApiDf.h */
/* API: Get Next Recovery History File Entry */
/* ... */
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 Struktur *sqlca*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

iHandle
Eingabe. Enthält die Kennung für den Suchzugriff, die von der API db2HistoryOpenScan zurückgegeben (siehe „db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386).

iCallerAction
Eingabe. Gibt die Art der auszuführenden Aktion an. Gültige Werte (in `db2ApiDf` 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*. Weitere Informationen zu dieser Struktur finden Sie in „Datenstruktur: db2HistData" auf Seite 403.

REXX-API-Syntax

```
GET RECOVERY HISTORY FILE ENTRY :scanid [USING :value]
```

REXX-API-Parameter

scanid Host-Variable mit der Such-ID, die von OPEN RECOVERY HISTORY FILE SCAN zurückgegeben wurde.

value Eine zusammengesetzte REXX-Host-Variable, in die Eintragsdaten der Datei des Wiederherstellungsprotokolls zurückgegeben werden. Im Folgenden steht XXX für den Namen der Host-Variablen:

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 (Art 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 neuesten 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 (siehe „db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386) angegeben wurden.

Eine detaillierte Beschreibung der Verwendung der APIs für die Datei des Wiederherstellungsprotokolls finden Sie in „db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386.

Siehe auch

„db2HistoryCloseScan - Suche in Datei des Wiederherstellungsprotokolls beenden (API)" auf Seite 380

„db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386

„db2Prune (API)" auf Seite 396

„db2HistoryUpdate - Datei des Wiederherstellungsprotokolls aktualisieren (API)" auf Seite 392

db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)

Startet das Durchsuchen einer Datei des Wiederherstellungsprotokolls.

Berechtigung

Keine

Erforderliche Verbindung

Exemplar. Vor dem Aufruf dieser API müssen Sie nicht **sqleatin** aufrufen. Sollte die Datenbank als fern katalogisiert sein, wird eine Exemplarverbindung zum fernen Knoten hergestellt.

API-Include-Datei

db2ApiDf.h

C-API-Syntax

```
/* File: db2ApiDf.h */
/* API: Open Recovery History File Scan */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryOpenScan (
    db2Uint32 version,
    void * pDB2HistoryOpenStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2HistoryOpenStruct;
/* ... */
```

Generische API-Syntax

```
/* File: db2ApiDf.h */
/* API: Open Recovery History File Scan */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryOpenScan (
    db2Uint32 version,
    void * pDB2GenHistoryOpenStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  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 Struktur *sqlca*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

piDatabaseAlias
Eingabe. Ein Zeiger auf eine Zeichenfolge, die den Aliasnamen der Datenbank 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 Datei des Wiederherstellungsprotokolls.

iCallerAction
Eingabe. Gibt die Art der auszuführenden Aktion an. Gültige Werte (in `db2ApiDf` 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 (siehe „db2HistoryGetEntry - Nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls abrufen (API)" auf Seite 382) mit der Aktion `DB2HISTORY_GET_DDL` sofort nach dem Abruf des Eintrags 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 (siehe „db2HistoryGetEntry - Nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls abrufen (API)" auf Seite 382) mit der Aktion `DB2HISTORY_GET_DDL` sofort nach dem Abruf des Eintrags aufgerufen werden.

DB2HISTORY_LIST_RUNSTATS
RUNSTATS-Operationen auflisten. Dieser Wert wird zum aktuellen Zeitpunkt nicht unterstützt.

DB2HISTORY_LIST_REORG
REORGANIZE TABLE-Operationen auflisten. Dieser Wert wird zum aktuellen Zeitpunkt nicht unterstützt.

oHandle
Ausgabe. Nach der Rückkehr von der API enthält dieser Parameter die Kennung für den Suchzugriff. Diese wird nachfolgend in der API db2HistoryGetEntry (siehe „db2HistoryGetEntry - Nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls abrufen (API)" auf Seite 382) und in der API db2HistoryCloseScan (siehe „db2HistoryCloseScan - Suche in Datei des Wiederherstellungsprotokolls beenden (API)" auf Seite 380) verwendet.

REXX-API-Syntax

```
OPEN [BACKUP] RECOVERY HISTORY FILE FOR database_alias
[OBJECT objname] [TIMESTAMP :timestamp]
USING :value
```

REXX-API-Parameter

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-Host-Variable, in die Daten der Datei des Wiederherstellungsprotokolls zurückgegeben werden. Im Folgenden steht XXX für den Namen der Host-Variablen:

XXX.0 Anzahl der Elemente in der Variablen (immer 2)

XXX.1 Bezeichner (Kennung) für den späteren Suchzugriff

XXX.2 Anzahl der passenden Einträge der Datei des Wiederherstellungsprotokolls

Hinweise

Die Kombination aus Zeitmarke, Objektname und aufrufender 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 sie 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 *db2HistData*-Struktur mit Speicherbereich für *n oTablespace*-Felder zuordnen, wobei *n* für eine willkürliche Zahl steht.
3. Das Feld *iDB2NumTablespace* der *db2HistData*-Struktur auf *n* setzen.
4. In einer Schleife Folgendes ausführen:
   - **db2HistoryGetEntry** für den Abruf aus der Protokolldatei aufrufen.
   - Wenn **db2HistoryGetEntry** den SQLCODE-Wert `SQL_RC_OK` zurückgibt, wird das Feld *sqld* der *db2HistData*-Struktur verwendet, 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; diesen freigeben und erneut der *db2HistData*-Struktur ausreichend Speicherbereich für *oDB2UsedTablespace* Tabellenbereichseinträge zuordnen sowie *iDB2NumTablespace* auf *oDB2UsedTablespace* setzen.
   - Wenn **db2HistoryGetEntry** einen SQLCODE-Wert von `SQLE_RC_NOMORE` zurückgibt, wurden alle Einträge der Datei des Wiederherstellungsprotokolls abgerufen.

- Alle anderen SQLCODE-Werte weisen auf ein Problem hin.

5. Sobald alle Daten abgerufen worden sind, die API db2HistoryCloseScan (siehe „db2HistoryCloseScan - Suche in Datei des Wiederherstellungsprotokolls beenden (API)" auf Seite 380) aufrufen, um Ressourcen freizugeben, die durch den Aufruf von **db2HistoryOpenScan** zugeordnet wurden.

Das Makro SQLUHINFOSIZE(*n*), das in `sqlutil` definiert ist, dient zur Bestimmung, wie viel Speicher für eine *db2HistData*-Struktur mit Speicherbereich für *n oTablespace*-Felder erforderlich ist.

Siehe auch

„db2HistoryCloseScan - Suche in Datei des Wiederherstellungsprotokolls beenden (API)" auf Seite 380

„db2HistoryGetEntry - Nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls abrufen (API)" auf Seite 382

„db2Prune (API)" auf Seite 396

„db2HistoryUpdate - Datei des Wiederherstellungsprotokolls aktualisieren (API)" auf Seite 392

db2HistoryUpdate - Datei des Wiederherstellungsprotokolls aktualisieren (API)

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

C-API-Syntax

```
/* File: db2ApiDf.h */
/* API: Update Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryUpdate (
    db2Uint32 version,
    void * pDB2HistoryUpdateStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2HistoryUpdateStruct;
/* ... */
```

Generische API-Syntax

```
/* File: db2ApiDf.h */
/* API: Update Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryUpdate (
    db2Uint32 version,
    void * pDB2GenHistoryUpdateStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2GenHistoryUpdateStruct;
/* ... */
```

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 Struktur *sqlca*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

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 Ladekopieimage 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.

iEID Eingabe. Eine eindeutige Kennung, anhand derer ein spezifischer Eintrag in der Protokolldatei aktualisiert wird.

REXX-API-Syntax

```
UPDATE RECOVERY HISTORY USING :value
```

REXX-API-Parameter

value Eine zusammengesetzte REXX-Host-Variable, die Daten zur neuen Position eines Eintrags in einer Datei des Wiederherstellungsprotokolls enthält. Im Folgenden steht XXX für den Namen der Host-Variablen:

XXX.0 Anzahl der Elemente in der Variablen (muss zwischen 1 und 4 liegen)

XXX.1 OBJECT_PART (Zeitmarke mit der 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 Protokolldatei wird nur zum Aufzeichen verwendet. Sie wird nicht direkt von den Funktionen zur Wiederherstellung oder aktualisierenden Wiederherstellung verwendet. Bei einer Wiederherstellungsoperation kann die Position des Sicherungsimage angegeben werden, und die Protokolldatei ist zum Verfolgen dieser Position nützlich.

Die Daten können daran anschließend in das Sicherungsdienstprogramm eingegeben werden (siehe „API zur Datenbanksicherung" auf Seite 109). Ähnliches gilt, wenn die Position eines Ladekopieimage versetzt wird: dem Dienstprogramm zur aktualisierenden Wiederherstellung muss die neue Position und der Typ des Datenträgers angegeben werden. Weitere Informationen finden Sie in „API zur aktualisierenden Wiederherstellung von Datenbanken" auf Seite 178.

Siehe auch

„db2HistoryCloseScan - Suche in Datei des Wiederherstellungsprotokolls beenden (API)" auf Seite 380

„db2HistoryGetEntry - Nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls abrufen (API)" auf Seite 382

„db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386

„db2Prune (API)" auf Seite 396

db2Prune (API)

Löscht Einträge aus der Datei des Wiederherstellungsprotokolls oder aus den Protokolldateien aus dem aktiven Protokollpfad.

Berechtigung

Eine der folgenden Berechtigungen:

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

Erforderliche Verbindung

Datenbank. Zum Löschen von Einträgen aus der Datei des Wiederherstellungsprotokolls 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

C-API-Syntax

```
/* File: db2ApiDf.h */
/* API: Prune Recovery History File */
/* ... */
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;
/* ... */
```

Generische API-Syntax

```
/* File: db2ApiDf.h */
/* API: Prune Recovery History File */
/* ... */
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 Struktur *sqlca*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

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 gleich der Zeitmarke oder kleiner als diese 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 die Art der auszuführenden Aktion an. Gültige Werte (in db2ApiDf definiert):

DB2PRUNE_ACTION_HISTORY
Protokolldateieinträge entfernen.

DB2PRUNE_ACTION_LOG
Protokolldateien aus dem Pfad für aktive Protokolldateien entfernen.

iOptions
Eingabe. Gültige Werte (in db2ApiDf definiert):

DB2PRUNE_OPTION_FORCE
Das Entfernen der letzten Sicherung erzwingen.

DB2PRUNE_OPTION_LSNSTRING
Angeben, dass der Wert *piString* eine Protokollfolgenummer ist, die verwendet wird, wenn die Aktion DB2PRUNE_ACTION_LOG angegeben wird.

REXX-API-Syntax

```
PRUNE RECOVERY HISTORY BEFORE :timestamp [WITH FORCE OPTION]
```

REXX-API-Parameter

timestamp
Eine Host-Variable, die eine Zeitmarke enthält. Alle Einträge mit Zeitmarken kleiner als die angegebene Zeitmarke oder gleich dieser Marke werden aus der Datei des Wiederherstellungsprotokolls gelöscht.

WITH FORCE OPTION
Wenn dieser Parameter angegeben ist, wird die Datei des Wiederherstellungsprotokolls 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 als die Zeitmarke oder 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 dem Katalogtabellenbereich und den Benutzertabellenbereichen, gesichert werden. Andernfalls erhalten Sie eine Datenbank, die Sie nicht wiederherstellen können, oder ein Teil der Benutzerdaten in der Datenbank geht verloren.

Siehe auch

„db2HistoryCloseScan - Suche in Datei des Wiederherstellungsprotokolls beenden (API)" auf Seite 380

„db2HistoryGetEntry - Nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls abrufen (API)" auf Seite 382

„db2HistoryOpenScan - Suche in Datei des Wiederherstellungsprotokolls starten (API)" auf Seite 386

„db2HistoryUpdate - Datei des Wiederherstellungsprotokolls aktualisieren (API)" auf Seite 392

sqlurlog - Protokolldaten asynchron lesen (API)

Ermöglicht dem aufrufenden Programm, bestimmte Protokollsätze aus den Datenbankprotokollen zu extrahieren und den Protokollmanager nach aktuellen Protokollstatusinformationen abzufragen. 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-Include-Datei

sqlutil.h

C-API-Syntax

```
/* File: sqlutil.h */
/* API: Asynchronous Read Log */
/* ... */
SQL_API_RC SQL_API_FN
  sqlurlog (
    sqluint32 CallerAction,
    SQLU_LSN * pStartLsn,
    SQLU_LSN * pEndLsn,
    char * pLogBuffer,
    sqluint32 LogBufferSize,
    SQLU_RLOG_INFO * pReadLogInfo,
    struct sqlca * pSqlca);
/* ... */
```

API-Parameter

CallerAction
Eingabe. Gibt die auszuführende Aktion an.

SQLU_RLOG_READ
Liest das Datenbankprotokoll von der Protokollstartfolgenummer bis zur Protokollendefolgenummer und gibt alle Protokollsätze innerhalb dieses Bereichs zurück, die weitergegeben werden können.

SQLU_RLOG_READ_SINGLE
Liest einen einzelnen Protokollsatz (unabhängig davon, ob er weitergegeben werden kann). Dieser Satz wird mit der Protokollstartfolgenummer angegeben.

SQLU_RLOG_QUERY
Fragt das Datenbankprotokoll ab. Ergebnisse der Abfrage werden über die Struktur SQLU_RLOG_INFO zurückgesendet (siehe „Datenstruktur: SQLU-RLOG-INFO" auf Seite 410).

pStartLsn
Eingabe. Die Protokollstartfolgenummer gibt die relative Bytestartadresse für das Lesen des Protokolls an. Dieser Wert muss der Anfang eines tatsächlichen Protokollsatzes sein.

pEndLsn
Eingabe. Die Protokollendefolgenummer 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.

pLogBuffer
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.

LogBufferSize
Ausgabe. Gibt die Größe des Protokollpuffers in Byte an.

pReadLogInfo
Ausgabe. Eine Struktur mit zusätzlichen Informationen zum Aufruf und zum Datenbankprotokoll. Weitere Informationen zu dieser Struktur finden Sie in „Datenstruktur: SQLU-RLOG-INFO" auf Seite 410.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur *sqlca*. Weitere Informationen zu dieser Struktur finden Sie im Handbuch *Administrative API Reference* oder im Handbuch *SQL Reference*.

Beispielprogramme

C \sqllib\samples\c\asynrlog.sqc

Hinweise

Wenn die angeforderte Aktion das Protokoll lesen soll, stellt das aufrufende Programm einen Protokollfolgenummernbereich und einen Puffer zum Speichern der Protokollsätze bereit. Die API liest das Protokoll sequenziell im angeforderten Bereich von Protokollfolgenummern und gibt Protokollsätze zurück, die Tabellen zugeordnet sind, für die DATA CAPTURE auf CHANGES gesetzt ist. Außerdem gibt die API die Struktur SQLU_RLOG_INFO mit den aktuellen Daten des aktiven Protokolls zurück. Wenn die angeforderte Aktion eine Abfrage ist, gibt die API die Struktur SQLU_RLOG_INFO mit den aktuellen Daten des aktiven Protokolls zurück.

Zum asynchronen Lesen von Protokolldaten müssen Sie zuerst das Datenbankprotokoll nach einer gültigen Protokollstartfolgenummer abfragen. Nach dem Abfrageaufruf enthält die Informationsstruktur zum Lesen von Protokolldaten (SQLU-RLOG-INFO) eine gültige Protokollstartfolgenummer (im Element initialLSN), die in einem Leseaufruf verwendet werden soll. Das Ende des aktuellen aktiven Protokolls bildet das Element curActiveLSN der Informationsstruktur zum Lesen von Protokolldaten. Der Wert, der bei einem Lesevorgang als Protokollendefolgenummer verwendet wird, kann einer der folgenden Werte sein:

- Der Wert von curActiveLSN
- 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.

Weitere Informationen zur Informationsstruktur zum Lesen von Protokolldaten Struktur finden Sie in „Datenstruktur: SQLU-RLOG-INFO" auf Seite 410.

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 **sqlurlog** zurückgegeben werden, finden Sie im Handbuch *Administrative API Reference*.

Zum Lesen des nächsten sequenziellen Protokollsatzes nach dem ersten Lesen addieren Sie eins zur zuletzt gelesenen Protokollfolgenummer, die in SQLU-RLOG-INFO zurückgegeben wurde. Wiederholen Sie den Aufruf mit dieser neuen Protokollstartfolgenummer und einer gültigen Protokollendefolgenummer. 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.

Datenstruktur: db2HistData

Diese Struktur können Sie verwenden, um Daten nach einem Aufruf von db2HistoryGetEntry zurückzugeben (siehe „db2HistoryGetEntry - Nächsten Eintrag aus der Datei des Wiederherstellungsprotokolls abrufen (API)" auf Seite 382).

Tabelle 17. 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 angeben, 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 Zeitmarke, kombiniert mit der Folgenummer, muss eindeutig sein.
oEndTime	db2Char	Eine Zeitmarke mit dem Format *jjjjmmtthhnnss*, die angibt, wann die Operation endete.
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

Tabelle 17. Felder in der Struktur db2HistData (Forts.)

Feldname	Datentyp	Beschreibung
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 Folgenummer, die durch *oObjectPart* definiert ist, 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 Datenträger-Header • Für TSM (`A`) als Servername • Für ein Benutzer-Exit oder für Sonstiges (`U` oder `O`) als Text mit freiem Format
oComment	db2Char	Kommentartext in freiem Format.
oCommandText	db2Char	Befehlstext oder DDL.
oLastLSN	SQLU_LSN	Letzte Protokollfolgenummer.
oEID	Struktur	Eindeutige Eintragskennung.
poEventSQLCA	Struktur	Ergebnis-*sqlca* des aufgezeichneten Ereignisses. Weitere Informationen zur Struktur *sqlca* finden Sie im Handbuch *Administrative API Reference* und im Handbuch *SQL Reference*.

Tabelle 17. Felder in der Struktur db2HistData (Forts.)

Feldname	Datentyp	Beschreibung
poTablespace	db2Char	Eine Liste von Tabellenbereichsnamen.
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 den gesicherten oder wiederhergestellten Tabellenbereichs als 18 Zeichen lange Zeichenfolge. In jeder Zeile wird ein Name eines Tabellenbereichs angezeigt.
oOperation	char	Siehe Tabelle 18.
oObject	char	Granularität der Operation: `D` für vollständige Datenbank, `P` für Tabellenbereich und `T` für Tabelle.
oOptype	char	Siehe Tabelle 19 auf Seite 406.
oStatus	char	Eintragsstatus: `D` für gelöscht (zukünftige Verwendung), `E` für abgelaufen, `I` für inaktiv, `N` für noch nicht festgeschrieben und `Y` für festgeschrieben oder aktiv.
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, `S` für Server, `T` für Band und `U` für Benutzer-Exit.

Tabelle 18. 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

Tabelle 18. Gültige oOperation-Werte in der Struktur db2HistData (Forts.)

Wert	Beschreibung	C-Definition	COBOL/FORTRAN-Definition
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
S	Statistik ausführen	DB2HISTORY_OP_RUNSTATS	DB2HIST_OP_RUNSTATS
T	Tabellenbereich ändern	DB2HISTORY_OP_ALT_ TABLESPACE	DB2HIST_OP_ALT_TBS
U	aus Tabelle laden	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD

Tabelle 19. 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 off-line	DB2HISTORY_OPTYPE_INCR_ OFFLINE
	O	inkrementell online	DB2HISTORY_OPTYPE_INCR_ ONLINE
	D	delta offline	DB2HISTORY_OPTYPE_DELTA_ OFFLINE
	V	delta online	DB2HISTORY_OPTYPE_DELTA_ ONLINE
F	V	Ende der Proto-kolle	DB2HISTORY_OPTYPE_EOL
	P	zu bestimmtem Zeitpunkt	DB2HISTORY_OPTYPE_PIT
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

Tabelle 19. Gültige oOptype-Werte in der Struktur db2HistData (Forts.)

oOperation	oOptype	Beschreibung	C/COBOL/FORTRAN-Definition
R	F	offline	DB2HISTORY_OPTYPE_OFFLINE
	N	online	DB2HISTORY_OPTYPE_ONLINE
	I	inkrementell off-line	DB2HISTORY_OPTYPE_INCR_ OFFLINE
	O	inkrementell online	DB2HISTORY_OPTYPE_INCR_ ONLINE
T	C	Behälter hinzufügen	DB2HISTORY_OPTYPE_ADD_CONT
	R	neu ausgleichen	DB2HISTORY_OPTYPE_REB

Tabelle 20. 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 21. Felder in der Struktur db2HistoryEID

Feldname	Datentyp	Beschreibung
ioNode	SQL_PDB_NODE_TYPE	Knotennummer.
ioHID	db2Uint32	ID des Eintrags in der lokalen Protokolldatei.

Datenstruktur: db2HistData

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;
/* ... */
```

Datenstruktur: SQLU-LSN

Diese Gesamtverknüpfung wird von der API sqlurlog (siehe „sqlurlog - Protokolldaten asynchron lesen (API)" auf Seite 400) 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 22. 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;
```

Datenstruktur: SQLU-RLOG-INFO

Diese Struktur enthält Daten zum Status von Aufrufen der API sqlurlog (siehe „sqlurlog - Protokolldaten asynchron lesen (API)" auf Seite 400) und des Datenbankprotokolls.

Tabelle 23. Felder in der Struktur SQLU-RLOG-INFO

Feldname	Datentyp	Beschreibung
initialLSN	SQLU_LSN	Gibt den Wert der Protokollfolgenummer des ersten Protokollsatzes an, welcher nach dem Absetzen der ersten Datenbank-CONNECT-Anweisung geschrieben wird. Weitere Informationen finden Sie in „Datenstruktur: SQLU-LSN" auf Seite 409.
firstReadLSN	SQLU_LSN	Gibt den Wert der Protokollfolgenummer des ersten gelesenen Protokollsatzes an.
lastReadLSN	SQLU_LSN	Gibt den Wert der Protokollfolgenummer des letzten gelesenen Protokollsatzes an.
curActiveLSN	SQLU_LSN	Gibt den Wert der Protokollfolgenummer des aktuellen (aktiven) Protokolls an.
logRecsWritten	sqluint32	Gibt die Anzahl von Protokollsätzen an, die in den Puffer geschrieben werden.
logBytesWritten	sqluint32	Gibt die Zahl der Byte an, die in den Puffer geschrieben werden.

Sprachsyntax

C-Struktur

```
typedef SQL_STRUCTURE SQLU_RLOG_INFO
{
SQLU_LSN       initialLSN ;
SQLU_LSN       firstReadLSN ;
SQLU_LSN       lastReadLSN ;
SQLU_LSN       curActiveLSN ;
sqluint32      logRecsWritten ;
sqluint32      logBytesWritten ;
} SQLU_RLOG_INFO;
```

Anhang E. Beispiele für Wiederherstellungsprogramme

Beispielprogramm ohne eingebettetes SQL (backrest.c)

Das folgende Beispielprogramm zeigt, wie Sie DB2-Sicherungs- und -Wiederherstellungs-APIs verwenden können:

- Sichern einer Datenbank
- Wiederherstellen der Datenbank
- Aktualisierendes Wiederherstellen der Datenbank

Zusatzinformationen zur Beispieldatenbank SAMPLE finden Sie im Handbuch *SQL Reference*.

Die Quellendatei für dieses Beispielprogramm (`backrest.c`) befindet sich bei Windows-Betriebssystemen bzw. OS/2 im Verzeichnis `\sqllib\samples\c`. Bei UNIX-Systemen befindet sie sich im Verzeichnis `/sqllib/samples/c`. Dieses Beispielprogramm enthält eine Reihe von DB2-APIs. Die Datei `makefile`, die sich im gleichen Verzeichnis befindet, enthält die Befehle zur Erstellung dieses und anderer Beispielprogramme. Basisinformationen zum Erstellen von Anwendungen, die DB2-Verwaltungs-APIs enthalten, und Zusatzinformationen zu Kompilierungs- und Programmverbindungsoptionen finden Sie im Handbuch *Application Building Guide*. Gehen Sie wie folgt vor, um unter Windows NT oder Windows 2000 das Beispielprogramm `backrest` aus der Quellendatei `backrest.c` zu erstellen:

1. Kopieren Sie die Dateien `backrest.c`, `makefile`, `utilapi.c` und `utilapi.h` in ein Arbeitsverzeichnis.
2. Wenn der Datenbankmanager nicht aktiv ist, setzen Sie in einem DB2-Befehlsfenster den Befehl **db2start** ab. Zum Öffnen eines DB2-Befehlsfensters und zur Initialisierung der DB2-Befehlszeilenumgebung unter dem Windows-Betriebssystem setzen Sie in einer Befehlszeile den Befehl **db2cmd** ab.
3. Geben Sie `nmake backrest` ein. Die folgenden Dateien werden generiert:

```
backrest.exe
backrest.ilk
backrest.obj
backrest.pdb
utilapi.obj
```

Beispielprogramm ohne eingebettetes SQL (backrest.c)

Geben Sie Folgendes ein, um das Beispielprogramm (ausführbare Datei) auszuführen:

```
backrest datenbank benutzer-id kennwort
```

Zum Beispiel:

```
backrest sample db2admin db2admin
```

Im Folgenden sehen Sie ein Beispiel für die Ausgabe dieses Programms:

```
This is sample program : backrest.c

NOTE:   Ensure the database is not in use prior to running this program.

Updating the sample database configuration parameter LOGRETAIN
to 'ON' to enable rollforward recovery.

Backing up the 'sample' database.
The database has been successfully backed up.

Updating the sample database configuration parameter LOGRETAIN to 'OFF'.

Restoring the database 'sample' as 'TESTBACK' (1st pass).
Should get returned value = SQLUD_INACCESSABLE_CONTAINER.
SQLUD_INACCESSABLE_CONTAINER is returned.

Need to SET TABLESPACES CONTAINERS
Tablespace container information for tablespace 1 obtained.
Tablespace containers have been set for tablespace 1.

Restoring the database 'sample' as 'TESTBACK' (2nd pass).
Database sample has been restored as 'TESTBACK'.

The database 'TESTBACK' has been successfully rolled forward.
The database 'TESTBACK' has been successfully dropped.
```

Im Folgenden sehen Sie das Listing des Quellcodes für das Beispielprogramm:

```
/*****************************************************************************
**
** Source File Name = backrest.c
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1995, 2000
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**    PURPOSE :
**       an example showing how to use BACKUP & RESTORE APIs in order to:
**          - backup a database
**          - restore a database
**          - roll forward database to return the database to a state it was
```

```
**           in prior to the damage...
**
**    APIs USED :
**       BACKUP DATABASE                   sqlubkp()
**       RESTORE DATABASE                  sqlurestore()
**       UPDATE DATABASE CONFIGURATION     sqlfudb()
**       GET TABLESPACE CONTAINER QUERY    sqlbtcq()
**       SET TABLESPACE CONTAINERS         sqlbstsc()
**       ROLLFORWARD DATABASE              sqluroll()
**
**    STRUCTURES USED :
**       sqlu_tablespace_bkrst_list
**       SQLB_TBSCONTQRY_DATA
**       sqlu_media_list
**       rfwd_input
**       rfwd_output
**       sqlurf_info
**       sqlca
**    OTHER FUNCTIONS DECLARED :
**       'C' COMPILER LIBRARY :
**          stdio.h  -  printf
**
**          internal :
**
**          external :
**             check_error :     Checks for SQLCODE error, and prints out any
**             [in UTIL.C]       related information available.
**                               This procedure is located in the UTIL.C file.
**
**    EXTERNAL DEPENDENCIES :
**       - Ensure existence of database for precompile purposes.
**       - Compile and link with the IBM Cset++ compiler (AIX and OS/2)
**         or the Microsoft Visual C++ compiler (Windows)
**         or the compiler supported on your platform.
**
** For more information about these samples see the README file.
** For more information on programming in C, see the:
**   -  "Programming in C and C++" section of the Application Development Guide
** For more information on building C applications, see the:
**   -  "Building C Applications" section of the Application Building Guide.
**
** For more information on the SQL language see the SQL Reference.
**
******************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlutil.h>
#include <sqlenv.h>
#include "utilapi.h"

/* You can change the following path to another directory for which */
/* you have permission. create the "backup" directory in the same path. */
```

Beispielprogramm ohne eingebettetes SQL (backrest.c)

```
   #define TEMPDIR "."

int main (int argc, char *argv[]) {

   /* Variables for the BACKUP API */
   sqluint32 buff_size;
   sqluint32 num_buff;
   struct sqlu_tablespace_bkrst_list *tablespace_list;
   struct SQLB_TBSCONTQRY_DATA *cont;
   struct sqlca sqlca;
   struct sqlu_media_entry media_entry;
   struct sqlu_media_list media_list;
   char applid[SQLU_APPLID_LEN+1];
   char timestamp[SQLU_TIME_STAMP_LEN+1];
   char *target_path;

   sqluint32 tsc_id;
   sqluint32 num_cont;

   /* variables for the Update Database Configuration API */
   struct sqlfupd itemList;
   short log_retain;

   /* variables for the ROLLFORWARD API */
   struct rfwd_input rfwdInput;
   char dbAlias[] = "TESTBACK";
   char overFlowLogPath[SQL_PATH_SZ] = TEMPDIR;

   struct rfwd_output rfwdOutput;
   sqlint32   numReplies;
   struct sqlurf_info nodeInfo;

   printf ("\nThis is sample program : backrest.c\n\n");

   printf ("NOTE:   Ensure the database is not in use prior to running
    this program.\n\n");

   if (argc != 4) {
      printf ("\nUSAGE: backrest database userID password\n\n");
      return 1;
   } /* endif */

   /* Altering the default Database Configuration to enable rollforward
      recovery of the TESTBACK database. */

   log_retain = 1;
   itemList.token = SQLF_DBTN_LOG_RETAIN;

   itemList.ptrvalue = (char *) &log_retain;


   printf ("Updating the %s database configuration parameter
    LOGRETAIN \n", argv[1]);
   printf ("to 'ON' to enable rollforward recovery.\n\n");
```

```
/*****************************************\
* UPDATE DATABASE CONFIGURATION API called *
\*****************************************/
sqlfudb(argv[1], 1, &itemList, &sqlca);
API_SQL_CHECK("updating the database configuration");

/* Initialize the BACKUP API variables */
buff_size = 16;
num_buff = 1;
tablespace_list = NULL;
media_list.media_type = 'L';
media_list.sessions = 1;

strcpy (media_entry.media_entry, TEMPDIR);

media_list.target.media = &media_entry;
printf ("Backing up the '%s' database.\n", argv[1]);
/*****************\
* BACKUP DATABASE *
\*****************/
sqlubkp (argv[1],
          buff_size,
          SQLUB_OFFLINE,
          SQLUB_FULL,
          SQLUB_BACKUP,
          applid,
          timestamp,
          num_buff,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
          NULL,
          0,
          NULL,
          1,
          NULL,
          NULL,
          NULL,
          &sqlca);

API_SQL_CHECK("backing up the database");
printf ("The database has been successfully backed up.\n\n");



/* Altering the default Database Configuration to disable rollforward
   recovery of the TESTBACK database. */

log_retain = 0;
itemList.token = SQLF_DBTN_LOG_RETAIN;
itemList.ptrvalue = (char *) &log_retain;


printf ("Updating the %s database configuration parameter
```

```
 LOGRETAIN \n", argv[1]);
printf ("to 'OFF'.\n\n");

/*****************************************\
* UPDATE DATABASE CONFIGURATION API called *
\*****************************************/
sqlfudb(argv[1], 1, &itemList, &sqlca);
API_SQL_CHECK("updating the database configuration");


/* Initialize the variables for the RESTORE API */
buff_size = 1024;
target_path = NULL;
printf ("Restoring the database '%s' as 'TESTBACK' (1st pass).\n", argv[1]);
printf ("Should get returned value = SQLUD_INACCESSABLE_CONTAINER.\n");
/******************\
* RESTORE DATABASE *
\******************/
sqlurestore(argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_RESTORE_STORDEF,
          applid,
          timestamp,
          target_path,
          num_buff,
          NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
          NULL,
          0,
          NULL,
          1,
          NULL,
          NULL,
          NULL,
          &sqlca);
if (sqlca.sqlcode != SQLUD_INACCESSABLE_CONTAINER)
   API_SQL_CHECK("restoring database");

printf ("SQLUD_INACCESSABLE_CONTAINER is returned.\n\n");
printf ("Need to SET TABLESPACES CONTAINERS\n");
/* Set the containers structure to a new list of containers */
tsc_id = 1;
cont = (struct SQLB_TBSCONTQRY_DATA *) malloc
   (sizeof(struct SQLB_TBSCONTQRY_DATA));

/*******************************\
* GET TABLESPACE CONTAINER QUERY *
```

```
\*******************************/
sqlbtcq (&sqlca, tsc_id, &num_cont, &cont);
API_SQL_CHECK("GET TABLESPACE CONTAINER QUERY");
printf ("Tablespace container information for tablespace 1 obtained.\n");
/**************************\
* SET TABLESPACE CONTAINERS *
\**************************/
sqlbstsc (&sqlca,
          SQLB_SET_CONT_INIT_STATE,
          tsc_id,
          num_cont,
          cont);
API_SQL_CHECK("SET TABLESPACE CONTAINERS");
printf ("Tablespace containers have been set for tablespace 1.\n\n");

printf ("Restoring the database '%s' as 'TESTBACK' (2nd pass).\n", argv[1]);
/******************\
* RESTORE DATABASE *
\******************/
sqlurestore (argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_CONTINUE,
          applid,
          timestamp,
          target_path,
          num_buff,
          NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
          NULL,
          0,
          NULL,
          1,
          NULL,
          NULL,
          NULL,
          &sqlca);
API_SQL_CHECK("restoring database 2");
printf ("Database %s has been restored as 'TESTBACK'.\n\n", argv[1]);


/**********************\
* ROLLFORWARD DATABASE *
\**********************/
rfwdInput.version=SQLUM_RFWD_VERSION;
rfwdInput.pDbAlias=dbAlias;
rfwdInput.CallerAction=SQLUM_ROLLFWD;
rfwdInput.pStopTime=SQLUM_INFINITY_TIMESTAMP;
```

```
    rfwdInput.pUserName=argv[2];
    rfwdInput.pPassword=argv[3];
    rfwdInput.pOverflowLogPath=overFlowLogPath;
    rfwdInput.NumChngLgOvrflw=0;
    rfwdInput.pChngLogOvrflw=NULL;
    rfwdInput.ConnectMode=SQLUM_OFFLINE;
    rfwdInput.pTablespaceList=NULL;
    rfwdInput.AllNodeFlag= SQLURF_ALL_NODES;
    rfwdInput.NumNodes=0;
    rfwdInput.pNodeList=NULL;
    rfwdInput.NumNodeInfo=1;
    rfwdInput.DlMode=0;
    rfwdInput.pReportFile=NULL;
    rfwdInput.pDroppedTblID=NULL;
    rfwdInput.pExportDir=NULL;


    rfwdOutput.pApplicationId=applid;
    rfwdOutput.pNumReplies=&numReplies;
    rfwdOutput.pNodeInfo=&nodeInfo;


    sqluroll( &rfwdInput,
              &rfwdOutput,
              &sqlca);
    API_SQL_CHECK("rolling database forward");
    printf ("The database 'TESTBACK' has been successfully rolled
     forward.\n\n", argv[1]);


    /***************\
    * DROP DATABASE *
    \***************/
    sqledrpd ("TESTBACK",
              &sqlca);
    API_SQL_CHECK("DROP DATABASE");
    printf ("The database 'TESTBACK' has been successfully dropped.\n");


    return 0;
}
```

Beispielprogramm mit eingebettetem SQL (dbrecov.sqc)

Das folgende Beispielprogramm zeigt, wie Sie DB2-Sicherungs- und -Wiederherstellungs-APIs verwenden können:

- Sichern einer Datenbank
- Wiederherstellen der Datenbank
- Aktualisierendes Wiederherstellen der Datenbank

Zusatzinformationen zur Beispieldatenbank SAMPLE finden Sie im Handbuch *SQL Reference*.

Anmerkung: Die `dbrecov`-Beispieldateien werden bei DB2 Version 7.2 nicht installiert. Diese Dateien sind nur zum Herunterladen aus dem Web verfügbar.

Gehen Sie wie folgt vor, um die erforderlichen Dateien für das Beispielprogramm `dbrecov` herunterzuladen:

1. Besuchen Sie die Webseite `http://www.ibm.com/software/data/db2/udb/ad/v7/abg.html`.
2. Klicken Sie unter „Recent Updates" den Link „dbrecov sample files by platform" an.
3. Wählen Sie auf der angezeigten Seite den Link zum Herunterladen der Dateien für die entsprechende Plattform aus. Für Windows-Betriebssysteme und für OS/2 befinden sich die Dateien in einer komprimierten ausführbaren zip-Datei. Für UNIX-Systeme sind die Dateien in einer tar.gz-Datei enthalten.

Achtung:
Führen Sie diese ausführbare Datei nicht in dem Verzeichnis aus, in dem sich bereits die Beispieldateien von DB2 UDB Version 7 befinden, da die vorhandenen Versionen der DB2 UDB Version 7-Dienstprogrammdateien mit den neuen Dateien überschrieben werden. Die neuen Dienstprogrmmdateien sind zu anderen Beispielprogrammen inkompatibel.

Die Quellendatei für dieses Beispielprogramm (`dbrecov.sqc`) enthält sowohl DB2-APIs als auch eingebettete SQL-Aufrufe. Basisinformationen zum Erstellen von Anwendungen, die DB2-Verwaltungs-APIs enthalten, und Zusatzinformationen zu Kompilierungs- und Programmverbindungsoptionen finden Sie im Handbuch *Application Building Guide*.

Beispielprogramm mit eingebettetem SQL (dbrecov.sqc)

Im Folgenden finden Sie Anweisungen zum Erstellen und Ausführen des Programms dbrecov unter dem Windows-Betriebssystem. Wenn Sie das Programm unter anderen unterstützten Betriebssystemen generieren möchten, finden Sie weitere Informationen in der README-Datei, die den Beispieldateien beigefügt ist.

1. Führen Sie die heruntergeladene ausführbare Datei dbrecov_win.exe im Arbeitsverzeichnis aus. Dadurch werden die folgenden Dateien extrahiert:

   ```
   dbrecov.sqc - Quellendatei für eingebettetes SQL für das Programm dbrecov
   utilapi.c   - Fehlerprüfungsdatei für DB2-API-Programme
   utilapi.h   - Header-Datei für utilapi.c
   utilemb.sqc - Fehlerprüfungsdatei für eingebettete SQL-Programme
   utilemb.h   - Header-Datei für utilemb.sqc
   bldmapp.bat - Erstellt Microsoft Visual C++-Anwendungsprogramme
   bldvapp.bat - Erstellt VisualAge C++-Anwendungsprogramme
   embprep.bat - Kompiliert eingebettete SQL-Programme und bindet diese
   README      - Enthält Informationen zu den Programmdateien
   ```

2. Wenn der Datenbankmanager nicht aktiv ist, setzen Sie in einem DB2-Befehlsfenster den Befehl **db2start** ab. Zum Öffnen eines DB2-Befehlsfensters und zur Initialisierung der DB2-Befehlszeilenumgebung unter dem Windows-Betriebssystem setzen Sie in einer Befehlszeile den Befehl **db2cmd** ab.

3. Geben Sie je nach Compiler bldmapp dbrecov oder bldvapp dbrecov ein. Die folgenden Dateien werden generiert:

   ```
   dbrecov.exe
   dbrecov.ilk
   dbrecov.c
   dbrecov.obj
   dbrecov.bnd
   dbrecov.pdb
   utilemb.c
   utilemb.obj
   ```

Geben Sie Folgendes ein, um das Beispielprogramm (ausführbare Datei) auszuführen:

```
dbrecov
```

Die Ausgabe variiert abhängig von der Betriebssystemumgebung. Im Folgenden sehen Sie ein Beispiel für die Ausgabe dieses Programms auf einem Windows-System:

```
THIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.

USE THE DB2 API:
  sqlfxdb -- Get Database Configuration
TO GET THE DATABASE CONFIGURATION AND DETERMINE
THE SERVER WORKING PATH.

NOTE: Backup images will be created on the server
      in the directory D:\DB2,
      and will not be deleted by the program.
```

```
*************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
*************************************

USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
  CONNECT  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.

  Verbindung zur Datenbank 'sample' wird hergestellt...
  Verbindung zur Datenbank 'sample' hergestellt.

  Prune the recovery history file for 'sample' database.

  Verbindung zur Datenbank 'sample' wird abgebrochen...
  Verbindung zur Datenbank 'sample' abgebrochen.

*************************************
*** BACK UP AND RESTORE A DATABASE ***
*************************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlurestore -- Restore Database
TO BACK UP AND RESTORE A DATABASE.

  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162032

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162032
    - target database        : sample

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database restore -- start
line                = 506
file                = dbrecov.sqc
SQLCODE             = 2539

SQL2539W Achtung! Wiederherstellung in eine bestehende Datenbank, die mit der
Datenbank des Sicherungsimage identisch ist. Die Datenbankdateien werden gelöscht.
```

```
---- end warning report ----------------------

  Continuing the restore operation...

  Restore finished.

*************************
*** REDIRECTED RESTORE ***
*************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
  sqlbmtsq -- Tablespace Query
  sqlbtcq -- Tablespace Container Query
  sqlbstsc -- Set Tablespace Containers
  sqlefmem -- Free Memory
  sqledrpd -- Drop Database
TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.

  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162125

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162125
    - target database        : RRDB

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database restore -- start
line                = 776
file                = dbrecov.sqc
SQLCODE             = 1277

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.

---- end warning report ----------------------

  Continuing the restore operation...

  Find and redefine inaccessible containers.
```

```
    Redefine inaccessible container:
      - table space name: SYSCATSPACE
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0000.0
      - container type: path
      - new container name: D:\DB2\SQLT0000.0

    Redefine inaccessible container:
      - table space name: TEMPSPACE1
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0001.0
      - container type: path
      - new container name: D:\DB2\SQLT0001.0

    Redefine inaccessible container:
      - table space name: USERSPACE1
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0002.0
      - container type: path
      - new container name: D:\DB2\SQLT0002.0

  Restore finished.

  Drop the 'RRDB' database.

****************************
*** ROLLFORWARD RECOVERY ***
****************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
  sqluroll -- Rollforward Database
  sqledrpd -- Drop Database
TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD.

  Update 'sample' database configuration:
    - Enable the configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162223

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162223
    - target database        : RFDB

  Restore finished.

  Rolling 'RFDB' database forward ...
  Rollforward finished.
```

```
  Drop the 'RFDB' database.

*****************************
*** ASYNCHRONOUS READ LOG ***
*****************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlurlog -- Asynchronous Read Log
AND THE SQL STATEMENTS:
  CONNECT  ALTER TABLE
  COMMIT  INSERT  DELETE  ROLLBACK
  CONNECT RESET
TO READ LOG RECORDS FOR THE CURRENT CONNECTION.

  Update 'sample' database configuration:
    - Enable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162247

  Verbindung zur Datenbank 'sample' wird hergestellt...
  Verbindung zur Datenbank 'sample' hergestellt.

  Invoke the following SQL statements:
    ALTER TABLE emp_resume DATA CAPTURE CHANGES;
    COMMIT;
    INSERT INTO emp_resume
      VALUES('000777', 'ascii', 'This is a new resume.');
            ('777777', 'ascii', 'This is another new resume');
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000777';
    DELETE FROM emp_resume WHERE empno = '777777';
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000140';
    ROLLBACK;
    ALTER TABLE emp_resume DATA CAPTURE NONE;
    COMMIT;

  Start reading database log.

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database log info -- get
line               = 1457
file               = dbrecov.sqc
SQLCODE            = 2653

SQL2653W  Bei der Wiederherstellung, der aktualisierenden Wiederherstellung
```

```
oder der Wiederherstellung nach einem Systemabsturz wurden
Protokollfolgenummern möglicherweise neu verwendet. Ursachencode: "%1".

---- end warning report ----------------------

    Record type: Normal
      component ID: DMS log record
      function ID:  Alter Table Attribute
        Propagation attribute is changed to: ON

    Record type: Normal
      component ID: DMS log record
      function ID: Update Record
        oldRID: 2
        old subrecord length: 76
        old subrecord offset: 0
        subrecord type: Updatable, Internal control
        newRID: 2
        new subrecord length: 76
        new subrecord offset: 2916
        subrecord type: Updatable, Internal control

    Record type: Local pending list
      UTC transaction committed (in seconds since 01/01/70): 989180591
      authorization ID of the application: MELNYK

    Record type: Normal
      component ID: DMS log record
      function ID: Insert Record
        RID: 12
        subrecord length: 88
        subrecord offset: 486
        subrecord type: Updatable, Formatted user data
        user data fixed length: 15
        user data:
        30 30 30 37 37 37 0F 00 05 00 *000777....*
        14 00 3C 00 00 61 73 63 69 69 *.....ascii*
        49 06 01 00 00 00 00 00 15 00 *I.........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 3C 00 01 00 00 00 15 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*

    Record type: Normal
      component ID: DMS log record
      function ID: Insert Record
        RID: 13
        subrecord length: 88
        subrecord offset: 398
        subrecord type: Updatable, Formatted user data
        user data fixed length: 15
        user data:
        37 37 37 37 37 37 0F 00 05 00 *777777....*
        14 00 3C 00 00 61 73 63 69 69 *.....ascii*
```

```
    49 06 01 00 00 00 00 00 1A 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 1A 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 01 00 00 00 *..........*

Record type: Normal commit
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 12
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 37 37 37 0F 00 05 00 *000777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 15 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 15 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 13
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    37 37 37 37 37 37 0F 00 05 00 *777777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 1A 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 1A 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 01 00 00 00 *..........*

Record type: Normal commit
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 6
```

```
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 05 00 *000140....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 24 05 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 01 3C 00 02 00 00 00 24 05 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 14 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 7
    subrecord length: 89
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 06 00 *000140....*
    15 00 3C 00 00 73 63 72 69 70 *.....scrip*
    74 49 06 01 00 00 00 00 00 56 *tI.......V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 01 3C 00 02 00 00 00 56 *.........V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 16 00 00 *..........*
    00                            *.         *

Record type: Compensation
  component ID: DMS log record
  function ID: Undo Delete Record
    RID: 7
    subrecord length: 89
    subrecord offset: 397
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 06 00 *000140....*
    15 00 3C 00 00 73 63 72 69 70 *.....scrip*
    74 49 06 01 00 00 00 00 00 56 *tI.......V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 01 3C 00 02 00 00 00 56 *.........V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 16 00 00 *..........*
    00                            *.         *

Record type: Compensation
  component ID: DMS log record
  function ID: Undo Delete Record
```

```
            RID: 6
            subrecord length: 88
            subrecord offset: 309
            subrecord type: Updatable, Formatted user data
            user data fixed length: 15
            user data:
            30 30 30 31 34 30 0F 00 05 00 *000140....*
            14 00 3C 00 00 61 73 63 69 69 *.....ascii*
            49 06 01 00 00 00 00 00 24 05 *I.........*
            00 00 00 00 00 00 00 00 00 00 *..........*
            00 01 3C 00 02 00 00 00 24 05 *..........*
            00 00 00 00 00 00 00 00 00 00 *..........*
            00 00 00 00 00 00 00 00 00 00 *..........*
            00 00 00 00 00 00 14 00 00 00 *..........*

          Record type: Normal abort
            authorization ID of the application: MELNYK

          Record type: Normal
            component ID: DMS log record
            function ID:  Alter Table Attribute
              Propagation attribute is changed to: OFF

          Record type: Local pending list
            UTC transaction committed (in seconds since 01/01/70): 989180591
            authorization ID of the application: MELNYK

        Verbindung zur Datenbank 'sample' wird abgebrochen...
        Verbindung zur Datenbank 'sample' abgebrochen.

******************************************
*** READ A DATABASE RECOVERY HISTORY FILE ***
******************************************

USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryCloseScan -- Close Recovery History File Scan
TO READ A DATABASE RECOVERY HISTORY FILE.

  Open recovery history file for 'sample' database.

  Read entry number 0.

  Display entry number 0.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
```

```
  history file entry ID: 48
  table spaces:
    SYSCATSPACE      USERSPACE1    type of operation: B
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:

Read entry number 1.

Display entry number 1.
  object part: 20010506162058001
  end time: 200105061621
  first log: S0000000
  last log: S0000000
  ID: 20010506162032
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 RESTORE SAMPLE NO RF
  command text:
  history file entry ID: 49
  table spaces:
    SYSCATSPACE      USERSPACE1    type of operation: R
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:

Read entry number 2.

Display entry number 2.
  object part: 20010506162125001
  end time: 200105061622
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 BACKUP SAMPLE OFFLINE
  command text:
  history file entry ID: 50
  table spaces:
    SYSCATSPACE      USERSPACE1    type of operation: B
  granularity of the operation: D
  operation type: F
```

```
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:

Read entry number 3.

Display entry number 3.
  object part: 20010506162223001
  end time: 200105061622
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 BACKUP SAMPLE OFFLINE
  command text:
  history file entry ID: 51
  table spaces:
    SYSCATSPACE      USERSPACE1    type of operation: B
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:

Read entry number 4.

Display entry number 4.
  object part: 20010506162247001
  end time: 200105061623
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 BACKUP SAMPLE OFFLINE
  command text:
  history file entry ID: 52
  table spaces:
    SYSCATSPACE      USERSPACE1    type of operation: B
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
```

```
      message:

  Close recovery history file for 'sample' database.

**************************************************
*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***
**************************************************

USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryUpdate -- Update Recovery History File
  db2HistoryCloseScan -- Close Recovery History File Scan
TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.

  Open the recovery history file for 'sample' database.

  Read the first entry in the recovery history file.

  Display the first entry.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 48
    table spaces:
      SYSCATSPACE      USERSPACE1    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Verbindung zur Datenbank 'sample' wird hergestellt...
  Verbindung zur Datenbank 'sample' hergestellt.

  Update the first entry in the history file:
    new location = 'this is the NEW LOCATION'
    new comment = 'this is the NEW COMMENT'

  Verbindung zur Datenbank 'sample' wird abgebrochen...
  Verbindung zur Datenbank 'sample' abgebrochen.

  Close recovery history file for 'sample' database.

  Read the first recovery history file entry.
```

```
  Display the first entry.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: this is the NEW LOCATION
    comment: this is the NEW COMMENT
    command text:
    history file entry ID: 48
    table spaces:
      SYSCATSPACE      USERSPACE1    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Close the recovery history file for 'sample' database.

**************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
**************************************

USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
  CONNECT  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.

  Verbindung zur Datenbank 'sample' wird hergestellt...
  Verbindung zur Datenbank 'sample' hergestellt.

  Prune the recovery history file for 'sample' database.

  Verbindung zur Datenbank 'sample' wird abgebrochen...
  Verbindung zur Datenbank 'sample' abgebrochen.
```

Im Folgenden sehen Sie das Listing des Quellcodes für das Beispielprogramm:

```
/****************************************************************************
**
** Source File Name: dbrecov.sqc
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 2001
** All Rights Reserved.
**
```

```
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**
**    PURPOSE :
**    This sample shows how to recover a database.
**
**    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
**        db2Prune -- Prune Recovery History File
**        sqlbmtsq -- Tablespace Query
**        sqlbstsc -- Set Tablespace Containers
**        sqlbtcq -- Tablespace Container Query
**        sqledrpd -- Drop Database
**        sqlefmem -- Free Memory
**        sqlfudb -- Update Database Configuration
**        sqlfxdb -- Get Database Configuration
**        sqlubkp -- Backup Database
**        sqlurestore -- Restore Database
**        sqlurlog -- Asynchronous Read Log
**        sqluroll -- Rollforward Database
**
** For detailed information about database backup and 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 about the sample programs, see the README file.
**
** For more information about programming in C, see the
** "Programming in C and C++" section of the "Application Development Guide".
**
** For more information about building C applications, see the
** section for your compiler in the "Building Applications" chapter
** for your platform in the "Application Building Guide".
**
** For more information about SQL, see the "SQL Reference".
**
** For more information about DB2 APIs, see the "Administrative API Reference".
**
** For the latest information on programming, compiling, and running DB2
** applications, visit the DB2 application development Web site at
**     http://www.software.ibm.com/data/db2/udb/ad
**
******************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlutil.h>
#include <db2ApiDf.h>
#include "utilemb.h"
```

```
int DbRecoveryHistoryFilePrune(char *, char *, char *);
int DbBackupAndRestore(char *, char *, char *, char *, char *);
int DbBackupAndRedirectedRestore(char *, char *, char *, char *, char *);
int DbBackupRestoreAndRollforward(char *, char *, char *, char *, char *);
int DbLogRecordsForCurrentConnectionRead(char *, char *, char *, char *);
int DbRecoveryHistoryFileRead(char *);
int DbFirstRecoveryHistoryFileEntryUpdate(char *, char *, char *);


/* support function called by the main() */
int ServerWorkingPathGet(char *, char *);

/* support function called by DbBackupAndRedirectedRestore() */
int InaccessableContainersRedefine(char *);

/* support function called by DbBackupAndRedirectedRestore() and
   DbBackupRestoreAndRollforward() */
int DbDrop(char *);

/* support function called by DbLogRecordsForCurrentConnectionRead() */
int LogBufferDisplay(char *, sqluint32);
int LogRecordDisplay(char *, sqluint32, sqluint16, sqluint16);
int SimpleLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32);
int ComplexLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32,
                            sqluint8, char *, sqluint32);
int LogSubRecordDisplay(char *, sqluint16);
int UserDataDisplay(char *, sqluint16);

/* support functions called by DbRecoveryHistoryFileRead() and
   DbFirstRecoveryHistoryFileEntryUpdate() */
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *);
int HistoryEntryDisplay(struct db2HistoryData);
int HistoryEntryDataFieldsFree(struct db2HistoryData *);

/* DbCreate will create a new database on the server with the server's
    code page.
   Use this function only if you want to restore a remote database.
   This support function is being called by DbBackupAndRedirectedRestore()
   and DbBackupRestoreAndRollforward().  */
int DbCreate(char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1];
  char serverWorkingPath[SQL_PATH_SZ + 1];
  char restoredDbAlias[SQL_ALIAS_SZ + 1];
  char redirectedRestoredDbAlias[SQL_ALIAS_SZ + 1];
  char rolledForwardDbAlias[SQL_ALIAS_SZ + 1];
  sqluint16 savedLogRetainValue;
  char dbAlias[SQL_ALIAS_SZ + 1];
  char user[USERID_SZ + 1];
  char pswd[PSWD_SZ + 1];
```

```
/* check the command line arguments */
rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
if (rc != 0)
{
  return rc;
}

printf("\nTHIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.\n");

strcpy(restoredDbAlias, dbAlias);
strcpy(redirectedRestoredDbAlias, "RRDB");
strcpy(rolledForwardDbAlias, "RFDB");

/* attach to a local or remote instance */
rc = InstanceAttach(nodeName, user, pswd);
if (rc != 0)
{
  return rc;
}

printf("\nUSE THE DB2 API:\n");
printf("  sqlfxdb -- Get Database Configuration\n");
printf("TO GET THE DATABASE CONFIGURATION AND DETERMINE\n");
printf("THE SERVER WORKING PATH.\n");

/* get the server working path */
rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
if (rc != 0)
{
  return rc;
}

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 functions */
rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);

rc = DbBackupAndRestore(dbAlias,
                        restoredDbAlias,
                        user,
                        pswd,
                        serverWorkingPath);

rc = DbBackupAndRedirectedRestore(dbAlias,
                                  redirectedRestoredDbAlias,
                                  user,
                                  pswd,
                                  serverWorkingPath);

rc = DbBackupRestoreAndRollforward(dbAlias,
                                   rolledForwardDbAlias,
                                   user,
                                   pswd,
```

```
                                        serverWorkingPath);

  rc = DbLogRecordsForCurrentConnectionRead(dbAlias,
                                            user,
                                            pswd,
                                            serverWorkingPath);

  rc = DbRecoveryHistoryFileRead(dbAlias);

  rc = DbFirstRecoveryHistoryFileEntryUpdate(dbAlias, user, pswd);

  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);

  /* detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* end main */

int ServerWorkingPathGet(char dbAlias[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbConfigFields[1];
  char serverLogPath[SQL_PATH_SZ + 1];
  int len;

  /* get the server log path */
  /* SQLF_DBTN_LOGPATH is a token of the non-updatable database configuration
     parameter 'logpath'; it is used to get the server log path */
  dbConfigFields[0].token = SQLF_DBTN_LOGPATH;
  dbConfigFields[0].ptrvalue =
    (char *)malloc((SQL_PATH_SZ + 1) * sizeof(char));

  /* get database configuration */
  /* the API sqlfxdb returns the values of individual entries in a
     database configuration file */
  sqlfxdb(dbAlias, 1, &dbConfigFields[0], &sqlca);
  DB2_API_CHECK("server log path -- get");

  strcpy(serverLogPath, dbConfigFields[0].ptrvalue);
  free(dbConfigFields[0].ptrvalue);

  /* choose the server working path; if, for example, serverLogPath =
     "C:\DB2\NODE0001\....", we'll keep "C:\DB2" for the serverWorkingPath
     variable; backup images created in this sample will be placed under
     the 'serverWorkingPath' directory */
  len = (int)(strstr(serverLogPath, "NODE") - serverLogPath - 1);
  memcpy(serverWorkingPath, serverLogPath, len);
  serverWorkingPath[len] = '\0';
```

```
  return 0;
} /* ServerWorkingPathGet */

int DbCreate(char existingDbAlias[], char newDbAlias[])
{
  struct sqlca sqlca;
  char dbName[SQL_DBNAME_SZ + 1];
  char dbLocalAlias[SQL_ALIAS_SZ + 1];
  char dbPath[SQL_PATH_SZ + 1];
  struct sqledbdesc dbDescriptor;
  struct sqledbcountryinfo countryInfo;
  struct sqlfupd dbConfigFields[2];

  printf("\n  Create '%s' empty database with the same code set as
   '%s' database.\n",
         newDbAlias, existingDbAlias);

  /* initialize dbConfigFields */
  dbConfigFields[0].token = SQLF_DBTN_TERRITORY;
  dbConfigFields[0].ptrvalue = (char *)malloc(10 * sizeof(char));
  memset(dbConfigFields[0].ptrvalue, '\0', 10);
  dbConfigFields[1].token = SQLF_DBTN_CODESET;
  dbConfigFields[1].ptrvalue = (char *)malloc(20 * sizeof(char));
  memset(dbConfigFields[1].ptrvalue, '\0', 20);

  /* get two database configuration fields */
  sqlfxdb(existingDbAlias, 2, &dbConfigFields[0], &sqlca);
  DB2_API_CHECK("DB Config. -- Get");

  /* create a new database */
  strcpy(dbName, newDbAlias);
  strcpy(dbLocalAlias, newDbAlias);
  strcpy(dbPath, "");

  strcpy(dbDescriptor.sqldbdid, SQLE_DBDESC_2);
  dbDescriptor.sqldbccp = 0;
  dbDescriptor.sqldbcss = SQL_CS_NONE;

  strcpy(dbDescriptor.sqldbcmt, "");
  dbDescriptor.sqldbsgp = 0;
  dbDescriptor.sqldbnsg = 10;
  dbDescriptor.sqltsext = -1;
  dbDescriptor.sqlcatts = NULL;
  dbDescriptor.sqlusrts = NULL;
  dbDescriptor.sqltmpts = NULL;

  strcpy(countryInfo.sqldbcodeset, (char *)dbConfigFields[1].ptrvalue);
  strcpy(countryInfo.sqldblocale, (char *)dbConfigFields[0].ptrvalue);

  /* create database */
  sqlecrea(dbName,
           dbLocalAlias,
           dbPath,
           &dbDescriptor,
           &countryInfo,
```

```
            '\0',
            NULL,
            &sqlca);
  DB2_API_CHECK("Database -- Create");

  /* free the allocated memory */
  free(dbConfigFields[0].ptrvalue);
  free(dbConfigFields[1].ptrvalue);

  return 0;
} /* DbCreate */

int DbDrop(char dbAlias[])
{
  struct sqlca sqlca;

  printf("\n  Drop the '%s' database.\n", dbAlias);

  /* drop and uncatalog the database */
  sqledrpd(dbAlias, &sqlca);
  DB2_API_CHECK("Database -- Drop");

  return 0;
} /* DbDrop */

int DbBackupAndRestore(char dbAlias[],
                       char restoredDbAlias[],
                       char user[],
                       char pswd[],
                       char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
```

```
char *restoreTargetPath;
sqluint32 restoreBuffersNb;
struct sqlu_tablespace_bkrst_list restoreTablespaceList;
struct sqlu_media_list restoreSourceMediaList;
struct sqlu_media_entry restoreSourceMediaEntries[1];
sqluint32 restoreParallelismDegree;

printf("\n************************************\n");
printf("*** BACK UP AND RESTORE A DATABASE ***\n");
printf("************************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlurestore -- 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");

/* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
   parameter 'logretain'; it is used to update the database configuration
   file */
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* disable the database configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_DISABLE;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");

/*******************************/
/*    BACK UP THE DATABASE     */
/*******************************/
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
```

```
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */

  /* depending on the sqlca.sqlcode value, user action may be */
  /* required, such as mounting a new tape */

  printf("\n  Continuing the backup operation...\n");

  backupCallerAction = SQLUB_CONTINUE;

  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
          NULL,
          0,
          NULL,
          backupParallelismDegree,
          &backupImageSize,
          NULL,
          NULL,
```

```
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}

printf("  Backup finished.\n");
printf("    - backup image size      : %d MB\n", backupImageSize);
printf("    - backup image path      : %s\n",
       backupTargetMediaEntries[0].media_entry);
printf("    - backup image time stamp: %s\n",
       backupStartTimestamp);

/*****************************/
/*    RESTORE THE DATABASE    */
/*****************************/

restoreBufferSize = 1024; /*  1024 x 4KB */
rollforwardPendingState = SQLUD_NOROLLFWD;
datalinkMode = SQLUD_NODATALINK;
restoreType = SQLUD_FULL;
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* number of table spaces */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* number of elements in the target */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;

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);

restoreCallerAction = SQLUD_RESTORE;

/* The API sqlurestore is used to restore a database that has been backed
   up using the API sqlubkp. */
sqlurestore(dbAlias,
            restoredDbAlias,
            restoreBufferSize,
            rollforwardPendingState,
            datalinkMode,
            restoreType,
            restoreMode,
            restoreCallerAction,
            restoreAppId,
            restoreTimestamp,
            restoreTargetPath,
            restoreBuffersNb,
            NULL,
            &restoreTablespaceList,
            &restoreSourceMediaList,
```

```
                user,
                pswd,
                NULL,
                0,
                NULL,
                restoreParallelismDegree,
                NULL,
                NULL,
                NULL,
                &sqlca);
/*  DB2_API_CHECK("database restore -- start");  */
  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 */

    restoreCallerAction = SQLUD_CONTINUE;

    /* restore the database */
    sqlurestore(dbAlias,
                restoredDbAlias,
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
                NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
                NULL,
                0,
                NULL,
                restoreParallelismDegree,
                NULL,
                NULL,
                NULL,
                &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }

  printf("\n  Restore finished.\n");

  return 0;
```

```
} /* DbBackupAndRestore */

int DbBackupAndRedirectedRestore(char dbAlias[],
                                 char restoredDbAlias[],
                                 char user[],
                                 char pswd[],
                                 char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
  char *restoreTargetPath;
  sqluint32 restoreBuffersNb;
  struct sqlu_tablespace_bkrst_list restoreTablespaceList;
  struct sqlu_media_list restoreSourceMediaList;
  struct sqlu_media_entry restoreSourceMediaEntries[1];
  sqluint32 restoreParallelismDegree;

  printf("\n*************************\n");
  printf("*** REDIRECTED RESTORE ***\n");
  printf("*************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  sqlfudb -- Update Database Configuration\n");
  printf("  sqlubkp -- Backup Database\n");
  printf("  sqlecrea -- Create Database\n");
  printf("  sqlurestore -- 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");

/* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
   parameter 'logretain'; it is used to update the database configuration
   file */
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* disable the database configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_DISABLE;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");

/*******************************/
/*    BACK UP THE DATABASE    */
/*******************************/
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
```

```
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */

  /* depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape */

  printf("\n  Continuing the backup operation...\n");

  backupCallerAction = SQLUB_CONTINUE;

  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
          NULL,
          0,
          NULL,
          backupParallelismDegree,
          &backupImageSize,
          NULL,
          NULL,
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}

printf("  Backup finished.\n");
printf("    - backup image size      : %d MB\n", backupImageSize);
printf("    - backup image path      : %s\n",
       backupTargetMediaEntries[0].media_entry);
printf("    - backup image time stamp: %s\n",
       backupStartTimestamp);

/* 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, restoredDbAlias);
  if (rc != 0)
  {
  return rc;
  }
*/

  /*****************************/
  /*    RESTORE THE DATABASE    */
  /*****************************/

  restoreBufferSize = 1024; /*  1024 x 4KB */
  rollforwardPendingState = SQLUD_NOROLLFWD;
  datalinkMode = SQLUD_NODATALINK;
  restoreType = SQLUD_FULL;
  restoreMode = SQLUD_OFFLINE;
  strcpy(restoreTimestamp, backupStartTimestamp);
  restoreTargetPath = NULL;
  restoreBuffersNb = 1;
  restoreTablespaceList.num_entry = 0; /* number of table spaces */
  restoreTablespaceList.tablespace = NULL;
  restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
  restoreSourceMediaList.sessions = 1; /* number of elements in the target */
  restoreSourceMediaList.target.media = restoreSourceMediaEntries;
  strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
  restoreParallelismDegree = 1;

  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);

  restoreCallerAction = SQLUD_RESTORE_STORDEF;

  /* The API sqlurestore is used to restore a database that has been backed
     up using the API sqlubkp. */
  sqlurestore(dbAlias,
              restoredDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
              restoreAppId,
              restoreTimestamp,
              restoreTargetPath,
              restoreBuffersNb,
              NULL,
              &restoreTablespaceList,
              &restoreSourceMediaList,
              user,
              pswd,
              NULL,
```

```
                0,
                NULL,
                restoreParallelismDegree,
                NULL,
                NULL,
                NULL,
                &sqlca);
/*  DB2_API_CHECK("database restore -- start"); */
  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);
      if (rc != 0)
      {
        return rc;
      }
    }

    restoreCallerAction = SQLUD_CONTINUE;

    /* restore the database */
    sqlurestore(dbAlias,
                restoredDbAlias,
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
                NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
                NULL,
                0,
                NULL,
                restoreParallelismDegree,
                NULL,
                NULL,
```

```
                NULL,
                &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }

  printf("\n  Restore finished.\n");

  /* drop the restored database */
  rc = DbDrop(restoredDbAlias);

  return 0;
} /* DbBackupAndRedirectedRestore */

int InaccessableContainersRedefine(char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  sqluint32 numTablespaces;
  struct SQLB_TBSPQRY_DATA **ppTablespaces;
  sqluint32 numContainers;
  struct SQLB_TBSCONTQRY_DATA *pContainers;
  int tspNb;
  int contNb;
  char pathSep[2];

  /* 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 inaccessible 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 inaccessible container */
        printf("\n    Redefine inaccessible 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;
} /* InaccessibleContainersRedefine */

int DbBackupRestoreAndRollforward(char dbAlias[],
                                  char rolledForwardDbAlias[],
                                  char user[],
```

```
                                    char pswd[],
                                    char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
  char *restoreTargetPath;
  sqluint32 restoreBuffersNb;
  struct sqlu_tablespace_bkrst_list restoreTablespaceList;
  struct sqlu_media_list restoreSourceMediaList;
  struct sqlu_media_entry restoreSourceMediaEntries[1];
  sqluint32 restoreParallelismDegree;
  struct rfwd_input rfwdInput;
  struct rfwd_output rfwdOutput;
  char rollforwardAppId[SQLU_APPLID_LEN + 1];
  sqlint32 numReplies;
  struct sqlurf_info nodeInfo;

  printf("\n****************************\n");
  printf("*** ROLLFORWARD RECOVERY ***\n");
  printf("****************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  sqlfudb -- Update Database Configuration\n");
  printf("  sqlubkp -- Backup Database\n");
  printf("  sqlecrea -- Create Database\n");
  printf("  sqlurestore -- Restore Database\n");
  printf("  sqluroll -- Rollforward Database\n");
  printf("  sqledrpd -- Drop Database\n");
  printf("TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD. \n");

  printf("\n  Update \'%s\' database configuration:\n", dbAlias);
  printf("    - Enable the configuration parameter LOGRETAIN \n");
  printf("        i.e., set LOGRETAIN = RECOVERY/YES\n");
```

```
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* enable the configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/* start the backup operation */
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");
```

```
  while (sqlca.sqlcode != 0)
  {
    /* continue the backup operation */
    printf("\n  Continuing the backup operation...\n");

    /* depending on the sqlca.sqlcode value, user action may be
       required, such as mounting a new tape. */

    backupCallerAction = SQLUB_CONTINUE;

    /* back up the database */
    sqlubkp(dbAlias,
            backupBufferSize,
            backupMode,
            backupType,
            backupCallerAction,
            backupAppId,
            backupStartTimestamp,
            backupBuffersNb,
            &backupTablespaceList,
            &backupTargetMediaList,
            user,
            pswd,
            NULL,
            0,
            NULL,
            backupParallelismDegree,
            &backupImageSize,
            NULL,
            NULL,
            &sqlca);
    DB2_API_CHECK("Database -- Backup");
  }

  printf("  Backup finished.\n");
  printf("    - backup image size      : %d MB\n", backupImageSize);
  printf("    - backup image path      : %s\n",
         backupTargetMediaEntries[0].media_entry);
  printf("    - backup image time stamp: %s\n",
         backupStartTimestamp);

  /* 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);
  if (rc != 0)
  {
    return rc;
  }
*/
```

```
/*****************************/
/*    RESTORE THE DATABASE    */
/*****************************/

restoreBufferSize = 1024; /*  1024 x 4KB */
rollforwardPendingState = SQLUD_ROLLFWD;
datalinkMode = SQLUD_NODATALINK;
restoreType = SQLUD_FULL;
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* number of table spaces */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* number of elements in the target */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;

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);

restoreCallerAction = SQLUD_RESTORE;

/* The API sqlurestore is used to restore a database that has been backed
   up using the API sqlubkp. */
sqlurestore(dbAlias,
            rolledForwardDbAlias,
            restoreBufferSize,
            rollforwardPendingState,
            datalinkMode,
            restoreType,
            restoreMode,
            restoreCallerAction,
            restoreAppId,
            restoreTimestamp,
            restoreTargetPath,
            restoreBuffersNb,
            NULL,
            &restoreTablespaceList,
            &restoreSourceMediaList,
            user,
            pswd,
            NULL,
            0,
            NULL,
            restoreParallelismDegree,
            NULL,
            NULL,
            NULL,
            &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. */

  restoreCallerAction = SQLUD_CONTINUE;

  /* restore the database */
  sqlurestore(dbAlias,
              rolledForwardDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
              restoreAppId,
              restoreTimestamp,
              restoreTargetPath,
              restoreBuffersNb,
              NULL,
              &restoreTablespaceList,
              &restoreSourceMediaList,
              user,
              pswd,
              NULL,
              0,
              NULL,
              restoreParallelismDegree,
              NULL,
              NULL,
              NULL,
              &sqlca);
  DB2_API_CHECK("database restore -- continue");
}

printf("\n  Restore finished.\n");


/*****************************/
/*    ROLLFORWARD RECOVERY    */
/*****************************/

printf("\n  Rolling '%s' database forward ...\n", rolledForwardDbAlias);

rfwdInput.version = SQLUM_RFWD_VERSION;
rfwdInput.pDbAlias = rolledForwardDbAlias;
rfwdInput.CallerAction = SQLUM_ROLLFWD_STOP;
rfwdInput.pStopTime = SQLUM_INFINITY_TIMESTAMP;
rfwdInput.pUserName = user;
rfwdInput.pPassword = pswd;
```

```
  rfwdInput.pOverflowLogPath = serverWorkingPath;
  rfwdInput.NumChngLgOvrflw = 0;
  rfwdInput.pChngLogOvrflw = NULL;
  rfwdInput.ConnectMode = SQLUM_OFFLINE;
  rfwdInput.pTablespaceList = NULL;
  rfwdInput.AllNodeFlag = SQLURF_ALL_NODES;
  rfwdInput.NumNodes = 0;
  rfwdInput.pNodeList = NULL;
  rfwdInput.NumNodeInfo = 1;
  rfwdInput.DlMode = 0;
  rfwdInput.pReportFile = NULL;
  rfwdInput.pDroppedTblID = NULL;
  rfwdInput.pExportDir = NULL;

  rfwdOutput.pApplicationId = rollforwardAppId;
  rfwdOutput.pNumReplies = &numReplies;
  rfwdOutput.pNodeInfo = &nodeInfo;

  /* rollforward database */
  /* The API sqluroll rollforward recovers a database by
     applying transactions recorded in the database log files. */
  sqluroll(&rfwdInput, &rfwdOutput, &sqlca);
  DB2_API_CHECK("rollforward -- start");

  printf("  Rollforward finished.\n");

  /* drop the restored database */
  rc = DbDrop(rolledForwardDbAlias);

  return 0;
} /* DbBackupRestoreAndRollforward */

int DbLogRecordsForCurrentConnectionRead(char dbAlias[],
                                         char user[],
                                         char pswd[],
                                         char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  SQLU_LSN startLSN;
  SQLU_LSN endLSN;
```

```
char *logBuffer;
sqluint32 logBufferSize;
SQLU_RLOG_INFO readLogInfo;
int i;

printf("\n*****************************\n");
printf("*** ASYNCHRONOUS READ LOG ***\n");
printf("*****************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlurlog -- 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");

dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* enable LOGRETAIN */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/* start the backup operation */
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
```

```
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */
  printf("\n  Continuing the backup operation...\n");

  /* Depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */

  backupCallerAction = SQLUB_CONTINUE;

  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
          NULL,
          0,
          NULL,
          backupParallelismDegree,
          &backupImageSize,
          NULL,
          NULL,
```

```
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}

printf("  Backup finished.\n");
printf("    - backup image size      : %d MB\n", backupImageSize);
printf("    - backup image path      : %s\n",
       backupTargetMediaEntries[0].media_entry);
printf("    - backup image time stamp: %s\n",
       backupStartTimestamp);

/* connect to the database */
rc = DbConn(dbAlias, user, pswd);
if (rc != 0)
{
  return rc;
}

/* 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 sqlurlog (Asynchronours 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: */
  rc = sqlurlog(SQLU_RLOG_QUERY,
                &startLSN,
                &endLSN,
                logBuffer,
                logBufferSize,
                &readLogInfo,
                &sqlca);
/*  DB2_API_CHECK("database log info -- get"); */
  EXPECTED_WARN_CHECK("database log info -- get");

  logBufferSize = 64 * 1024;
  logBuffer = (char *)malloc(logBufferSize);

  memcpy(&startLSN, &(readLogInfo.initialLSN), sizeof(startLSN));
  memcpy(&endLSN, &(readLogInfo.curActiveLSN), sizeof(endLSN));

  /* Extract a log record from the database logs, and
     read the first log sequence asynchronously: */
  rc = sqlurlog(SQLU_RLOG_READ,
                &startLSN,
                &endLSN,
                logBuffer,
                logBufferSize,
                &readLogInfo,
                &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);

  while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    /* read the next log sequence */

    memcpy(&startLSN, &(readLogInfo.lastReadLSN), sizeof(startLSN));
    /* increase startLSN by 1 */
    startLSN.lsnChar[5] = startLSN.lsnChar[5] + 1;
    i = 5;
    while (startLSN.lsnChar[i] == 0 && i > 0)
    {
      startLSN.lsnChar[i - 1] = startLSN.lsnChar[i - 1] + 1;
      i = i - 1;
    }

    /* Extract a log record from the database logs, and
     read the next log sequence asynchronously: */
    rc = sqlurlog(SQLU_RLOG_READ,
                  &startLSN,
                  &endLSN,
                  logBuffer,
                  logBufferSize,
                  &readLogInfo,
                  &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
    {
      DB2_API_CHECK("database logs -- read");
    }

    /* display log buffer */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
  }

  /* free the log buffer */
  free(logBuffer);

  /* disconnect from the database */
  rc = DbDisconn(dbAlias);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* DbLogRecordsForCurrentConnectionRead */

int LogBufferDisplay(char *logBuffer, sqluint32 numLogRecords)
```

```
{
  int rc = 0;
  sqluint32 logRecordNb;
  sqluint32 recordSize;
  sqluint16 recordType;
  sqluint16 recordFlag;
  char *recordBuffer;

  /* initialize recordBuffer */
  recordBuffer = logBuffer + sizeof(SQLU_LSN);

  for (logRecordNb = 0; logRecordNb < numLogRecords; logRecordNb++)
  {
    recordSize = *(sqluint32 *)(recordBuffer);
    recordType = *(sqluint16 *)(recordBuffer + 4);
    recordFlag = *(sqluint16 *)(recordBuffer + 6);

    rc = LogRecordDisplay(recordBuffer, recordSize, recordType, recordFlag);
    /* update recordBuffer */
    recordBuffer = recordBuffer + recordSize + sizeof(SQLU_LSN);
  }

  return 0;
} /* LogBufferDisplay */

int LogRecordDisplay(char *recordBuffer,
                     sqluint32 recordSize,
                     sqluint16 recordType,
                     sqluint16 recordFlag)
{
  int rc = 0;
  sqluint32 logManagerLogRecordHeaderSize;
  char *recordDataBuffer;
  sqluint32 recordDataSize;
  char *recordHeaderBuffer;
  sqluint8 componentIdentifier;
  sqluint32 recordHeaderSize;

  /* determine logManagerLogRecordHeaderSize */
  if ((char)recordType == 'C')
  { /* compensation */
    if (recordFlag & 0x0002)
    { /* propagatable */
      logManagerLogRecordHeaderSize = 32;
    }
    else {
      logManagerLogRecordHeaderSize = 26;
    }
  }
  else
  { /* non compensation */
    logManagerLogRecordHeaderSize = 20;
  }

  switch ((char)recordType)
```

```
  {
    case 'E':
    case 'M':
    case 'A':
      recordDataBuffer = recordBuffer + logManagerLogRecordHeaderSize;
      recordDataSize = recordSize - logManagerLogRecordHeaderSize;
      rc = SimpleLogRecordDisplay(recordType,
                                  recordFlag,
                                  recordDataBuffer,
                                  recordDataSize);
      break;
    case 'N':
    case 'C':
      recordHeaderBuffer = recordBuffer + logManagerLogRecordHeaderSize;
      componentIdentifier = *(sqluint8 *)recordHeaderBuffer;
      switch (componentIdentifier)
      {
        case 1:
          recordHeaderSize = 6;
          break;
        default:
          printf("    Unknown complex log record: %lu %c %u\n",
                 recordSize, recordType, componentIdentifier);
          return 1;
      }
      recordDataBuffer = recordBuffer +
                         logManagerLogRecordHeaderSize +
                         recordHeaderSize;
      recordDataSize = recordSize -
                       logManagerLogRecordHeaderSize -
                       recordHeaderSize;
      rc = ComplexLogRecordDisplay(recordType,
                                   recordFlag,
                                   recordHeaderBuffer,
                                   recordHeaderSize,
                                   componentIdentifier,
                                   recordDataBuffer,
                                   recordDataSize);
      break;
    default:
      printf("    Unknown log record: %lu %c\n",
             recordSize, (char)recordType);
      break;
  }

  return 0;
} /* LogRecordDisplay */

int SimpleLogRecordDisplay(sqluint16 recordType,
                           sqluint16 recordFlag,
                           char *recordDataBuffer,
                           sqluint32 recordDataSize)
{
  int rc = 0;
  sqluint32 timeTransactionCommited;
```

```
  sqluint16 authIdLen;
  char authId[129];

  switch ((char)recordType)
  {
    case 'E':
      printf("\n    Record type: Local pending list\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = *(sqluint16 *)(recordDataBuffer + 4);
      memcpy(authId, (char *)(recordDataBuffer + 6), authIdLen);
      authId[authIdLen] = '\0';
      printf("      %s: %lu\n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s\n", authId);
      break;
    case 'M':
      printf("\n    Record type: Normal commit\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = (sqluint16) (recordDataSize - 4);
      memcpy(authId, (char *)(recordDataBuffer + 4), authIdLen);
      authId[authIdLen] = '\0';
      printf("      %s: %lu\n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s\n", authId);
      break;
    case 'A':
      printf("\n    Record type: Normal abort\n");
      authIdLen = (sqluint16) (recordDataSize);
      memcpy(authId, (char *)(recordDataBuffer), authIdLen);
      authId[authIdLen] = '\0';
      printf("      authorization ID of the application: %s\n", authId);
      break;
    default:
      printf("    Unknown simple log record: %c %lu\n",
             (char)recordType, recordDataSize);
      break;
  }

  return 0;
} /* SimpleLogRecordDisplay */

int ComplexLogRecordDisplay(sqluint16 recordType,
                            sqluint16 recordFlag,
                            char *recordHeaderBuffer,
                            sqluint32 recordHeaderSize,
                            sqluint8 componentIdentifier,
                            char *recordDataBuffer,
                            sqluint32 recordDataSize)
{
  int rc = 0;
  sqluint8 functionIdentifier;
  /* for insert, delete, undo delete */
  sqluint32 RID;
```

```
sqluint16 subRecordLen;
sqluint16 subRecordOffset;
char *subRecordBuffer;
/* for update */
sqluint32 newRID;
sqluint16 newSubRecordLen;
sqluint16 newSubRecordOffset;
char *newSubRecordBuffer;
sqluint32 oldRID;
sqluint16 oldSubRecordLen;
sqluint16 oldSubRecordOffset;
char *oldSubRecordBuffer;
/* for alter table attributes */
sqluint32 alterBitMask;
sqluint32 alterBitValues;

switch ((char)recordType)
{
  case 'N':
    printf("\n    Record type: Normal\n");
    break;
  case 'C':
    printf("\n    Record type: Compensation\n");
    break;
  default:
    printf("\n    Unknown complex log record type: %c\n", recordType);
    break;
}

switch (componentIdentifier)
{
  case 1:
    printf("      component ID: DMS log record\n");
    break;
  default:
    printf("      unknown component ID: %d\n", componentIdentifier);
    break;
}

functionIdentifier = *(sqluint8 *)(recordHeaderBuffer + 1);
switch (functionIdentifier)
{
  case 106:
    printf("      function ID: Delete Record\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        subrecord length: %u\n", subRecordLen);
    printf("        subrecord offset: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 111:
    printf("      function ID: Undo Delete Record\n");
```

```
  RID = *(sqluint32 *)(recordDataBuffer + 2);
  subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
  subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
  printf("        RID: %lu\n", RID);
  printf("        subrecord length: %u\n", subRecordLen);
  printf("        subrecord offset: %u\n", subRecordOffset);
  subRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
  break;
case 118:
  printf("      function ID: Insert Record\n");
  RID = *(sqluint32 *)(recordDataBuffer + 2);
  subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
  subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
  printf("        RID: %lu\n", RID);
  printf("        subrecord length: %u\n", subRecordLen);
  printf("        subrecord offset: %u\n", subRecordOffset);
  subRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
  break;
case 120:
  printf("      function ID: Update Record\n");
  oldRID = *(sqluint32 *)(recordDataBuffer + 2);
  oldSubRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
  oldSubRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
  newRID = *(sqluint32 *)(recordDataBuffer +
                          12 + oldSubRecordLen + recordHeaderSize + 2);
  newSubRecordLen = *(sqluint16 *)(recordDataBuffer +
                                   12 + oldSubRecordLen +
                                   recordHeaderSize + 6);
  newSubRecordOffset =
    *(sqluint16 *)(recordDataBuffer + 12 + oldSubRecordLen +
                   recordHeaderSize + 10);
  printf("        oldRID: %lu\n", oldRID);
  printf("        old subrecord length: %u\n", oldSubRecordLen);
  printf("        old subrecord offset: %u\n",
         oldSubRecordOffset);
  oldSubRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(oldSubRecordBuffer, oldSubRecordLen);
  printf("        newRID: %lu\n", newRID);
  printf("        new subrecord length: %u\n", newSubRecordLen);
  printf("        new subrecord offset: %u\n",
         newSubRecordOffset);
  newSubRecordBuffer = recordDataBuffer +
    12 + oldSubRecordLen + recordHeaderSize + 12;
  rc = LogSubRecordDisplay(newSubRecordBuffer, newSubRecordLen);
  break;
case 124:
  printf("      function ID:  Alter Table Attribute\n");
  alterBitMask = *(sqluint32 *)(recordDataBuffer + 2);
  alterBitValues = *(sqluint32 *)(recordDataBuffer + 6);
  if (alterBitMask & 0x00000001)
  {
    /* Alter the value of the 'propagation' attribute: */
    printf("        Propagation attribute is changed to: ");
```

```
  if (alterBitValues & 0x00000001)
  {
    printf("ON\n");
  }
  else {
    printf("OFF\n");
  }
}
if (alterBitMask & 0x00000002)
{
  /* Alter the value of the 'pending' attribute: */
  printf("        Pending attribute is changed to: ");
  if (alterBitValues & 0x00000002)
  {
    printf("ON\n");
  }
  else {
    printf("OFF\n");
  }
}
if (alterBitMask & 0x00010000)
{
  /* Alter the value of the 'append mode' attribute: */
  printf("        Append Mode attribute is changed to: ");
  if (alterBitValues & 0x00010000)
  {
    printf("ON\n");
  }
  else {
    printf("OFF\n");
  }
}
if (alterBitMask & 0x00200000)
{
  /* Alter the value of the 'LF Propagation' attribute: */
  printf("        LF Propagation attribute is changed to: ");
  if (alterBitValues & 0x00200000)
  {
    printf("ON\n");
  }
  else {
    printf("OFF\n");
  }
}
if (alterBitMask & 0x00400000)
{
  /* Alter the value of the 'LOB Propagation' attribute: */
  printf("        LOB Propagation attribute is changed to: ");
  if (alterBitValues & 0x00400000)
  {
    printf("ON\n");
  }
  else {
    printf("OFF\n");
  }
```

```
      }
      break;
    default:
      printf("      unknown function identifier: %u\n",
             functionIdentifier);
      break;
  }

  return 0;
} /* ComplexLogRecordDisplay */

int LogSubRecordDisplay(char *recordBuffer, sqluint16 recordSize)
{
  int rc = 0;
  sqluint8 recordType;
  sqluint8 updatableRecordType;
  sqluint16 userDataFixedLength;
  char *userDataBuffer;
  sqluint16 userDataSize;

  recordType = *(sqluint8 *)(recordBuffer);
  if (recordType != 0 && recordType != 4)
  {
    printf("        Unknown subrecord type.\n");
  }
  else if (recordType == 4)
  {
    printf("        subrecord type: Special control\n");
  }
  else {
    /* recordType == 0 */
    printf("        subrecord type: Updatable, ");
    updatableRecordType = *(sqluint8 *)(recordBuffer + 4);
    if (updatableRecordType != 1)
    {
      printf("Internal control\n");
    }
    else {
      printf("Formatted user data\n");
      userDataFixedLength = *(sqluint16 *)(recordBuffer + 6);
      printf("        user data fixed length: %u\n",
             userDataFixedLength);
      userDataBuffer = recordBuffer + 8;
      userDataSize = recordSize - 8;
      rc = UserDataDisplay(userDataBuffer, userDataSize);
    }
  }

  return 0;
} /* LogSubRecordDisplay */

int UserDataDisplay(char *dataBuffer, sqluint16 dataSize)
{
  int rc = 0;
```

```
  sqluint16 line, col;

  printf("        user data:\n");

  for (line = 0; line * 10 < dataSize; line = line + 1)
  {
    printf("        ");
    for (col = 0; col < 10; col = col + 1)
    {
      if (line * 10 + col < dataSize)
      {
        printf("%02X ", dataBuffer[line * 10 + col]);
      }
      else {
        printf("   ");
      }
    }
    printf("*");
    for (col = 0; col < 10; col = col + 1)
    {
      if (line * 10 + col < dataSize)
      {
        if (isalpha(dataBuffer[line * 10 + col]) ||
            isdigit(dataBuffer[line * 10 + col]))
        {
          printf("%c", dataBuffer[line * 10 + col]);
        }
        else {
          printf(".");
        }
      }
      else {
        printf(" ");
      }
    }
    printf("*");
    printf("\n");
  }

  return 0;
} /* UserDataDisplay */

int DbRecoveryHistoryFileRead(char dbAlias[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint32 numEntries;
  sqluint16 recoveryHistoryFileHandle;
  sqluint32 entryNb;
  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);
if (rc != 0)
{
  return rc;
}

/*****************************************/
/* 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(db2Version710, &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 = entryNb + 1)
{
  printf("\n  Read entry number %u.\n", entryNb);

  /* get the next entry from the recovery history file */
  db2HistoryGetEntry(db2Version710, &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);
}

/*****************************************/
/* 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(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);

  return 0;
} /* DbRecoveryHistoryFileRead */

int HistoryEntryDataFieldsAlloc(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb;

  strcpy(pHistEntryData->ioHistDataID, "SQLUHINF");

  pHistEntryData->oObjectPart.pioData = malloc(17 + 1);
  pHistEntryData->oObjectPart.iLength = 17 + 1;

  pHistEntryData->oEndTime.pioData = malloc(12 + 1);
  pHistEntryData->oEndTime.iLength = 12 + 1;

  pHistEntryData->oFirstLog.pioData = malloc(8 + 1);
  pHistEntryData->oFirstLog.iLength = 8 + 1;

  pHistEntryData->oLastLog.pioData = malloc(8 + 1);
  pHistEntryData->oLastLog.iLength = 8 + 1;

  pHistEntryData->oID.pioData = malloc(128 + 1);
  pHistEntryData->oID.iLength = 128 + 1;

  pHistEntryData->oTableQualifier.pioData = malloc(128 + 1);
  pHistEntryData->oTableQualifier.iLength = 128 + 1;

  pHistEntryData->oTableName.pioData = malloc(128 + 1);
  pHistEntryData->oTableName.iLength = 128 + 1;

  pHistEntryData->oLocation.pioData = malloc(128 + 1);
  pHistEntryData->oLocation.iLength = 128 + 1;

  pHistEntryData->oComment.pioData = malloc(128 + 1);
  pHistEntryData->oComment.iLength = 128 + 1;

  pHistEntryData->oCommandText.pioData = malloc(128 + 1);
  pHistEntryData->oCommandText.iLength = 128 + 1;

  pHistEntryData->poEventSQLCA =
    (struct sqlca *)malloc(sizeof(struct sqlca));

  pHistEntryData->poTablespace = (db2Char *)malloc(3 * sizeof(db2Char));
  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
```

```
  {
    pHistEntryData->poTablespace[tsNb].pioData = malloc(18 + 1);
    pHistEntryData->poTablespace[tsNb].iLength = 18 + 1;
  }

  pHistEntryData->iNumTablespaces = 3;

  return 0;
} /* HistoryEntryDataFieldsAlloc */

int HistoryEntryDisplay(struct db2HistoryData histEntryData)
{
  int rc = 0;
  char buf[129];
  sqluint32 tsNb;

  memcpy(buf, histEntryData.oObjectPart.pioData,
              histEntryData.oObjectPart.oLength);
  buf[histEntryData.oObjectPart.oLength] = '\0';
  printf("    object part: %s\n", buf);

  memcpy(buf, histEntryData.oEndTime.pioData,
              histEntryData.oEndTime.oLength);
  buf[histEntryData.oEndTime.oLength] = '\0';
  printf("    end time: %s\n", buf);

  memcpy(buf, histEntryData.oFirstLog.pioData,
              histEntryData.oFirstLog.oLength);
  buf[histEntryData.oFirstLog.oLength] = '\0';
  printf("    first log: %s\n", buf);

  memcpy(buf, histEntryData.oLastLog.pioData,
              histEntryData.oLastLog.oLength);
  buf[histEntryData.oLastLog.oLength] = '\0';
  printf("    last log: %s\n", buf);

  memcpy(buf, histEntryData.oID.pioData, histEntryData.oID.oLength);
  buf[histEntryData.oID.oLength] = '\0';
  printf("    ID: %s\n", buf);

  memcpy(buf, histEntryData.oTableQualifier.pioData,
              histEntryData.oTableQualifier.oLength);
  buf[histEntryData.oTableQualifier.oLength] = '\0';
  printf("    table qualifier: %s\n", buf);

  memcpy(buf, histEntryData.oTableName.pioData,
              histEntryData.oTableName.oLength);
  buf[histEntryData.oTableName.oLength] = '\0';
  printf("    table name: %s\n", buf);

  memcpy(buf, histEntryData.oLocation.pioData,
              histEntryData.oLocation.oLength);
  buf[histEntryData.oLocation.oLength] = '\0';
  printf("    location: %s\n", buf);
```

```
  memcpy(buf, histEntryData.oComment.pioData,
              histEntryData.oComment.oLength);
  buf[histEntryData.oComment.oLength] = '\0';
  printf("    comment: %s\n", buf);

  memcpy(buf, histEntryData.oCommandText.pioData,
              histEntryData.oCommandText.oLength);
  buf[histEntryData.oCommandText.oLength] = '\0';
  printf("    command text: %s\n", buf);
  printf("    history file entry ID: %u\n", histEntryData.oEID.ioHID);
  printf("    table spaces:\n");

  for (tsNb = 0; tsNb < histEntryData.oNumTablespaces; tsNb = tsNb + 1)
  {
    memcpy(buf, histEntryData.poTablespace[tsNb].pioData,
                histEntryData.poTablespace[tsNb].oLength);
    buf[histEntryData.poTablespace[tsNb].oLength] = '\0';
    printf("      %s\n", 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);
  memcpy(buf, histEntryData.poEventSQLCA->sqlstate, 5);
  buf[5] = '\0';
  printf("      sqlstate: %s\n", buf);
  memcpy(buf, histEntryData.poEventSQLCA->sqlerrmc,
              histEntryData.poEventSQLCA->sqlerrml);
  buf[histEntryData.poEventSQLCA->sqlerrml] = '\0';
  printf("      message: %s\n", buf);

  return 0;
} /* HistoryEntryDisplay */

int HistoryEntryDataFieldsFree(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb;

  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);
```

```
  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
  {
    free(pHistEntryData->poTablespace[tsNb].pioData);
  }

  free(pHistEntryData->poTablespace);

  return 0;
} /* HistoryEntryDataFieldsFree */

int DbFirstRecoveryHistoryFileEntryUpdate(char dbAlias[],
                                          char user[],
                                          char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint16 recoveryHistoryFileHandle;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;
  char newLocation[DB2HISTORY_LOCATION_SZ + 1];
  char newComment[DB2HISTORY_COMMENT_SZ + 1];
  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);
  if (rc != 0)
  {
    return rc;
  }

  /*****************************************/
  /* 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(db2Version710, &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(db2Version710, &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);

/* update the first history file entry */
rc = DbConn(dbAlias, user, pswd);
if (rc != 0)
{
  return rc;
}

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(db2Version710, &dbHistoryUpdateParam, &sqlca);
DB2_API_CHECK("first history file entry -- update");

rc = DbDisconn(dbAlias);
if (rc != 0)
{
  return rc;
}

/*******************************************/
/* 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(db2Version710, &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(db2Version710, &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(db2Version710, &dbHistoryEntryGetParam, &sqlca);
  DB2_API_CHECK("first recovery history file entry -- read");

  printf("\n  Display the first entry.\n");
  rc = HistoryEntryDisplay(histEntryData);

  /*******************************************/
  /* 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(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);

  return 0;
} /* DbFirstRecoveryHistoryFileEntryUpdate */

int DbRecoveryHistoryFilePrune(char dbAlias[], char user[], char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2PruneStruct histPruneParam;
  char timeStampPart[14 + 1];

  printf("\n*************************************\n");
```

```
  printf("*** PRUNE THE RECOVERY HISTORY FILE ***\n");
  printf("****************************************\n");
  printf("\nUSE THE DB2 API:\n");
  printf("  db2Prune -- Prune Recovery History File\n");
  printf("AND THE SQL STATEMENTS:\n");
  printf("  CONNECT\n");
  printf("  CONNECT RESET\n");
  printf("TO PRUNE THE RECOVERY HISTORY FILE.\n");

  /* Connect to the database: */
  rc = DbConn(dbAlias, user, pswd);
  if (rc != 0)
  {
    return rc;
  }

  /* Prune the recovery history file: */
  printf("\n  Prune the recovery history file for '%s' database.\n",
         dbAlias);

  /* timeStampPart is a pointer to a string specifying a time stamp or
     log sequence number. Time stamp is used here to select records for
     deletion. All entries equal to or less than the time stamp will be
     deleted. */
  histPruneParam.piString = timeStampPart;
  strcpy(timeStampPart, "2010"); /* year 2010 */

  /* The action DB2PRUNE_ACTION_HISTORY removes history file entries: */
  histPruneParam.iAction = DB2PRUNE_ACTION_HISTORY;

  /* The option DB2PRUNE_OPTION_FORCE forces the removal of the last backup: */
  histPruneParam.iOptions = DB2PRUNE_OPTION_FORCE;

  /* db2Prune can be called to delete entries from the recovery history file
     or log files from the active log path. Here we call it to delete
     entries from the recovery history file.
     You must have SYSADM, SYSCTRL, SYSMAINT, or DBADM authority to prune
     the recovery history file. */
  db2Prune(db2Version710, &histPruneParam, &sqlca);
  DB2_API_CHECK("recovery history file -- prune");

  /* Disconnect from the database: */
  rc = DbDisconn(dbAlias);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* DbRecoveryHistoryFilePrune */
```

Anhang F. Befehlszeilen-Script zur Wiederherstellung

Das folgende DB2-Befehls-Script zeigt, wie Sie Befehle des Befehlszeilenprozessor verwenden können:

- Sichern einer Datenbank
- Wiederherstellen der Datenbank
- Akualisierendes Wiederherstellen der Datenbank

Stellen Sie sicher, dass die Beispieldatenbank SAMPLE vorhanden ist und gerade nicht verwendet wird. Zusatzinformationen zur Beispieldatenbank SAMPLE finden Sie im Handbuch *SQL Reference*. Basisinformationen zum DB2-Befehlszeilenprozessor finden Sie im Handbuch *Command Reference*.

Sowohl eine Windows-kompatible als auch eine UNIX-kompatible Version des Scripts werden beschrieben.

Beispielbefehls-Script für Windows-Betriebssysteme

Gehen Sie wie folgt vor, um das Script unter Windows NT oder Windows 2000 auszuführen:

1. Speichern Sie das Script in einer Datei, z. B. mit dem Namen `backrest.db2`.
2. Wenn der Datenbankmanager nicht aktiv ist, setzen Sie in einem DB2-Befehlsfenster den Befehl **db2start** ab. Zum Öffnen eines DB2-Befehlsfensters und zur Initialisierung der DB2-Befehlszeilenumgebung unter dem Windows-Betriebssystem setzen Sie in einer Befehlszeile den Befehl **db2cmd** ab.
3. Geben Sie den Befehl `db2 -f backrest.db2 -t` ein.

Im Folgenden sehen Sie ein Beispiel für die Ausgabe (nur in Englisch verfügbar) dieses Scripts:

```
D:\>db2 -f backrest.db2 -t
This is CLP script: backrest.db2


Deleting old SAMPLE database backup images...

process SAMPLE.0\DB2\NODE0000\CATN0000
process 20010403

Updating the database configuration parameter LOGRETAIN to 'ON'...

DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
```

```
DB21026I  Bei den meisten Konfigurationsparametern müssen die Verbindungen
aller Anwendungen zur Datenbank unterbrochen werden, damit die Änderungen
wirksam werden.


Backing up the SAMPLE database...


Sicherung erfolgreich. Die Zeitmarke für diese Sicherungsabbilddatei
ist: 20010403131027


Restoring the SAMPLE database as TESTBACK (1st pass)...

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.

Listing the table spaces for the TESTBACK database...


           Tabellenbereiche für aktuelle Datenbank

 Tabellenbereichs-ID                 = 0
 Name                                = SYSCATSPACE
 Typ                                 = Vom System verwalteter Bereich
 Inhalt                              = Beliebige Daten
 Status                              = 0x2001100
   Detaillierte Erläuterung:
     Wiederherstellung anstehend
     Speicher muss definiert werden
     Speicher kann definiert werden

 Tabellenbereichs-ID                 = 1
 Name                                = TEMPSPACE1
 Typ                                 = Vom System verwalteter Bereich
 Inhalt                              = Temporäre Systemdaten
 Status                              = 0x2001100
   Detaillierte Erläuterung:
     Wiederherstellung anstehend
     Speicher muss definiert werden
     Speicher kann definiert werden

 Tabellenbereichs-ID                 = 2
 Name                                = USERSPACE1
 Typ                                 = Vom System verwalteter Bereich
 Inhalt                              = Beliebige Daten
 Status                              = 0x2001100
   Detaillierte Erläuterung:
     Wiederherstellung anstehend
     Speicher muss definiert werden
     Speicher kann definiert werden
```

```
Defining new table space containers for Tablespace 2...

DB20000I  Der Befehl SET TABLESPACE CONTAINERS wurde erfolgreich ausgeführt.

Listing table space containers for Tablespace 2 (TESTBACK database)...


         Tabellenbereichsbehälter für Tabellenbereich 2

 Behälter-ID                          = 0
 Name                                 = c:\ts2con1
 Typ                                  = Pfad


Restoring the SAMPLE database as TESTBACK (2nd pass)...

DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.

Rolling the TESTBACK database forward...


                Status der aktualisierenden Wiederherstellung

 Aliasname der Eingabedatenbank               = testback
 Anzahl der Knoten, die Status zurückgaben    = 1

 Knotennummer                                 = 0
 Aktualisierende Wiederherstellung: Status    = nicht anstehende
 Nächste zu lesende Protokolldatei            =
 Verarbeitete Protokolldateien                =  -
 Letzte festgeschriebene Transaktion          = 2001-04-03-03.16.07.000000

DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.

Dropping the TESTBACK database...

DB20000I  Der Befehl DROP DATABASE wurde erfolgreich ausgeführt.

Terminating the command line processor's back-end process...

DB20000I  Der Befehl TERMINATE wurde erfolgreich ausgeführt.


D:\>
```

Beispielbefehls-Script für Windows-Betriebssysteme

Im Folgenden sehen Sie das Listing des Quellcodes für das Script:

```
-- Before proceeding, ensure that:
--   The database manager is running
--   The SAMPLE database exists and is not in use.

-- Run the script by issuing:
--   db2 -f backrest.db2 -t
--     where -f tells the command line processor to read command input
--               from a file instead of from standard input, and
--           -t tells the command line processor to use a semicolon (;)
--              as the statement termination character.

!ECHO This is CLP script: backrest.db2;

-- Ensure that the DB2 profile registry variable DB2_ENABLE_LDAP
--   is set to 'NO':
!db2set DB2_ENABLE_LDAP=NO;

!ECHO Deleting old SAMPLE database backup images...;
!rd! SAMPLE.0\DB2\NODE0000\CATN0000;

!ECHO Updating the database configuration parameter LOGRETAIN to 'ON'...;
update db cfg for sample using logretain on;

!ECHO Backing up the SAMPLE database...;
backup db sample;

!ECHO Restoring the SAMPLE database as TESTBACK (1st pass)...;
restore db sample into testback redirect;

!ECHO Listing the table spaces for the TESTBACK database...;
list tablespaces;

!ECHO Defining new table space containers for Tablespace 2...;
set tablespace containers for 2 using (path "c:\ts2con1");

!ECHO Listing table space containers for Tablespace 2 (TESTBACK database)...;
list tablespace containers for 2;

!ECHO Restoring the SAMPLE database as TESTBACK (2nd pass)...;
restore db sample continue;

!ECHO Rolling the TESTBACK database forward...;
rollforward db testback stop;

!ECHO Dropping the TESTBACK database...;
drop db testback;

!ECHO Terminating the command line processor's back-end process...;
terminate;

-- End file
```

Beispielbefehls-Script für UNIX-Systeme

Gehen Sie wie folgt vor, um das Script auf einem UNIX-System auszuführen:

1. Speichern Sie das Script in einer Datei, z. B. mit dem Namen `backrest.db2`.
2. Wenn der Datenbankmanager nicht aktiv ist, setzen Sie an einer Eingabeaufforderung den Befehl **db2start** ab.
3. Geben Sie den Befehl `db2 -f backrest.db2 -t` ein.

Im Folgenden sehen Sie ein Beispiel für die Ausgabe dieses Scripts:

```
sunfish /export/home2/falexand/samples/clp>db2 -f backrest.db2 -t
This is CLP script: backrest.db2

Deleting old SAMPLE database backup images...

Updating the database configuration parameter LOGRETAIN to 'ON'...
DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
DB21026I  Bei den meisten Konfigurationsparametern müssen die Verbindungen
aller Anwendungen zur Datenbank unterbrochen werden, damit die Änderungen
wirksam werden.

Backing up the SAMPLE database...

Sicherung erfolgreich. Die Zeitmarke für diese Sicherungsabbilddatei
ist: 20010531172525

Restoring the SAMPLE database as TESTBACK (1st pass)...
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.

Listing the table spaces for the TESTBACK database...

           Tabellenbereiche für aktuelle Datenbank

 Tabellenbereichs-ID                 = 0
 Name                                = SYSCATSPACE
 Typ                                 = Vom System verwalteter Bereich
 Inhalt                              = Beliebige Daten
 Status                              = 0x2001100
   Detaillierte Erläuterung:
     Wiederherstellung anstehend
     Speicher muss definiert werden
     Speicher kann definiert werden

 Tabellenbereichs-ID                 = 1
 Name                                = TEMPSPACE1
 Typ                                 = Vom System verwalteter Bereich
 Inhalt                              = Temporäre Systemdaten
 Status                              = 0x2001100
   Detaillierte Erläuterung:
```

```
    Wiederherstellung anstehend
    Speicher muss definiert werden
    Speicher kann definiert werden

 Tabellenbereichs-ID                = 2
 Name                               = USERSPACE1
 Typ                                = Vom System verwalteter Bereich
 Inhalt                             = Beliebige Daten
 Status                             = 0x2001100
   Detaillierte Erläuterung:
     Wiederherstellung anstehend
     Speicher muss definiert werden
     Speicher kann definiert werden


Defining new table space containers for Tablespace 2...
DB20000I  Der Befehl SET TABLESPACE CONTAINERS wurde erfolgreich ausgeführt.

Listing table space containers for Tablespace 2 (TESTBACK database)...

         Tabellenbereichsbehälter für Tabellenbereich 2

 Behälter-ID                        = 0
 Name                               = /export/home2/falexand/falexand...
                                        .../NODE0000/SQL00020/ts2con1
 Typ                                = Pfad


Restoring the SAMPLE database as TESTBACK (2nd pass)...
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.

Rolling the TESTBACK database forward...

               Status der aktualisierenden Wiederherstellung

 Aliasname der Eingabedatenbank             = testback
 Anzahl der Knoten, die Status zurückgaben  = 1
 Knotennummer                               = 0
 Aktualisierende Wiederherstellung: Status  = nicht anstehende
 Nächste zu lesende Protokolldatei          =
 Verarbeitete Protokolldateien              =  -
 Letzte festgeschriebene Transaktion        = 2001-05-29-21.20.16.000000

DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.

Dropping the TESTBACK database...
DB20000I  Der Befehl DROP DATABASE wurde erfolgreich ausgeführt.

Terminating the command line processor's back-end process...
DB20000I  Der Befehl TERMINATE wurde erfolgreich ausgeführt.
```

Im Folgenden sehen Sie das Listing des Quellcodes für das Script:

```
-- Before proceeding, ensure that:
--   The database manager is running
--   The SAMPLE database exists and is not in use.

-- Run the script by issuing:
--   db2 -f backrest.db2 -t
--     where -f tells the command line processor to read command input
--               from a file instead of from standard input, and
--           -t tells the command line processor to use a semicolon (;)
--             as the statement termination character.

echo This is CLP script: backrest.db2;
-- Ensure that the DB2 profile registry variable DB2_ENABLE_LDAP
--   is set to 'NO':
!db2set DB2_ENABLE_LDAP=NO;

echo Deleting old SAMPLE database backup images...;
!rm -f ./SAMPLE.0.*;

echo Updating the database configuration parameter LOGRETAIN to 'ON'...;
update db cfg for sample using logretain on;

echo Backing up the SAMPLE database...;
backup db sample;

echo Restoring the SAMPLE database as TESTBACK (1st pass)...;
restore db sample into testback redirect;

echo Listing the table spaces for the TESTBACK database...;
list tablespaces;

echo Defining new table space containers for Tablespace 2...;
set tablespace containers for 2 using (path "ts2con1");

echo Listing table space containers for Tablespace 2 (TESTBACK database)...;
list tablespace containers for 2;

echo Restoring the SAMPLE database as TESTBACK (2nd pass)...;
restore db sample continue;

echo Rolling the TESTBACK database forward...;
rollforward db testback stop;

echo Dropping the TESTBACK database...;
drop db testback;

echo Terminating the command line processor's back-end process...;
terminate;

-- End file
```

Beispielbefehls-Script für UNIX-Systeme

Anhang G. Tivoli Storage Manager

Wenn Sie das DB2-Sicherungs- oder DB2-Wiederherstellungsdienstprogramm aufrufen, können Sie angeben, dass Sie das Produkt Tivoli Storage Manager (TSM, früher ADSM) verwenden wollen, um die Sicherungs- oder Wiederherstellungsoperation zu verwalten. Sie können den TSM-Client Version 3.1.x.3 und höher mit DB2 verwenden.

Einrichten eines Tivoli Storage Manager-Clients auf UNIX-Plattformen

Bevor der Datenbankmanager die Option TSM verwenden kann, müssen Sie die folgenden Konfigurationsschritte ausführen:

1. Führen Sie in SunOS- und Solaris-Umgebungen die folgenden Schritte aus. (Bei anderen UNIX-Plattformen beginnen Sie mit Schritt 2.)
   a. Stellen Sie sicher, dass die erforderliche Version des Betriebssystems installiert ist: SunOS 5.5.1 bzw. Solaris 2.5.1.
   b. Installieren Sie den TSM-Client Version 3.1.x.3 oder höher. Stellen Sie sicher, dass Sie *alle* früheren TSM-Pakete entfernen, bevor Sie diese Client-Version installieren.
   c. Überprüfen Sie, ob TSM in den Verzeichnissen `/opt/IBMDSMap5`, `/opt/IBMDSMba5` und `/opt/IBMDSMsa5` installiert ist.
   d. Erstellen Sie die folgenden symbolischen Verbindungen im Verzeichnis `/usr/lib`, sofern diese nicht bereits vorhanden sind:

      ```
      libApiDS.so -> libApiDS.so.1
      libApiDS.so.1 -> /opt/IBMDSMap5/api/libApiDS.so.2
      ```

2. Erstellen oder ändern Sie die Datei mit den TSM-Benutzerkonfigurationsoptionen `/usr/sbin/dsm.opt` und die Datei mit den TSM-Systemkonfigurationsoptionen `/usr/sbin/dsm.sys` so, dass sie für Ihre Umgebung geeignet sind.
3. Führen Sie in SunOS- und Solaris-Umgebungen die folgenden Schritte aus. (Bei anderen UNIX-Plattformen fahren Sie mit Schritt 4 fort.)
   a. Kopieren Sie `/usr/sbin/dsm.opt` und `/usr/sbin/dsm.sys` in das Verzeichnis `/opt/IBMDSMap5`.
   b. Kopieren Sie `/opt/IBMDSMap5/solaris/dsmaptica` in das Verzeichnis `/opt/IBMDSMap5`.
4. Definieren Sie die von TSM verwendeten Umgebungsvariablen:

   DSMI_DIR Gibt den benutzerdefinierten Verzeichnispfad an, in dem sich die API-Datei für den Trusted Agent (`dsmapicta` oder `dsmtca`) befindet. In SunOS- und Solaris-Umgebungen muss dieser Wert auf `/opt/IBMDSMap5` gesetzt sein.

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.

In SunOS- und Solaris-Umgebungen muss dieser Wert auf `/opt/IBMDSMap5/dsm.opt` gesetzt werden.

DSMI_LOG Gibt den benutzerdefinierten Verzeichnispfad an, in dem das Fehlerprotokoll (`dsierror.log`) erstellt wird.

5. Definieren Sie das TSM-Kennwort.

   Damit ein TSM-Client mit einem TSM-Server Daten austauschen kann, benötigt er ein Kennwort für den Server. Die ausführbare Datei `dsmapipw` ist im Verzeichnis `INSTHOME/sqllib/adsm` des Exemplareigners installiert. Mit Hilfe dieser ausführbaren Datei können Sie das TSM-Kennwort definieren und zurücksetzen.

   Um den Befehl `dsmapipw` ausführen zu können, müssen Sie mit der Berechtigung „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. (Beachten Sie, dass Sie zweimal zur Eingabe des neuen Kennworts aufgefordert werden, um Eingabefehler festzustellen.)

   Anmerkung: Benutzer, die das Sicherungsdienstprogramm oder das Wiederherstellungsdienstprogramm aufrufen, brauchen dieses Kennwort nicht zu kennen. Sie müssen den Befehl `dsmapipw` nur aufrufen, wenn Sie für die erste Verbindung ein Kennwort festlegen, und wenn das Kennwort auf dem TSM-Server zurückgesetzt worden ist.

6. Gehen Sie wie folgt vor, wenn der Datenbankmanager aktiv ist:
   - Stoppen Sie den Datenbankmanager mit dem Befehl **db2stop**.
   - Starten Sie den Datenbankmanager mit dem Befehl **db2start**.

Einrichten eines Tivoli Storage Manager-Clients auf anderen Plattformen

Bevor der Datenbankmanager die Option TSM verwenden kann, müssen Sie die folgenden Konfigurationsschritte ausführen:

1. Definieren Sie die von TSM verwendeten Umgebungsvariablen:

   DSMI_DIR Gibt den benutzerdefinierten Verzeichnispfad an, in dem sich die API-Datei für den Trusted Agent (`dsmapicta` oder `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.

2. Erstellen (oder ändern) Sie die Datei mit den TSM-Systemkonfigurationsoptionen (`dsm.sys`), falls dies für Ihr Betriebssystem erforderlich ist.
3. Erstellen (oder ändern) Sie die Datei mit den TSM-Benutzerkonfigurationsoptionen (`dsm.opt`). Die Umgebungsvariable `DSMI_CONFIG` verweist auf diese Datei.
4. Definieren Sie das TSM-Kennwort.

   Damit ein TSM-Client mit einem TSM-Server Daten austauschen kann, benötigt er ein Kennwort für den Server. 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 mit der Berechtigung des lokalen Administrators 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 aufrufen, 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. (Beachten Sie, dass Sie zweimal zur Eingabe des neuen Kennworts aufgefordert werden, um Eingabefehler festzustellen.)

Anmerkung: Benutzer, die das Sicherungsdienstprogramm oder das Wiederherstellungsdienstprogramm aufrufen, brauchen dieses Kennwort nicht zu kennen. Sie müssen den Befehl `dsmapipw` nur aufrufen, wenn Sie für die erste Verbindung ein Kennwort festlegen, und wenn das Kennwort auf dem TSM-Server zurückgesetzt worden ist.

5. Gehen Sie wie folgt vor, wenn der Datenbankmanager aktiv ist:
   - Stoppen Sie den Datenbankmanager mit dem Befehl **db2stop**.
   - Starten Sie den Datenbankmanager mit dem Befehl **db2start**.

Überlegungen zur 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 den Betriebssystemen Windows NT und OS/2 das Zeichen\ statt / eingeben müssen.) Vollständig qualifizierte Pfadnamen:

- Der Pfadname eines Objekts für eine Datenbankgesamtsicherung setzt sich wie folgt zusammen:
  `/<datenbank>/NODEnnnn/FULL_BACKUP.zeitmarke.folgenr`
- Der Pfadname eines Objekts für eine Tabellenbereichssicherung setzt sich wie folgt zusammen:
  `/<datenbank>/NODEnnnn/TSP_BACKUP.zeitmarke.folgenr`
- Der Pfadname eines Objekts für eine Ladekopie setzt sich wie folgt zusammen: `/<datenbank>/NODEnnnn/LOAD_COPY.zeitmarke.folgenr`

Hier steht `<datenbank>` für den Aliasnamen der Datenbank und `NODEnnnn` für die Knotennummer. Die Namen in Großbuchstaben müssen Sie genau wie dargestellt eingeben.

- Wenn Sie mehrere Sicherungsimages mit demselben Aliasnamen der Datenbank 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.
- Einzelne Sicherungsimages sind der grafischen TSM-Benutzerschnittstelle nicht bekannt. 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 (siehe „db2adutl - Mit von TSM archivierten Images arbeiten” auf Seite 348).

- Auf dem TSM-Server kommt es zu einer Zeitlimitüberschreitung einer Sitzung, wenn der Tivoli-Client innerhalb des Zeitraums, der im Parameter COMMTIMEOUT der Konfigurationsdatei des Servers angegeben wurde, nicht antwortet. 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.
- Der Datenbankmanager verwendet die TSM-Option für die Gesamtsicherung; TSM-Teilsicherungsoperationen werden nicht unterstützt.
- Verwenden Sie mehrere Sitzungen, um den Durchsatz zu erhöhen.
- Bei Nicht-UNIX-Systemen lassen die Sicherungs- und Wiederherstellungsdienstprogramme nur eine einzige TSM-Sitzung zu.

Die aktuellen Tivoli-Clients unter den Windows-Betriebssystemen und OS/2 sind simultan verwendbar. Deshalb können Sie mit den Sicherungs-, Wiederherstellungs- und Ladedienstprogrammen auf einer einzelnen Maschine mehrere E/A-Sitzungen starten. Die Benutzer müssen allerdings bestätigen, dass die installierte Version des TSM-Clients diese Funktion unterstützt.

Angenommen, ein Benutzer versucht in einer Konfiguration mit einem Einzelknoten, einen Datenbanksicherungsbefehl wie den folgenden abzusetzen:

```
db2 backup db sample use tsm open 3 sessions
```

Daraufhin erkennt DB2, dass Mehrfachsitzungen von TSM nicht unterstützt werden und gibt eine Fehlernachricht zurück. (Dies gilt auch für Ladeoperationen, die unter Verwendung von TSM mit der Option COPY YES aufgerufen werden.)

Hierbei ist zu beachten, dass DB2 in einer Konfiguration mit mehreren logischen Knoten (MLN - Multiple Logical Node) unter Windows NT möglicherweise nicht erkennen kann, dass auf einer einzelnen Maschine mehrere Sitzungen verwendet werden, wenn jeder logische Knoten versucht, lediglich eine Sitzung zu verwenden. Aus diesem Grund ist es in MLN-Konfigurationen sehr wichtig, sicherzustellen, dass der verwendete TSM-Client die simultane Verwendung unterstützt.

Während mehrere logische Knoten unter Verwendung von TSM parallel gesichert, wiederhergestellt oder geladen werden, lässt DB2 die Fortsetzung der Operation zu, wenn die einzelnen Knoten versuchen, eine Einzelsitzung zu verwenden. Dies gilt auch, wenn die logischen Knoten sich in Wirklichkeit auf derselben Hardware befinden. Dadurch können Sicherungsversuche und Ladevorgänge fehlschlagen. Sie sollten dies also nicht versuchen, ohne dass die aktuellste Version des TSM-Clients auf dem System installiert ist.

Verwalten von Sicherungen und Protokollarchiven unter TSM

Das Dienstprogramm **db2adutl** ermöglicht das Abfragen, Extrahieren und Löschen von Sicherungsimages, Protokollen und Ladekopieimages, die mit TSM gesichert wurden. Das Dienstprogramm ist auf UNIX-Plattformen im Verzeichnis `INSTHOME/sqllib/misc` und unter Windows-Betriebssystemen und unter OS/2 im Verzeichnis `\sqllib\misc` installiert. Weitere Informationen zu diesem Dienstprogramm finden Sie in „db2adutl - Mit von TSM archivierten Images arbeiten” auf Seite 348.

Mit der Option QUERY können Sie Sicherungsimages, Ladekopieimages oder Protokolle auflisten. Sie können einen Bereich von aufzulistenden Protokollen auswählen. Außerdem können Sie die Anzeige der inaktiven Sicherungsimages anfordern.

Mit der Option EXTRACT können Sie Sicherungsimages aus TSM in das aktuelle Verzeichnis kopieren. Sie können auswählen, welche Sicherungsimages oder Protokolle Sie extrahieren.

Mit der Option DELETE können Sie Sicherungsimages inaktivieren oder Protokolle aus TSM löschen. Sie können auswählen, welche Sicherungsimages oder Protokolle verarbeitet werden. Mit der Option KEEP *n* können Sie die neuesten *n* Sicherungsimages beibehalten. Außerdem können Sie mit der Option OLDER THAN *zeitmarke* oder *n* DAYS die DELETE-Anforderung anpassen.

Integration von Tivoli Space Manager und Data Links

DB2 Data Links Manager kann Tivoli Space Manager (TSM) und dessen virtuelles Dateisystem (FSM) verwenden, das sich über das native Journaled File System (JFS) legt. Der Zugriff auf FSM und die Konfiguration von FSM erfolgen auf dieselbe Weise wie bei JFS.

Diese Einrichtung ist nützlich für Benutzer von Dateisystemen mit großen Dateien, die periodisch auf einen tertiären Datenträger versetzt werden müssen. Der Speicherplatz für diese Dateisysteme muss regelmäßig verwaltet werden. Mit der TSM-Unterstützung von DB2 Data Links Manager erhalten Sie eine größere Flexibilität bei der Verwaltung des Speicherplatzes für DATALINK-Dateien. Statt vorab genügend Speicher im Dateisystem von DB2 Data Links Manager für alle Dateien zuzuordnen, die dort möglicherweise gespeichert werden, können Sie mit TSM Zuordnungen des von Data Links verwalteten Dateisystems während eines Zeitraums einstellen. Dadurch vermeiden Sie die Gefahr, im normalen Betrieb versehentlich das Dateisystem zu vollständig belegen.

Rahmenbedingungen und Einschränkungen

- Diese Funktion wird zur Zeit nur unter AIX unterstützt.
- Die selektive Migration (mit **dsmmigrate**) und der Rückruf einer FC-Datei (Datenbank für Lesezugriff) sollte nur der Root ausführen.

  Die selektive Migration kann nur vom Dateieigner ausgeführt werden. Im Fall der Datenbankdateien für den Lesezugriff ist dies der DataLink Manager-Administrator (dlfm). Zum Zugriff auf derartige Dateien ist ein Token von der Host-Datenbank erforderlich. Der einzige Benutzer, der kein Token benötigt ist der Root. Für den Root ist es einfacher, die selektive Migration und den Rückruf für solche Dateien auszuführen. Der Benutzer dlfm kann nur eine FC-Datei migrieren, wenn er beim ersten Mal ein gültiges Token verwendet. Beim zweiten Versuch (nach einem erneuten Aufruf) schlägt die Migration fehl, und die Nachricht ANS1028S mit dem Hinweis auf einen internen Programmfehler und der Aufforderung, sich an den IBM Ansprechpartner zu wenden, wird ausgegeben. Wenn ein Nicht-Root versucht, den Befehl **dsmmigrate** für eine FC-Datei auszuführen, schlägt dies fehl. Diese Einschränkung ist unbedeutend, da typischerweise Administratoren auf Dateien des Dateiservers zugreifen.
- Die Systemaufrufe **stat** und **statfs** zeigen *Vfs-type* als `fsm`, und nicht als `dlfs`, obwohl dlfs über fsm gemountet ist.

  Dieses Verhalten unterstützt die normale Funktionalität von **dsmrecalld**-Dämonen, die für das Dateisystem den Befehl **statfs** ausführen, um zu bestimmen, ob `fsm` der zugehörige Typ *Vfs-type* ist.
- Der Befehl **dsmls** gibt nichts aus, falls eine Datei, die die Mindestzahl *inode* aufweist, eine FC-Datei (Datenbank für Lesezugriff) ist.

  Der Befehl **dsmls** ähnelt dem Befehl **ls**: Er listet die Dateien auf, die von TSM verwaltet werden. Es ist keine Benutzeraktion erforderlich.

Anhang H. Benutzer-Exit zur Datenbankwiederherstellung

Sie können ein *Benutzer-Exit-Programm* entwickeln, um die Archivierung und den Abruf von Protokolldateien zu automatisieren. (Unter OS/2 können Sie Benutzer-Exit-Programme auch für Sicherungs- und Wiederherstellungsoperationen verwenden.) Bevor Sie ein Benutzer-Exit-Programm zur Archivierung oder zum Abruf von Protokolldateien aufrufen, muss der Datenbankkonfigurationsparameter *userexit* auf `YES` gesetzt sein. Außerdem kann dadurch die Datenbank aktualisierend wiederhergestellt werden.

Wenn Sie ein Benutzer-Exit-Programm aufrufen, übergibt der Datenbankmanager die Steuerung an die ausführbare Datei `db2uext2`. (Unter OS/2 rufen Sicherungs- und Wiederherstellungsoperationen zuerst die Datei `db2uexit.cmd` auf, die ihrerseits die ausführbare Datei `db2uext2` aufruft.) 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 Benutzer-Exit-Programm Fehlerbedingungen verarbeiten können (siehe „Fehlerbehandlung" auf Seite 499). Und da Sie innerhalb eines Datenbankmanagerexemplars nur ein Benutzer-Exit-Programm aufrufen können, muss es für jede Operation, die es möglicherweise ausführen soll, einen Abschnitt aufweisen.

Die folgenden Themen werden behandelt:

- „Beispiele für Benutzer-Exit-Programme"
- „Aufrufformat" auf Seite 496
- „Hinweise zum Sichern und Wiederherstellen (nur DB2 für OS/2)" auf Seite 498
- „Fehlerbehandlung" auf Seite 499

Beispiele für Benutzer-Exit-Programme

Beispiel-Benutzer-Exit-Programme 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 Benutzer-Exit-Programme 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 Beispiel-Benutzer-Exit-Programme, die mit DB2 ausgeliefert werden.

- **UNIX-Systeme**

  Die Benutzer-Exit-Beispielprogramme für DB2 für UNIX-Systeme, befinden sich im Unterverzeichnis `sqllib/samples/c`. Obwohl die Beispielprogramme in der Programmiersprache C codiert sind, können Sie das Benutzer-Exit-Programm auch in einer anderen Programmiersprache schreiben.

  Das Benutzer-Exit-Programm muss eine ausführbare Datei sein, deren Name `db2uext2` lautet.

  Folgende vier Benutzer-Exit-Beispielprograme für UNIX-Systeme sind verfügbar:

  – `db2uext2.cadsm`

    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 Festplatten, um Datenbankprotokolldateien zu archivieren und abzurufen.

  – `db2uxt2.cxbsa`

    Dieses Beispiel verwendet das Programm Legato NetWorker** Version 4.2.5 von Legato** Systems Inc. zum Archivieren und Abrufen von Datenbankprotokolldateien. Dieses Beispielprogramm wird nur unter AIX unterstützt.

- **Windows-Betriebssysteme**

  Die Benutzer-Exit-Beispielprogramme für DB2 für Windows-Betriebssysteme befinden sich im Unterverzeichnis `sqllib\samples\c`. Obwohl die Beispielprogramme in der Programmiersprache C codiert sind, können Sie das Benutzer-Exit-Programm auch in einer anderen Programmiersprache schreiben.

  Das Benutzer-Exit-Programm muss eine ausführbare Datei sein, deren Name `db2uext2` lautet.

  Für Windows-Betriebssysteme sind zwei Benutzer-Exit-Beispielprogramme verfügbar:

  – `db2uext2.cadsm`

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.

- **OS/2**

Die Benutzer-Exit-Beispielprogramme für DB2 für OS/2 befinden sich im Exemplarunterverzeichnis des Verzeichnisses `\sqllib\samples\rexx`. (Das Programm `dbuexit.CAD` bildet eine Ausnahme: Es befindet sich im Exemplarunterverzeichnis des Verzeichnisses `\sqllib\samples\c`.) Obwohl die Beispielprogramme in der Regel in der Programmiersprache REXX codiert sind, können Sie das Benutzer-Exit-Programm auch in einer anderen Programmiersprache schreiben.

Das Beispielprogramm, das Sie zur Implementierung auswählen, müssen Sie in `db2uexit` mit der Erweiterung `.cmd` oder `.exe` umbenennen. Versetzen Sie die umbenannte Datei in das Verzeichnis `\sqllib\bin`.

Fünf OS/2-Benutzer-Exit-Beispielprogramme sind verfügbar:

– `db2uexit.ex1`

Dieses Beispiel verwendet das Programm Sytos Premium** Version 2.2, das von Seagate** Software Inc. erhältlich ist, um Daten auf einer externen IBM Bandeinheit zu speichern und davon abzurufen. Nur Version 2.2 des Produkts Sytos Premium wird zurzeit unterstützt. (Sie benötigen OS/2 FixPak 26, um dieses Produkt einsetzen zu können.) Dem Listing des Beispielprogramms können Sie weitere Voraussetzungen entnehmen.

– `db2uexit.ex2`

Dieses Beispiel verwendet das Programm Filesafe** von Mountain** Corporation. Sie können es verwenden, um Daten auf einer externen Mountain-Bandeinheit zu speichern oder davon abzurufen. Jeder Sicherungskopie einer Datenbank wird ein eindeutiger Datenträger-Header zugeordnet, so dass mehrere Sicherungsimages einer oder mehrerer Datenbanken auf dem gleichen Band gespeichert werden können.

– `db2uexit.ex3`

In diesem Beispiel wird das Programm MaynStream** von Maynard** Corporation verwendet. Sie können es verwenden, um Daten auf einer externen Maynard-Bandeinheit zu speichern oder davon abzurufen. MaynStream bietet keine Unterstützung für das Umleiten der Datenbankwiederherstellungsoperation auf ein anderes Laufwerk als das, auf dem die Datenbank gesichert wurde.

– `db2uexit.ex4`

In diesem Beispielprogramm wird der OS/2-Befehl XCOPY verwendet. Als Speichereinheit können Sie jede von OS/2 unterstützte Einheit wie

Festplatten, Diskettenlaufwerke oder Laufwerke für optische Kassetten verwenden. Diese Einheiten können umgeleitete LAN-Laufwerke sein, sofern die Workstation entsprechend konfiguriert ist.

XCOPY kann zum Sichern und Wiederherstellen von Datenbanken nicht verwendet werden.

– `db2uexit.CAD`

Dieses in C geschriebene Beispielprogramm ist äquivalent zum Beispielprogramm von Tivoli Storage Manager (TSM). Sie können damit Datenbankprotokolldateien archivieren und abrufen.

Aufrufformat

Wenn der Datenbankmanager ein Benutzer-Exit-Programm aufruft, übergibt er einen Satz von Parametern (mit dem Datentyp CHAR) an das Programm. Das Aufrufformat hängt vom eingesetzten Betriebssystem ab.

- Aufrufformat für UNIX-Systeme oder für Windows NT/Windows 2000:

```
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`, `Dynix/ptx`, `SGI` und `NT`.

db2rel Gibt den Releasestand von DB2 an. Beispiel: `SQL07020`

anforderung Gibt die Anforderungsart 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 sein, wird dieser Wert an das Benutzer-Exit-Programm übergeben.)

startseite Gibt die Zahl der relativen 4-KB-Adress-Seiten 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.

- Aufrufformat für OS/2:

```
aktion laufwerk db-aliasname protokollpfad protokolldatei bezugswert
```

aktion Gültige Werte: `BACKUP`, `RESTORE`, `ARCHIVE` und `RETRIEVE`.

laufwerk Gibt für eine Sicherungsoperation das Laufwerk an, auf dem sich die zu sichernde Datenbank befindet. Gibt für eine Wiederherstellungsoperation das Laufwerk an, auf dem die Datenbank wiederhergestellt werden soll. Gibt für eine Archivierungs- oder Abrufoperation das Laufwerk an, auf dem sich die Datenbank befindet. In beiden Fällen geben Sie einen Laufwerkbuchstaben, gefolgt von einem Doppelpunkt, ein (z. B. `C:`).

db-aliasname Gibt den Aliasnamen der Datenbank an (oder den Datenbanknamen, falls kein Aliasname vorhanden ist).

protokollpfad Gibt für eine Sicherungs- oder Wiederherstellungsoperation den vollständig qualifizierten Namen der Antwortdatei an, die eine Liste mit zu sichernden oder wiederherzustellenden Dateien enthält. Die einzelnen Namen in der Liste sind vollständig qualifizierte Dateinamen, die Platzhalterzeichen enthalten können.

Für Wiederherstellungsoperationen geben der Laufwerkbuchstabe und der Pfad das Quellenlaufwerk und den Quellenpfad an, von dem die Datenbankdatei gesichert wurde. Wenn z. B. in der Antwortdatei der Pfad `C:\SQLUTIL\dbname.MH1` enthalten ist, bedeutet dies, dass die Datei `dbname.MH1` aus dem Verzeichnis `C:\SQLUTIL` gesichert wurde.

Gibt für eine Archivierungs- oder Abrufoperation das Protokollpfadverzeichnis an. Beispiel: `C:\SQL00001\SQLOGDIR\`

protokolldatei Gibt für eine Sicherungsoperation die Datenträgerbezeichnung an, die vom Sicherungsdienstprogramm generiert wurde. Dieser Kennsatz besteht aus dem Aliasnamen der Datenbank und aus einer Zeitmarke. Gibt für eine Wiederherstellungsoperation den Pfadnamen des Datenbankunterverzeichnisses an, in dem die Dateien wiederhergestellt werden sollen. Der Laufwerkbuchstabe ist nicht Teil dieser Angabe, da er im Parameter *laufwerk* angegeben wird. Das Format lautet: `\SQL`*`nnnnn`*`\`

Gibt für eine Archivierungs- oder Abrufoperation den Namen der Protokolldatei an. Beispiel: `S0000001.LOG`

bezugswert Gibt einen Bezugswert an, der die Verwendung mehrerer Aufrufe während einer Sicherungs- oder Wiederherstellungsoperation ermöglicht. Der erste Aufruf weist den Zeichenwert `1` auf, und nachfolgende Aufrufe weisen den Zeichenwert `2` auf.

Das Benutzer-Exit-Programm wird bei einer Sicherungs- oder Wiederherstellungsoperation mehrmals aufgerufen. Beim ersten Aufruf werden die Datenträger-Header (`.MH`*n*-Dateien) gesichert bzw. wiederhergestellt, während beim zweiten Aufruf der gesamte Satz von Datenbankdateien gesichert bzw. wiederhergestellt wird.

Dieser Parameter wird für Archivierungs- oder Abrufoperationen nicht verwendet.

Hinweise zum Sichern und Wiederherstellen (nur DB2 für OS/2)

Folgende Hinweise und Informationen gelten für das Schreiben eines Benutzer-Exit-Programms, das vom Sicherungs- oder Wiederherstellungsdienstprogramm aufgerufen wird:

- Ein Rückkehrcode ungleich null, der von einem Benutzer-Exit-Programm übergeben wird, bewirkt, dass das Dienstprogramm fehlschlägt und kein erneuter Versuch unternommen wird.
- Verwenden Sie in vollständig qualifizierten Dateinamen nur unterstützte Platzhalterzeichen. Gültige Suchbedingungen sind z. B. `C:\SQL00001\*.*` und `C:\*.MH*`.
- Das Benutzer-Exit-Programm muss das Format der Antwortdatei verarbeiten können. Bei diesem Format steht ein vollständig qualifizierter Dateiname in jeder Zeile, wobei diese am Zeilenende durch ein Rücklauf-Zeilenvorschubzeichen (CRLF) abgeschlossen ist. Die Datei enthält kein Dateiendezeichen (EOF).
- Wenn Sie mehrere Sicherungsimages für dieselbe Datenbank auf einem Datenträger speichern wollen, muss das Benutzer-Exit-Programm bei der Wiederherstellungsoperation das richtige Image auswählen können. Weitere Informationen zu `db2uexit.ex2` finden Sie in „Beispiele für Benutzer-Exit-Programme” auf Seite 493.
- Zwei gleichzeitig ablaufende Sicherungsoperationen, die auf dieselbe Sicherungseinheit zugreifen, müssen serialisiert werden.
- Wenn sich ein Sicherungsimage über mehrere Datenträger erstreckt, muss die Anforderung der verschiedenen Datenträger ebenfalls vom Benutzer-Exit-Programm oder von einer von ihm aufgerufenen Anwendung bearbeitet werden. Damit diese Funktion unterstützt wird, öffnen das Sicherungs-

und das Wiederherstellungsdienstprogramm zum Aufrufen des Benutzer-Exit-Programms eine Vordergrundsitzung des Betriebssystems.

- Das Benutzer-Exit-Programm darf kein Unterverzeichnis innerhalb des Datenbankverzeichnisses sichern.
- Beim Wiederherstellen einer Datenbank mit einem Benutzer-Exit-Programm verlangt das Wiederherstellungsdienstprogramm die vollständige Steuerung dieser Datenbank. Allerdings kann die Workstation über aktive Verbindungen zu anderen Datenbanken als der wiederherzustellenden Datenbank verfügen.
- Wenn Sie eine Datenbank mit einem Benutzer-Exit-Programm sichern oder wiederherstellen und eine andere Operation auf dieselbe Bandeinheit zugreift, könnte die Sicherungs- oder Wiederherstellungsoperation fehlschlagen, und Sie müssen sie erneut starten. Diese Situation können Sie vermeiden, indem Sie sicherstellen, dass keine anderen Datenbanken, die das Benutzer-Exit-Programm zum Protokollieren aufrufen, verwendet werden, solange eine Sicherungs- oder Wiederherstellungsoperation läuft. Oder Sie stellen sicher, dass das Benutzer-Exit-Programm die Sicherungs- oder Wiederherstellungsoperation zu einem späteren Zeitpunkt erneut versucht, falls die Einheit nicht bereit ist.
- Während der Wiederherstellungsoperation können sich die Angaben zu Laufwerk und Pfad von den während der Sicherungsoperation gemachten Angaben unterscheiden. Wenn z. B. die Datei `dbname.MH1` aus `C:\SQLUTIL` gesichert wird, können Sie sie im Verzeichnis `D:\SQLUTIL2` wiederherstellen.

Fehlerbehandlung

Das Benutzer-Exit-Programm sollte so konzipiert sein, dass spezifische und aussagekräftige Rückkehrcodes zurückgegeben werden, damit der Datenbankmanager diese richtig interpretieren kann. Da das Benutzer-Exit-Programm 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.

Unter OS/2 bewirken alle Codes ungleich null, die von einem Benutzer-Exit-Programm zurückgegeben werden, dass das Sicherungs- oder Wiederherstellungsdienstprogramm fehlschlägt und dass kein erneuter Versuch unternommen wird. Die Dienstprogramme melden einen allgemeinen SQLCODE-Wert -2029, dessen Nachrichtentext den vom Benutzer-Exit-Programm oder vom Betriebssystem zurückgegebenen Code anzeigt.

Tabelle 24 zeigt die Codes, die von einem Benutzer-Exit-Programm 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 24. Rückkehrcodes von Benutzer-Exit-Programmen. 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 Benutzer-Exit-Programm oder einer vom Programm verwendeten Softwarefunktion.[b]
20	Fehler bei mindestens einem Parameter, der an das Benutzer-Exit-Programm übergeben wurde. Prüfen Sie, dass das Benutzer-Exit-Programm die angegebenen Parameter korrekt verarbeitet.[b]
24	Das Benutzer-Exit-Programm wurde nicht gefunden. Unter OS/2 kann diese Fehlernachricht auch bedeuten, dass eine Datei, die zur Ausführung einer Wiederherstellungsoperation erforderlich ist, auf den aktuellen Sicherungsdatenträgern nicht gefunden werden konnte.[b]
28	Ein Fehler trat auf, der durch einen Fehler der Eingabe/Ausgabe (E/A) oder durch das Betriebssystem verursacht wird.[b]
32	Das Benutzer-Exit-Programm wurde vom Benutzer beendet.[b]
255	Das Benutzer-Exit-Programm verursachte einen Fehler, da es die Bibliotheksdatei für die ausführbare Datei nicht laden konnte.[c]

Tabelle 24. Rückkehrcodes von Benutzer-Exit-Programmen (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 Benutzer-Exit-Programm auf Abrufanforderungen für dieselbe Protokolldatei weiterhin 4 oder 8 zurückgibt, ist DB2 blockiert. (Dies gilt für aktualisierende Wiederherstellungen oder für Aufrufe an die API **sqlurlog**, die vom Replikationsdienstprogramm verwendet wird.) [b] Aufrufe für Benutzer-Exit-Programme werden fünf Minuten lang zurückgestellt. Während dieser Zeit werden alle Anforderungen ignoriert, einschließlich der Anforderung, die die Fehlerbedingung verursachte. Nach dieser fünfminütigen Zurückstellung wird die nächste Anforderung verarbeitet. Wenn diese Anforderung ohne Fehler verarbeitet wurde, wird die Verarbeitung neuer Benutzer-Exit-Anforderungen fortgesetzt, und DB2 setzt die Archivierungsanforderung, die fehlgeschlagen oder vorher zurückgestellt worden war, erneut ab. Wenn während der Wiederholung ein Rückkehrcode größer als acht generiert wird, werden Anforderungen für weitere fünf Minuten zurückgestellt. Die fünfminütigen Zurückstellungen 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 Benutzer-Exit-Programm Protokolldateien nicht archivieren kann, füllt sich die Platte möglicherweise mit Protokolldateien an, und die Leistung sinkt. Wenn die Platte voll ist, nimmt der Datenbankmanager keine weiteren Anwendungsanforderungen für Datenbankaktualisierungen mehr entgegen. Wenn das Benutzer-Exit-Programm zum Abrufen von Protokolldateien aufgerufen wurde, wird die aktualisierende Wiederherstellung zurückgestellt, 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 wiederaufnehmen. [c] Wenn das Benutzer-Exit-Programm den Fehlercode 255 zurückgibt, kann das Programm die Bibliotheksdatei für die ausführbare Datei wahrscheinlich nicht laden. Rufen Sie das Benutzer-Exit-Programm manuell auf, um dies zu prüfen. Weitere Informationen werden angezeigt. **Anmerkung:** Während Archivierungs- und Abrufaktionen wird ein Alert für alle Rückkehrcodes außer 0, 4 und 24 ausgegeben. Der Alert enthält den Rückkehrcode vom Benutzer-Exit-Programm und eine Kopie der Eingabeparameter, die an das Benutzer-Exit-Programm übergeben wurden.	

Hinweise zum Sichern und Wiederherstellen (OS/2)

Anhang I. APIs zum Sichern und Wiederherstellen für Produkte anderer Lieferanten

DB2 bietet Schnittstellen für Produkte zur Datenträgerverwaltung von Fremdanbietern, die Sie in Sicherungs- und Wiederherstellungsoperationen zum Speichern und Abrufen von Daten verwenden können. Diese Einrichtung dient zur Erweiterung der Sicherungs- und Wiederherstellungsdatenziele Diskette, Platte, Band und Tivoli Storage Manager, welche von DB2 unterstützt werden.

Dies Datenträgerverwaltungsprodukte von Fremdanbietern werden in diesem Anhang als Produkte anderer Lieferanten bezeichnet.

DB2 definiert einen Satz von Funktionsprototypen, der eine für allgemeine Zwecke verwendbare Datenschnittstelle zur Sicherung und Wiederherstellung 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 OS/2 oder Windows-Betriebssysteme in einer DLL-Datei bereitgestellt werden. Wenn DB2 die Funktionen aufruft, wird die gemeinsam benutzte Bibliothek bzw. die DLL-Datei geladen, die von der aufrufenden Sicherungs- oder Wiederherstellungsroutine angegeben wird. Die Funktionen des Produkts eines anderen Lieferanten werden zur Ausführung der erforderlichen Tasks aufgerufen.

Dieser Anhang ist in folgende vier Teile untergliedert:

- Funktionsübersicht der Interaktion von DB2 mit Produkten anderer Lieferanten
- Detaillierte Beschreibungen der DB2-APIs für Produkte anderer Lieferanten
- Informationen zu den Datenstrukturen, die in den API-Aufrufen verwendet werden
- Details zum Aufruf der Sicherung und Wiederherstellung unter Verwendung von Produkten anderer Lieferanten

Funktionsübersicht

Als Schnittstelle zwischen DB2 und Produkten anderer Lieferanten sind die folgenden fünf 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

DB2 ruft diese Funktionen auf, die vom Produkt eines anderen Lieferanten für UNIX-Systeme in einer gemeinsam benutzten Bibliothek und für OS/2 oder Windows-Bertriebssysteme in einer DLL-Datei bereitgestellt werden müssen.

Anmerkung: Der Code der gemeinsam benutzten Bibliothek oder der DLL-Datei 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
- Art der Operation (Sicherung oder Wiederherstellung)
- PROMPTING-Modus, der für die Sicherungs- und Wiederherstellungsoperation angegeben ist
- Kenndaten der Einheit, auf der die Daten gespeichert sind
- Fehler, die im Betrieb auftreten können

Anzahl der 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, welche von **sqluvint** verwendet wird (siehe „sqluvint - Initialisieren und Verbindung zu Einheit herstellen” auf Seite 513).

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, kann DB2 die Initialisierung

der Sitzung anfordern. 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 Datenträger-Header-Satz. Dieser Satz enthält Daten, anhand derer DB2 die Sitzung während einer Wiederherstellungsoperation bestimmt. DB2 gibt die einzelnen Sitzungen an, indem an den Namen des Sicherungsimage eine Folgenummer angehängt wird. Die Nummer beginnt mit eins für die erste Sitzung und wird bei jeder Initialisierung einer neuen Sitzung, in der mit einem **sqluvint**-Aufruf eine Sicherungs- oder Wiederherstellungsoperation aufgerufen wird, um eins erhöht. Weitere Informationen finden Sie in „INIT-INPUT" auf Seite 535.

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 Angabe der Sicherung erforderlich sind (siehe „DB2-INFO" auf Seite 529). 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 Angabe 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 Robotersysteme einsetzen. DB2 prüft immer den Datenträger-Header (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 auf der 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 eine intelligente Robotereinheit 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 gesendet werden. Dazu muss die Sicherungs- oder Wiederherstellungsoperation mit aktivierter Funktion PROMPTING 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:

- Wenn der Aufruf **sqluvput** war, markiert DB2 den letzten Puffer, der an die Sitzung gesendet wurde so, dass er erneut gesendet wird. Er wird später an die Sitzung gesendet.
- 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 aus der Sitzung zu lesen oder in sie zu schreiben. Zur eindeutigen Angabe der Sitzung beim Schreiben erhöht DB2 die Folgenummer. Die Folgenummer befindet sich in der Struktur DB2-INFO des Aufrufs **sqluvint** im Datenträger-Header-Satz. 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 Benuzter 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 Fehlerbedingungen an DB2" auf Seite 509.

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 Medien 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 Medien wiederhergestellte Puffer an DB2 zurückzugeben. Andernfalls schlägt die Wiederherstellungsoperation fehl.

Bei Rückgabe von Fehlerbedingungen 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 Aufrufs **sqluvend** (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. Der Aufruf **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 auf die folgenden drei Arten antworten: fortsetzen, Einheit beenden oder beenden:

- Will der Benutzer die Aktion *fortsetzen*, versucht DB2, den Puffer mit dem Aufruf **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.

Datei des Wiederherstellungsprotokolls

Die Datei des Wiederherstellungsprotokolls können Sie bei Datenbankwiederherstellungen zur Hilfe nehmen. Diese Datei ist jeder Datenbank zugeordnet und wird bei jeder Sicherungs- oder Wiederherstellungsoperation automatisch aktualisiert. Weitere Informationen zur Datei des Wiederherstellungsprotokolls finden Sie in „Datei des Wiederherstellungsprotokolls" auf Seite 58. 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 in „Datenstruktur: db2HistData" auf Seite 403.

Wenn eine Sicherungsoperation beendet wird, wird mindestens ein Datensatz in die Datei geschrieben. Wurde die Ausgabe der Sicherungsoperation auf Einheiten anderer Lieferanten umgeleitet, enthält das Feld DEVICE im Protokollsatz den Wert 0, und das Feld LOCATION kann einen der folgenden beiden 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 540.

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 Sicherungsimage verwendet wurden und der Datenträger physisch an eine andere Speicherposition (möglicherweise an einen anderen Standort) versetzt wurde. Wenn dies zutrifft, können Sie anhand der Datei des Wiederherstellungsprotokolls ein Sicherungsimage suchen, sobald eine Wiederherstellungsoperation erforderlich wird.

Funktionen und Datenstrukturen

In den folgenden Abschnitten erhalten Sie eine Beschreibung der generischen Funktionen und Datenstrukturen, die Sie in Produkten anderer Lieferanten verwenden können:

APIs für Produkte anderer Lieferanten:

- „sqluvint - Initialisieren und Verbindung zu Einheit herstellen" auf Seite 513
- „sqluvget - Daten von Einheit lesen" auf Seite 518
- „sqluvput - Daten auf Einheit schreiben" auf Seite 521

- „sqluvend - Verbindung zu Einheit aufheben und Ressourcen freigeben" auf Seite 524
- „sqluvdel - Festgeschriebene Sitzung löschen" auf Seite 527

Von Lieferanten-APIs verwendete Datenstrukturen:

„DB2-INFO" auf Seite 529
Enthält Informationen, die DB2 gegenüber der Lieferanteneinheit kennzeichnen.

„VENDOR-INFO" auf Seite 533
Enthält Informationen, die den Lieferanten und die Version der Einheit angeben.

„INIT-INPUT" auf Seite 535
Definiert eine logische Verbindung zwischen DB2 und der Einheit des anderen Lieferanten.

„INIT-OUTPUT" auf Seite 537
Enthält die Ausgabe aus der Einheit.

„DATA" auf Seite 538
Enthält Daten, die zwischen DB2 und der Einheit des anderen Lieferanten übertragen wurden.

„RETURN-CODE" auf Seite 539
Enthält einen Rückkehrcode und eine Erläuterung des Fehlers.

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-Include-Datei

sql.h

C-API-Syntax

```
/* 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 einem Rückkehrcode aus. Wenn der Rückkehrcode einen Fehler angibt, beendet DB2 möglicherweise die Operation mit dem Aufruf der Funktion **sqluvend**. Einzel-

angaben zu den möglichen Rückkehrcodes und zum jeweiligen Verhalten von DB2 finden Sie in der Tabelle mit Rückkehrcodes (siehe Tabelle 25 auf Seite 515).

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 Sicherungsimage 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 (siehe „PROMPTING-Modus" auf Seite 506 und „Kenndaten von Einheiten" auf Seite 507).

DB2 gibt die zu beschreibende Sicherungskopie oder die zu lesende Wiederherstellungskopie über Felder in der Struktur DB2-INFO an. Wurde action = SQLUV_READ angegeben, muss das Produkt eines anderen Lieferanten auf das Vorhandensein des benannten 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 25. Gültige Rückkehrcodes für sqluvint und resultierende DB2-Aktion

Literal in Header-Datei	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 benannten 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 25. Gültige Rückkehrcodes für sqluvint und resultierende DB2-Aktion (Forts.)

Literal in Header-Datei	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 benannten 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.

Tabelle 25. Gültige Rückkehrcodes für sqluvint und resultierende DB2-Aktion (Forts.)

Literal in Header-Datei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
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-Include-Datei

sqluvend.h

C-API-Syntax

```
/* 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 dem 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 26. Gültige Rückkehrcodes für sqluvget und resultierende DB2-Aktion

Literal in Header-Datei	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	Weitere Informationen zur DB2-Bearbeitung von Warnungen finden Sie in „Warnungsbedingungen" auf Seite 510.
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	Weitere Informationen zur DB2-Bearbeitung von Bedingungen für das Datenträgerende finden Sie in „PROMPTING-Modus" auf Seite 506 und „Kenndaten von Einheiten" auf Seite 507.

Tabelle 26. Gültige Rückkehrcodes für sqluvget und resultierende DB2-Aktion (Forts.)

Literal in Header-Datei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
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 (siehe „PROMPTING-Modus" auf Seite 506 und „Kenndaten von Einheiten" auf Seite 507).
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-Include-Datei

sqluvend.h

C-API-Syntax

```
/* 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 dem 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 27. Gültige Rückkehrcodes für sqluvput und resultierende DB2-Aktion

Literal in Header-Datei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
SQLUV_OK	Operation erfolgreich	sqluvput oder sqluvend, sofern beendet (z. B. verfügt DB2 über keine Daten mehr)	Informieren anderer Prozesse über eine beendete Operation
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	Weitere Informationen zur DB2-Bearbeitung von Bedingungen für das Datenträgerende finden Sie in „PROMPTING-Modus" auf Seite 506 und „Kenndaten von Einheiten" auf Seite 507.
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	Weitere Informationen zur DB2-Bearbeitung von Warnungen finden Sie in „Warnungsbedingungen" auf Seite 510.
SQLUV_LINK_NOT_EXIST	Momentan ist keine Programmverbindung vorhanden.	sqluvend, action = SQLU_ABORT[a]	Die Sitzung wird beendet.

Tabelle 27. Gültige Rückkehrcodes für sqluvput und resultierende DB2-Aktion (Forts.)

Literal in Header-Datei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
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 (siehe „Bei Rückgabe von Fehlerbedingungen an DB2" auf Seite 509).			

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-Include-Datei

sql.h

C-API-Syntax

```
/* 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

aktion 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 eine Aktion festgeschrieben werden muss. Die Daten werden für eine Sicherung freigegeben, sofern die Aktion abbricht.

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 mit der gesamten Sicherungs- und Wiederherstellungsoperation Probleme auftreten. Alle aktiven Sitzungen werden von **sqluvend**-Aufrufen mit action = SQLUV_ABORT beendet. Bei einer Sicherungsoperation empfangen festgeschriebene Sitzungen die Aufruffolge **sqluvint**, **sqluvdel** und **sqluvend** (siehe „Bei Rückgabe von Fehlerbedingungen an DB2" auf Seite 509).

- 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 Sitzung werden von **sqluvend**-Aufrufen mit action = SQLUV_ABORT beendet, festgeschriebene Sitzungen empfangen die Aufruffolge **sqluvint**, **sqluvdel** und **sqluvend** (siehe „Bei Rückgabe von Fehlerbedingungen an DB2" auf Seite 509).

  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 igno-

riert. Damit DB2 in diesem Fall den Aufruf von **sqluvend** mit action = SQLUV_ABORT veranlasst, muss ein erster schwer wiegender Fehler eingetreten sein.

Rückkehrcodes

Tabelle 28. Gültige Rückkehrcodes für sqluvend und resultierende DB2-Aktion

Literal in Header-Datei	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-Include-Datei

sqluvend.h

C-API-Syntax

```
/* 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 29. Gültige Rückkehrcodes für sqluvdel und resultierende DB2-Aktion

Literal in Header-Datei	Beschreibung	Möglicher nächster Aufruf	Sonstige Kommentare
SQLUV_OK	Operation erfolgreich	Keine weiteren Aufrufe	
SQLUV_DELETE_FAILED	Löschanforderung ist fehlgeschlagen	Keine weiteren Aufrufe	

DB2-INFO

Diese Struktur enthält Daten, die DB2 gegenüber der Einheit des anderen Lieferanten definieren.

Tabelle 30. 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.

Tabelle 30. Felder in der Struktur DB2-INFO (Forts.). Alle Felder sind auf Null endende Zeichenfolgen.

Feldname	Datentyp	Beschreibung

type	char	Gibt den Typ der Sicherung, die gerade ausgeführt wird, oder den Typ der Wiederherstellung, die gerade stattfindet, an. Mögliche Werte:
		
		Wenn die Aktion SQLUV_WRITE ist:
		`0 - Datenbankgesamtsicherung`
		`3 - Sicherung auf Tabellenbereichsebene`
		
		Wenn die Aktion SQLUV_READ ist:
		`0 - Vollständige Wiederherstellung`
		`3 - Onlinewiederherstellung eines Tabellenbereichs`
		`4 - Wiederherstellung eines Tabellenbereichs`
		`5 - Wiederherstellung einer Protokolldatei`

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	Listet die Objekte im Sicherungsimage auf. Dies ist nur für Informationen von Lieferanten gedacht.
max_bytes_per_txn	sqlint32	Gibt für das Produkt eines anderen Lieferanten die Übertragungspuffergröße, die der Benutzer angab, in Bytes an.
image_filename	char	Zur zukünftigen Verwendung reserviert.
reserve	void	Zur zukünftigen Verwendung reserviert.

Tabelle 30. Felder in der Struktur DB2-INFO (Forts.). Alle Felder sind auf Null endende Zeichenfolgen.

Feldname	Datentyp	Beschreibung
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 in der Angabe *sequence* gibt die Dateierweiterung an. Wenn Sie ein Sicherungsimage wiederherstellen müssen, müssen Sie zum Abruf des Sicherungsimage 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 31. 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 dies 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 32. 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., es wird keine Zwischenstufe 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 33. 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 34. 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 gesendet oder empfangen wurde. 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 35. 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[60],
  void      *reserve,
} Return_code;
```

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
- Anwendungsprogrammierschnittstelle

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-Datei (Windows-Betriebssystem oder OS/2) an.
Anzahl der Sitzungen	*Sitzungen*
Lieferantenoptionen	Nicht unterstützt
Lieferantendateiname	Nicht unterstützt
Größe des Übertragungspuffers	*Puffergröße* (für Sicherung), und nicht zutreffend (für Wiederherstellung).

Befehlszeilenprozessor

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*

Anwendungsprogrammierschnittstelle

Zwei API-Funktionsaufrufe (Application Programming Interface - Anwendungsprogrammierschnittstelle) unterstützen Sicherungs- und Wiederherstellungsoperationen: **sqlubkp** für die Sicherung (siehe „API zur Datenbanksicherung" auf Seite 109) und **sqlurestore** für die Wiederherstellung (siehe „API zur Wiederherstellung von Datenbanken" auf Seite 140).

Angabe von	**API-Parameter (für sqlubkp und sqlurestore)**
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-Datei 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*

Aufrufen einer Sicherung

Anhang J. Verwenden der DB2-Bibliothek

Die Bibliothek für DB2 Universal Database besteht aus Online-Hilfe, Handbüchern (PDF und HTML) und Beispielprogrammen in HTML-Format. Im Folgenden wird beschrieben, welche Informationen bereitgestellt werden und wie Sie darauf zugreifen können.

Über **Information - Unterstützung** können Sie online auf die Produktinformationen zugreifen. Weitere Informationen finden Sie in „Zugreifen auf Informationen mit ″Information - Unterstützung″“ auf Seite 562. Sie können sich im Web Informationen zu Tasks und zur Fehlerbehebung sowie DB2-Bücher, Beispielprogramme und DB2-Informationen anzeigen lassen.

PDF-Dateien und gedruckte Bücher für DB2

Informationen zu DB2

In der folgenden Tabelle sind die DB2-Handbücher in vier Kategorien unterteilt:

DB2-Benutzerhandbücher und -Referenzinformationen
Diese Bücher enthalten die allgemeinen DB2-Informationen für alle Plattformen.

DB2-Installations- und -Konfigurationsinformationen
Diese Bücher gelten für DB2 auf einer bestimmten Plattform. So steht beispielsweise jeweils ein separates Handbuch *Einstieg* (Quick Beginnings) für DB2 auf OS/2-, Windows- und UNIX-Plattformen zur Verfügung.

Plattformübergreifende Beispielprogramme in HTML
Bei diesen Beispielen handelt es sich um die HTML-Versionen der mit Application Development Client installierten Beispielprogramme. Sie dienen zur Information und können die Programme selbst nicht ersetzen.

Release-Informationen
Diese Dateien enthalten die neuesten Informationen, die in die DB2-Handbücher nicht mehr aufgenommen werden konnten.

Die Installationshandbücher, Release-Informationen und Lernprogramme können im HTML-Format direkt von der Produkt-CD-ROM angezeigt werden. Die meisten Handbücher stehen auf der Produkt-CD-ROM im HTML-Format zur Verfügung und können angezeigt werden. Auf der CD-ROM mit DB2-Veröffentlichungen stehen die Handbücher im PDF-Format zur Verfügung und

können mit Adobe Acrobat angezeigt und gedruckt werden. Darüber hinaus können Sie gedruckte Veröffentlichungen bei IBM bestellen. Siehe hierzu „Bestellen der gedruckten Handbücher" auf Seite 557. Die folgende Tabelle enthält eine Liste der Bücher, die bestellt werden können.

Auf OS/2- und Windows-Plattformen können Sie die HTML-Dateien im Verzeichnis `sqllib\doc\html` installieren. Die DB2-Informationen werden in verschiedene Sprachen übersetzt, jedoch nicht alle Informationen in alle Sprachen. Sind bestimmte Informationen in einer Sprache nicht verfügbar, wird stattdessen die englische Version dieser Informationen zur Verfügung gestellt.

Auf UNIX-Plattformen können Sie die HTML-Dateien in mehreren Sprachen installieren, und zwar in den Unterverzeichnissen `doc/%L/html`, wobei *%L* für den Code der jeweiligen Landessprache steht. Weitere Informationen finden Sie im entsprechenden Handbuch *Einstieg*.

Es gibt verschiedene Möglichkeiten, auf DB2-Bücher und -Informationen zuzugreifen:

- „Anzeigen von Online-Informationen" auf Seite 561
- „Suchen nach Online-Informationen" auf Seite 566
- „Bestellen der gedruckten Handbücher" auf Seite 557
- „Drucken der PDF-Handbücher" auf Seite 556

Tabelle 36. Informationen zu DB2

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
DB2-Benutzerhandbücher und -Referenzinformationen			
Systemverwaltung	*Systemverwaltung: Konzept*. Dieses Handbuch enthält eine Übersicht über Datenbankkonzepte, Informationen zu Aspekten des Datenbankentwurfs (wie z. B. zum logischen und physischen Datenbankentwurf) sowie eine Erläuterung zur hohen Verfügbarkeit. *Systemverwaltung: Implementierung*. Dieses Handbuch enthält Informationen zu Implementierungsaspekten, wie beispielsweise zur Implementierung des Datenbankentwurfs, zum Zugriff auf Datenbanken sowie zu Prüfungs-, Sicherungs- und Wiederherstellungsverfahren. *Systemverwaltung: Optimierung*. Dieses Handbuch enthält Informationen zur Datenbankumgebung sowie zur Auswertung und Optimierung der Anwendungsleistung. Sie können die drei Bände des Handbuchs *Systemverwaltung* in englischer Sprache in den USA und Kanada über die Formnummer SBOF-8934 bestellen.	SC12-2879 db2d1g70 SC12-2877 db2d2g70 SC12-2878 db2d3g70	db2d0
Administrative API Reference	Dieses Handbuch enthält eine Beschreibung zu den DB2-Anwendungsprogrammierschnittstellen (APIs) und -Datenstrukturen, die Sie zum Verwalten Ihrer Datenbank verwenden können. Darüber hinaus wird in diesem Handbuch erläutert, wie Sie APIs von Ihren Anwendungen aus aufrufen können.	SC09-2947 db2b0e70	db2b0
Application Building Guide	Dieses Handbuch umfasst Informationen zur Umgebungskonfiguration sowie Anweisungsschritte zum Kompilieren, Verbinden und Ausführen von DB2-Anwendungen auf Windows-, OS/2- und UNIX-Plattformen.	SC09-2948 db2axe70	db2ax

Tabelle 36. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
APPC, CPI-C, and SNA Sense Codes	Dieses Handbuch enthält Basisinformationen zu APPC-, CPI-DFV- und SNA-Prüfcodes, die bei der Arbeit mit DB2 Universal Database-Produkten ausgegeben werden können. Nur im HTML-Format verfügbar.	Keine Formnummer db2ape70	db2ap
Application Development Guide	Dieses Handbuch enthält eine Erläuterung zur Entwicklung von Anwendungen, die mit Hilfe von eingebettetem SQL bzw. JAVA (JDBC und SQLJ) auf DB2-Datenbanken zugreifen. Unter anderem wird das Schreiben von gespeicherten Prozeduren, das Schreiben von benutzerdefinierten Funktionen, das Erstellen von benutzerdefinierten Typen, das Verwenden von Auslösern und das Entwickeln von Anwendungen in partitionierten Umgebungen oder mit Systemen zusammengeschlossener Datenbanken beschrieben.	SC09-2949 db2a0e70	db2a0
CLI Guide and Reference	Dieses Handbuch erklärt die Entwicklung von Anwendungen, die für den Zugriff auf DB2-Datenbanken DB2 Call Level Interface verwenden, eine aufrufbare SQL-Schnittstelle, die mit der Microsoft-ODBC-Spezifikation kompatibel ist.	SC09-2950 db2l0e70	db2l0
Command Reference	Dieses Handbuch enthält eine Erläuterung zur Verwendung des Befehlszeilenprozessors und eine Beschreibung der DB2-Befehle für die Datenbankverwaltung.	SC09-2951 db2n0e70	db2n0

Tabelle 36. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Konnektivität Ergänzung	Dieses Handbuch enthält Konfigurations- und Referenzinformationen zur Verwendung von DB2 für AS/400, DB2 für OS/390, DB2 für MVS oder DB2 für VM als DRDA-Anwendungs-Requester mit DB2 Universal Database-Servern. Darüber hinaus enthält dieses Handbuch Informationen zur Verwendung von DRDA-Anwendungs-Servern mit DB2 Connect-Anwendungs-Requestern. Dieses Buch ist lediglich im HTML- und PDF-Format verfügbar.	Keine Formnummer db2h1g70	db2h1
Versetzen von Daten Dienstprogramme und Referenz	Dieses Handbuch enthält eine Erläuterung zur Verwendung der DB2-Dienstprogramme, wie beispielsweise IMPORT, EXPORT, LOAD, AUTOLOADER und DPROP, die das Verschieben von Daten vereinfachen.	SC12-2881 db2dmg70	db2dm
Data Warehouse-Zentrale Verwaltung	Dieses Handbuch enthält Informationen zur Erstellung und Verwaltung eines Data Warehouse mit Hilfe der Data Warehouse-Zentrale.	SC12-2885 db2ddg70	db2dd
Data Warehouse Center Application Integration Guide	Dieses Handbuch enthält Informationen, die Programmierer bei der Integration von Anwendungen in die Data Warehouse-Zentrale sowie in den Information Catalog Manager unterstützen.	SC26-9994 db2ade70	db2ad
DB2 Connect Benutzerhandbuch	Dieses Handbuch enthält eine Beschreibung der Konzepte der DB2 Connect-Produkte, allgemeine Informationen zur Verwendung sowie Informationen zur Programmierung dieser Produkte.	SC12-2880 db2c0g70	db2c0
DB2 Query Patroller Administration Guide	Dieses Handbuch enthält eine Übersicht über den Betrieb des DB2 Query Patroller-Systems, spezifische Informationen zum Systembetrieb und zur Verwaltung sowie Task-Informationen zu den GUI-Verwaltungsdienstprogrammen.	SC09-2958 db2dwe70	db2dw
DB2 Query Patroller User's Guide	In diesem Handbuch wird die Verwendung der Tools und Funktionen von DB2 Query Patroller beschrieben.	SC09-2960 db2wwe70	db2ww

Tabelle 36. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Glossar	Dieses Handbuch enthält Definitionen zu den in DB2 und den zugehörigen Komponenten verwendeten Begriffen. Es ist im Handbuch *SQL Reference* enthalten und steht außerdem separat im HTML-Format zur Verfügung.	Keine Formnummer db2t0g70	db2t0
DB2 UDB Image, Audio und Video Extender Verwaltung und Programmierung	Dieses Handbuch enthält Basisinformationen zu DB2 Extender, Informationen zur Verwaltung und Konfiguration von IAV Extender sowie Informationen zur Programmierung mit Hilfe von IAV Extender. Es enthält Referenzinformationen, Diagnoseinformationen (mit Nachrichten) und Beispiele.	SC12-2892 dmbu7g70	dmbu7
Information Catalog Manager Systemverwaltung	Dieses Handbuch enthält eine Anleitung zur Verwaltung von Informationskatalogen.	SC12-2886 db2dig70	db2di
Information Catalog Manager Programming Guide and Reference	Dieses Handbuch enthält Definitionen für die Architekturschnittstellen für Information Catalog Manager.	SC26-9997 db2bie70	db2bi
Information Catalog Manager Benutzerhandbuch	Dieses Handbuch enthält Informationen zur Verwendung der Information Catalog Manager-Benutzerschnittstelle.	SC12-2887 db2aig70	db2ai
Installation und Konfiguration Ergänzung	Dieses Handbuch enthält Anweisungen zur Planung, Installation und Konfiguration von plattformspezifischen DB2-Clients. Darüber hinaus enthält es Informationen zu Bindevorgängen, zum Einrichten der Client/Server-Kommunikation, zu DB2-GUI-Tools, zu DRDR-AS, zur verteilten Installation, zur Konfiguration von verteilten Anforderungen sowie zum Zugriff auf heterogene Datenquellen.	GC12-2864 db2iyg70	db2iy

Tabelle 36. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Fehlernachrichten	Dieses Handbuch enthält eine Liste der Nachrichten und Codes, die von DB2, vom Information Catalog Manager und von der Data Warehouse-Zentrale ausgegeben werden, sowie eine Beschreibung der jeweils erforderlichen Benutzeraktionen. Sie können beide Bände des Handbuchs *Fehlernachrichten* in englischer Sprache in den USA und Kanada unter der Formnummer SBOF-8932 bestellen.	Band 1 GC12-2875 db2m1g70 Band 2 GC12-2888 db2m2g70	db2m0
OLAP Integration Server Administration Guide	Dieses Handbuch enthält eine Erläuterung zur Verwendung der Komponente Administration Manager von OLAP Integration Server.	SC27-0782 db2dpe70	n/v
OLAP Integration Server Metaoutline User's Guide	Dieses Handbuch enthält eine Erläuterung zum Erstellen und Ausfüllen von OLAP-Metastrukturen mit Hilfe der OLAP Metaoutline-Standardschnittstelle (nicht mit Hilfe des OLAP Metaoutline Assistant).	SC27-0784 db2upe70	n/v
OLAP Integration Server Model User's Guide	Diese Handbuch enthält eine Erläuterung zum Erstellen von OLAP-Modellen mit Hilfe der OLAP Model-Standardschnittstelle (nicht mit Hilfe des OLAP Model Assistant).	SC27-0783 db2lpe70	n/v
OLAP Konfiguration und Benutzerhandbuch	Dieses Handbuch enthält Informationen zur Konfiguration und Einrichtung von OLAP Starter Kit.	SC12-2889 db2ipg70	db2ip
OLAP Tabellenkalkulations-Add-In Benutzerhandbuch für Excel	Dieses Handbuch enthält eine Beschreibung zur Verwendung des Tabellenkalkulationsprogramms Excel zum Analysieren von OLAP-Daten.	SC12-2890 db2epg70	db2ep
OLAP Tabellenkalkulations-Add-In Benutzerhandbuch für Lotus 1-2-3	Dieses Handbuch enthält eine Beschreibung zur Verwendung des Tabellenkalkulationsprogramms Lotus 1-2-3 zum Analysieren von OLAP-Daten.	SC12-2891 db2tpg70	db2tp
Replikation Benutzer- und Referenzhandbuch	Dieses Handbuch enthält Informationen zur Planung, Konfiguration, Verwaltung und Verwendung der mit DB2 gelieferten Replikations-Tools.	SC12-2884 db2e0g70	db2e0

Tabelle 36. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Spatial Extender Benutzer- und Referenzhandbuch	Diese Handbuch enthält Informationen zur Installation, Konfiguration, Verwaltung, Programmierung und Fehlerbehebung für den Spatial Extender. Darüber hinaus enthält es zentrale Beschreibungen räumlicher Datenkonzepte sowie spezifische Referenzinformationen (Nachrichten und SQL) für den Spatial Extender.	SC12-2894 db2sbg70	db2sb
SQL Erste Schritte	Dieses Handbuch enthält eine Einführung in die SQL-Konzepte sowie Beispiele für eine Reihe von Konstrukten und Tasks.	SC12-2882 db2y0g70	db2y0
SQL Reference, Band 1 und Band 2	Dieses Handbuch beschreibt die Syntax, die Semantik und die Regeln von SQL. Darüber hinaus enthält das Handbuch Informationen zu Inkompatibilitäten zwischen Release-Ständen, Produkteinschränkungen und Katalogsichten. Sie können beide Bände des Handbuchs *SQL Reference* in englischer Sprache in den USA und Kanada unter der Formnummer SBOF-8933 bestellen.	Band 1 ^ SC09-2974 db2s1e70 Band 2 SC09-2975 db2s2e70	db2s0
System Monitor Guide and Reference	Dieses Handbuch enthält eine Beschreibung zum Sammeln unterschiedlicher Informationen zu Datenbanken und dem Datenbankmanager. In diesem Buch wird erläutert, wie Sie mit Hilfe dieser Informationen einen Einblick in Datenbankaktivitäten erhalten, die Leistung verbessern und Fehlerursachen feststellen können.	SC09-2956 db2f0e70	db2f0
Text Extender Verwaltung und Programmierung	Dieses Handbuch enthält Basisinformationen zu DB2 Extender, Informationen zur Verwaltung und Konfiguration von Text Extender sowie zur Programmierung mit Hilfe von Text Extender. Es bietet Referenzinformationen, Diagnoseinformationen (mit Nachrichten) und Beispiele.	SC12-2893 desu9g70	desu9

Tabelle 36. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Troubleshooting Guide	Dieses Handbuch hilft Ihnen bei der Bestimmung von Fehlerquellen, bei der Fehlerbehebung sowie bei der Verwendung von Diagnose-Tools, wenn Sie den DB2-Kundendienst in Anspruch nehmen.	GC09-2850 db2p0e70	db2p0
Neue Funktionen	Dieses Handbuch enthält eine Beschreibung der neuen Einrichtungen, Funktionen und Erweiterungen in DB2 Universal Database Version 7.	SC12-2883 db2q0g70	db2q0
DB2-Installations- und -Konfigurationsinformationen			
DB2 Connect Enterprise Edition für OS/2 und Windows Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation und Konfiguration für DB2 Connect Enterprise Edition unter OS/2 und 32-Bit-Windows-Betriebssystemen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2863 db2c6g70	db2c6
DB2 Connect Enterprise Edition für UNIX Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation, Konfiguration und Ausführung von Tasks für DB2 Connect Enterprise Edition auf UNIX-Plattformen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2862 db2cyg70	db2cy
DB2 Connect Personal Edition Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation, Konfiguration und Ausführung von Tasks für DB2 Connect Personal Edition unter OS/2 und 32-Bit-Windows-Betriebssystemen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für alle unterstützten Clients.	GC12-2869 db2c1g70	db2c1
DB2 Connect Personal Edition für Linux Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Migration und Konfiguration für DB2 Connect Personal Edition für alle unterstützten Linux-Varianten.	GC12-2865 db2c4g70	db2c4

Tabelle 36. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
DB2 Data Links Manager Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Konfiguration und Ausführung von Tasks für DB2 Data Links Manager unter AIX und 32-Bit-Windows-Betriebssystemen.	GC12-2868 db2z6g70	db2z6
DB2 Enterprise - Extended Edition für UNIX Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation und Konfiguration für DB2 Enterprise - Extended Edition auf UNIX-Plattformen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2867 db2v3g70	db2v3
DB2 Enterprise - Extended Edition für Windows Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation und Konfiguration für DB2 Enterprise - Extended Edition unter 32-Bit-Windows-Betriebssystemen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2866 db2v6g70	db2v6
DB2 für OS/2 Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation und Konfiguration von DB2 Universal Database für das Betriebssystem OS/2. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2870 db2i2g70	db2i2
DB2 für UNIX Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation und Konfiguration von DB2 Universal Database auf UNIX-Plattformen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2872 db2ixg70	db2ix

Tabelle 36. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
DB2 für Windows Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Migration und Konfiguration für DB2 Universal Database unter 32-Bit-Windows-Betriebssystemen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2873 db2i6g70	db2i6
DB2 Personal Edition Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Migration und Konfiguration für DB2 Universal Database Personal Edition unter OS/2 und 32-Bit-Windows-Betriebssystemen.	GC12-2871 db2i1g70	db2i1
DB2 Personal Edition für Linux Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Migration und Konfiguration für DB2 Universal Database Personal Edition für alle unterstützten Linux-Varianten.	GC12-2874 db2i4g70	db2i4
DB2 Query Patroller Installation Guide	Dieses Handbuch enthält Installationsinformationen zu DB2 Query Patroller.	GC09-2959 db2iwe70	db2iw
DB2 Warehouse Manager Installation	Dieses Handbuch enthält Installationsinformationen für Warehouse-Agenten, Warehouse-Umsetzungsprogramme und den Information Catalog Manager.	GC12-2876 db2ide70	db2id
Plattformübergreifende Beispielprogramme in HTML			
Beispielprogramme in HTML	Dieses Handbuch enthält die Beispielprogramme für die Programmiersprachen auf allen von DB2 unterstützten Plattformen im HTML-Format. Die Beispielprogramme werden lediglich zu Informationszwecken zur Verfügung gestellt. Nicht alle Beispiele sind für alle Programmiersprachen verfügbar. Die HTML-Beispiele stehen nur dann zur Verfügung, wenn der DB2 Application Development Client installiert ist. Weitere Informationen zu den Programmen finden Sie im Handbuch *Application Building Guide*.	Keine Formnummer	db2hs

Tabelle 36. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
	Release-Informationen		
DB2 Connect Release-Informationen	Dieses Dokument enthält die neuesten Informationen, die in die DB2 Connect-Handbücher nicht mehr aufgenommen werden konnten.	Siehe Anmerkung 2.	db2cr
DB2 Installationsinformationen	Dieses Dokument enthält die neuesten Informationen zur Installation, die in die DB2-Handbücher nicht mehr aufgenommen werden konnten.	Nur auf der Produkt-CD-ROM verfügbar.	
DB2-Release-Informationen	Dieses Dokument enthält die neuesten Informationen zu allen DB2-Produkten und -Funktionen, die in die DB2-Handbücher nicht mehr aufgenommen werden konnten.	Siehe Anmerkung 2.	db2ir

Anmerkungen:

1. Das Zeichen an der sechsten Stelle des Dateinamens gibt die Landessprache eines Buchs an. So kennzeichnet der Dateiname `db2d0e70` die englische Version des Handbuchs *Systemverwaltung*, der Dateiname `db2d0f70` kennzeichnet die französische Version des Buchs. Folgende Buchstaben werden an der sechsten Stelle des Dateinamens verwendet, um die Landessprache für ein Handbuch anzugeben:

Sprache	Kennung
Brasilianisches Portugiesisch	b
Bulgarisch	u
Dänisch	d
Deutsch	g
Englisch	e
Finnisch	y
Französisch	f
Griechisch	a
Italienisch	i
Japanisch	j
Koreanisch	k
Niederländisch	q
Norwegisch	n
Polnisch	p
Portugiesisch	v
Russisch	r
Schwedisch	s
Slowenisch	l
Spanisch	z
Trad. Chinesisch	t
Tschechisch	x
Türkisch	m
Ungarisch	h
Vereinf. Chinesisch	c

2. Kurzfristig verfügbare Informationen, die in die DB2-Handbücher nicht mehr aufgenommen werden können, sind in den Release-Informationen enthalten, die im HTML-Format und als ASCII-Datei verfügbar sind. Die HTML-Version steht über 'Information - Unterstützung' und auf den Produkt-CD-ROMs zur Verfügung. Gehen Sie wie folgt vor, um die ASCII-Dateien anzuzeigen:
   - Rufen Sie auf UNIX-Plattformen die Datei `Release.Notes` auf. Diese Datei befindet sich im Verzeichnis `DB2DIR/Readme/%L`. Dabei ist `%L` die länderspezifische Angabe und `DB2DIR` eine der folgenden Angaben:
     - `/usr/lpp/db2_07_01` (unter AIX)
     - `/opt/IBMdb2/V7.1` (unter HP-UX, PTX, Solaris und Silicon Graphics IRIX)
     - `/usr/IBMdb2/V7.1` (unter Linux)
   - Rufen Sie auf anderen Plattformen die Datei `RELEASE.TXT` auf. Diese Datei befindet sich in dem Verzeichnis, in dem das Produkt installiert ist. Auf OS/2-Plattformen können Sie auch den Ordner **IBM DB2** und anschließend das Symbol **Release-Informationen** doppelt anklicken.

Drucken der PDF-Handbücher

Wenn Sie eine gedruckte Version der Handbücher bevorzugen, können Sie die PDF-Dateien auf der CD-ROM mit DB2-Veröffentlichungen ausdrucken. Mit Adobe Acrobat Reader können Sie entweder das gesamte Handbuch oder bestimmte Teile des Handbuchs ausdrucken. Die Namen der einzelnen Handbücher in der Bibliothek finden Sie in Tabelle 36 auf Seite 545.

Die neueste Version von Adobe Acrobat Reader finden Sie auf der Adobe-Web-Site unter http://www.adobe.com.

Die PDF-Dateien befinden sich auf der CD-ROM mit DB2-Veröffentlichungen und haben die Dateierweiterung PDF. Führen Sie folgende Schritte aus, um auf die PDF-Dateien zuzugreifen:

1. Legen Sie die CD-ROM mit DB2-Veröffentlichungen in das CD-ROM-Laufwerk ein. Auf UNIX-Plattformen: Hängen Sie die CD-ROM mit den DB2-Veröffentlichungen an. Das Handbuch *Einstieg* enthält Anweisungen zu den Mount-Prozeduren.
2. Starten Sie Acrobat Reader.
3. Öffnen Sie die gewünschte PDF-Datei von einer der folgenden Positionen aus:
   - Auf OS/2- und Windows-Plattformen:

     Verzeichnis *x*`:\doc\`*sprache* . Dabei gibt *x* das CD-ROM-Laufwerk an, *sprache* den zweistelligen Landescode für die verwendete Sprache (z. B. EN für Englisch).

- Auf UNIX-Plattformen:

  Verzeichnis */cdrom*/doc/%*L* auf der CD-ROM. Dabei gibt */cdrom* den Mount-Punkt der CD-ROM an und %*L* den Namen der gewünschten länderspezifischen Angaben.

Sie können die PDF-Dateien auch von der CD-ROM in ein lokales Laufwerk oder ein Netzlaufwerk kopieren und sie von dort aus lesen.

Bestellen der gedruckten Handbücher

Sie können die gedruckten DB2-Handbücher einzeln bestellen. In den USA und Kanada ist es außerdem möglich, mehrere Bücher als Paket unter einer SBOF-Nummer zu bestellen. Setzen Sie sich mit Ihrem IBM Vertragshändler oder Vertriebsbeauftragten in Verbindung, oder bestellen Sie die Handbücher telefonisch bei IBM Direkt unter der Nummer 0180/55 090. Darüber hinaus können Sie die Handbücher über die Web-Seite mit Veröffentlichungen unter http://www.elink.ibmlink.ibm.com/pbl/pbl bestellen.

Es sind zwei Gruppen von Handbüchern verfügbar. Die Gruppe mit der Formnummer SBOF-8935 umfasst Referenzinformationen und Informationen zur Verwendung für DB2 Warehouse Manager. Die Gruppe mit der Formnummer SBOF-8931 umfasst Referenzinformationen und Informationen zur Verwendung für alle anderen DB2 Universal Database-Produkte und -Funktionen. Der Inhalt der SBOF-Gruppen ist in der folgenden Tabelle aufgeführt.

Tabelle 37. Bestellen der gedruckten Handbücher

SBOF-Nummer	In dieser Gruppe enthaltene Handbücher	
SBOF-8931	• Administration Guide: Planning • Administration Guide: Implementation • Administration Guide: Performance • Administrative API Reference • Application Building Guide • Application Development Guide • CLI Guide and Reference • Command Reference • Data Movement Utilities Guide and Reference • Data Warehouse Center Administration Guide • Data Warehouse Center Application Integration Guide • DB2 Connect User's Guide • Installation and Configuration Supplement • Image, Audio, and Video Extenders Administration and Programming • Message Reference, Volumes 1 and 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP Setup and User's Guide • OLAP Spreadsheet Add-in User's Guide for Excel • OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL Getting Started • SQL Reference, Volumes 1 and 2 • System Monitor Guide and Reference • Text Extender Administration and Programming • Troubleshooting Guide • What's New
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager User's Guide • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

DB2-Online-Dokumentation

Zugreifen auf die Online-Hilfefunktion

Die Online-Hilfefunktion ist für alle DB2-Komponenten verfügbar. In der folgenden Tabelle werden die verschiedenen Hilfearten beschrieben.

Hilfearten	Inhalt	Zugriff
Hilfe für Befehl	Erklärt die Syntax von Befehlen im Befehlszeilenprozessor.	Geben Sie im interaktiven Modus des Befehlszeilenprozessors Folgendes ein: `? befehl` Dabei stellt *befehl* 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` lediglich Hilfe für den Befehl CATALOG DATABASE angezeigt wird.
Hilfe für **Client-Konfiguration - Unterstützung** *Hilfe für die Befehlszentrale* *Hilfe für die Steuerzentrale* *Hilfe für die Data Warehouse-Zentrale* *Hilfe für* **Event Analyzer** *Hilfe für Information Catalog Manager* *Hilfe für die Satellitenverwaltungszentrale* *Hilfe für die Prozedurenzentrale*	Erläutert die Tasks, die Sie in einem Fenster oder Notizbuch ausführen können. Die Hilfe umfasst Übersichtsinformationen und unbedingt erforderliche Informationen sowie eine Beschreibung zur Verwendung der Steuerelemente im Fenster oder Notizbuch.	Klicken Sie in einem Fenster oder in einem Notizbuch den Druckknopf **Hilfe** an oder drücken Sie die Taste **F1**.

Hilfearten	Inhalt	Zugriff

Nachrichtenhilfe	Beschreibt die Ursache von Nachrichten sowie die auszuführenden Benutzeraktionen.	Geben Sie im interaktiven Modus des Befehlszeilenprozessors Folgendes ein:
		`? XXXnnnnn`
		Dabei ist *XXXnnnnn* eine gültige Nachrichtenkennung.
		Bei Eingabe von `? SQL30081` wird z. B. die Hilfe zur Nachricht SQL30081 angezeigt.
		Wenn Sie die Nachrichtenhilfe seitenweise anzeigen möchten, geben Sie den folgenden Befehl ein:
		`? XXXnnnnn | more`
		Geben Sie folgenden Befehl ein, um die Nachrichtenhilfe in einer Datei zu speichern:
		`? XXXnnnnn > datei.erw`
		Dabei ist *datei.erw* die Datei, in der Sie die Nachrichtenhilfe speichern möchten.

Hilfe für SQL	Erklärt die Syntax von SQL-Anweisungen.	Geben Sie im interaktiven Modus des Befehlszeilenprozessors Folgendes ein: `help anweisung` Dabei gibt *anweisung* eine SQL-Anweisung an. So kann beispielsweise durch die Eingabe von `help SELECT` die Hilfe zur Anweisung SELECT angezeigt werden. **Anmerkung:** Die Hilfe für SQL ist auf UNIX-Plattformen nicht verfügbar.
SQLSTATE-Hilfe	Erklärt SQLSTATE-Werte und SQL-Klassencodes.	Geben Sie im interaktiven Modus des Befehlszeilenprozessors Folgendes ein: `? sqlstate` oder `? klassencode` Dabei ist *sqlstate* ein gültiger, fünfstelliger SQL-Status, und *klassencode* stellt die ersten zwei Ziffern des SQL-Statuswerts dar. So kann beispielsweise durch die Eingabe von `? 08003` Hilfe für den SQL-Statuswert 08003 angezeigt werden, während mit `? 08` Hilfe für den Klassencode 08 angezeigt wird.

Anzeigen von Online-Informationen

Die zum Lieferumfang dieses Produkts gehörenden Handbücher werden als Softcopy im HTML-Format (HTML - Hypertext Markup Language) bereitgestellt. In einer Softcopy können Sie die Informationen auf einfache Art suchen und anzeigen und über Hypertextverbindungen auf zugehörige Informationen zugreifen. Außerdem wird die gemeinsame Nutzung der Bibliothek in Ihrem gesamten Unternehmen erleichtert.

Sie können die Online-Bücher und Beispielprogramme mit jedem Browser anzeigen, der den Spezifikationen von HTML Version 3.2 entspricht.

Führen Sie die nachfolgend beschriebenen Schritte aus, um Online-Bücher oder Beispielprogramme anzuzeigen:

- Wenn Sie DB2-Verwaltungs-Tools ausführen, verwenden Sie **Information - Unterstützung**.
- Klicken Sie in einem Browser **Datei—>Seite öffnen** an. Die geöffnete Seite enthält eine Übersicht über die DB2-Informationen und Verbindungen (Links) zu diesen Informationen:
  - Öffnen Sie auf UNIX-Plattformen die folgende Seite:

    ```
    INSTHOME/sqllib/doc/%L/html/index.htm
    ```

    Dabei ist *%L* die länderspezifische Angabe.
  - Öffnen Sie auf anderen Plattformen die folgende Seite:

    ```
    sqllib\doc\html\index.htm
    ```

    Der Pfad befindet sich auf dem Laufwerk, auf dem DB2 installiert ist.

    Wenn Sie **Information - Unterstützung** nicht installiert haben, können Sie die Seite öffnen, indem Sie das Symbol **DB2-Informationen** doppelt anklicken. Je nach verwendetem Betriebssystem befindet sich das Symbol im Hauptproduktordner bzw. unter Windows im Menü **Start**.

Installieren des Netscape-Browsers

Wenn Sie nicht bereits einen Web-Browser installiert haben, können Sie Netscape von der im Lieferumfang des Produkts enthaltenen Netscape-CD-ROM aus installieren. Führen Sie folgende Schritte aus, um ausführliche Informationen zur Installation zu erhalten:

1. Legen Sie die Netscape-CD-ROM ein.
2. Nur auf UNIX-Plattformen: Hängen Sie die CD-ROM an. Das Handbuch *Einstieg* enthält Anweisungen zu den Mount-Prozeduren.
3. Installationsanweisungen finden Sie in der Datei `CDNAV`*nn*`.txt`. Dabei ist *nn* die zweistellige Landeskennung. Die Datei befindet sich im Stammverzeichnis der CD-ROM.

Zugreifen auf Informationen mit "Information - Unterstützung"

Information - Unterstützung ermöglicht Ihnen den schnellen Zugriff auf DB2-Produktinformationen. **Information - Unterstützung** ist auf allen Plattformen mit DB2-Verwaltungs-Tools verfügbar.

Sie können 'Information - Unterstützung' öffnen, indem Sie das entsprechende Symbol doppelt anklicken. Abhängig vom verwendeten System befindet sich das Symbol im Hauptproduktordner im Ordner 'Information' bzw. unter Windows im Menü **Start**.

Sie können auf 'Information - Unterstützung' auch zugreifen, indem Sie die Funktionsleiste und das Menü **Hilfe** auf der DB2-Windows-Plattform verwenden.

Unter 'Information - Unterstützung' finden Sie sechs verschiedene Arten von Informationen. Klicken Sie die entsprechende Indexzunge an, um die für diese Informationsart verfügbaren Themen aufzurufen.

Funktionen Die Hauptfunktionen, die Sie mit DB2 ausführen können.

Referenz DB2-Referenzinformationen, wie beispielsweise Schlüsselwörter, Befehle und APIs.

Handbücher DB2-Handbücher.

Fehlerbehebung
Kategorien von Fehlernachrichten sowie die entsprechenden Benutzeraktionen.

Beispielprogramme
Beispielprogramme, die in DB2 Application Development Client enthalten sind. Wenn Sie DB2 Application Development Client nicht installiert haben, wird diese Indexzunge nicht angezeigt.

Web DB2-Informationen im World Wide Web. Sie müssen über Ihr System eine Verbindung zum Web herstellen können, um auf diese Informationen zugreifen zu können.

Wenn Sie einen Eintrag aus einer der Listen auswählen, startet **Information - Unterstützung** eine Funktion zum Anzeigen der Informationen. Bei der Anzeigefunktion kann es sich abhängig von der ausgewählten Informationsart um die Hilfeanzeige des Systems, einen Editor oder einen Web-Browser handeln.

In 'Information - Unterstützung' steht eine Suchfunktion zur Verfügung, mit der Sie nach einem bestimmten Thema suchen können, ohne in den Listen blättern zu müssen.

Rufen Sie über die Hypertextverbindung in 'Information - Unterstützung' das Suchformular **In DB2-Online-Informationen suchen** auf.

Der HTML-Such-Server wird normalerweise automatisch gestartet. Wenn eine Suche in HTML-Informationen fehlschlägt, müssen Sie möglicherweise mit einer der nachfolgend aufgeführten Methoden den Such-Server starten:

Unter Windows
Klicken Sie **Start** an und wählen Sie **Programme —> IBM DB2 —> Informationen —> HTML-Such-Server starten** aus.

Unter OS/2
Klicken Sie den Ordner **DB2 für OS/2** und anschließend das Symbol für **HTML-Such-Server starten** doppelt an.

Falls andere Probleme bei der Suche in HTML-Informationen auftreten, finden Sie möglicherweise entsprechende Hinweise in den Release-Informationen.

Anmerkung: Die Suchfunktion steht in Linux-, PTX- und Silicon Graphics IRIX-Umgebungen nicht zur Verfügung.

Verwenden der DB2-Assistenten

Assistenten unterstützen Sie bei der Ausführung bestimmter Verwaltungsaufgaben, indem sie Sie Schritt für Schritt durch jede Aufgabe führen. Assistenten stehen über die Steuerzentrale und 'Client-Konfiguration - Unterstützung' zur Verfügung. In der folgenden Tabelle sind die einzelnen Assistenten und deren Verwendungszweck aufgeführt.

Anmerkung: In Umgebungen mit partitionierten Datenbanken sind die Assistenten **Datenbank erstellen**, **Index erstellen**, **Aktualisierung auf mehreren Systemen konfigurieren** und **Leistungskonfiguration** verfügbar.

Assistent	Verwendung	Zugriff
Datenbank hinzufügen	Katalogisieren einer Datenbank auf einer Client-Workstation.	Klicken Sie in **Client-Konfiguration - Unterstützung** die Option **Hinzufügen** an.
Datenbank sichern	Festlegen, Erstellen und Terminieren eines Sicherungsplans.	Klicken Sie in der Steuerzentrale die zu sichernde Datenbank mit der rechten Maustaste an und wählen Sie **Sichern —> Datenbank mit Assistent** aus.
Aktualisierung auf mehreren Systemen konfigurieren	Konfigurieren einer Aktualisierung auf mehreren Systemen, einer verteilten Transaktion oder einer zweiphasigen Festschreibung.	Klicken Sie in der Steuerzentrale den Ordner **Datenbanken** mit der rechten Maustaste an und wählen Sie **Aktualisierung auf mehreren Systemen** aus.

Assistent	Verwendung	Zugriff
Datenbank erstellen	Erstellen einer Datenbank und Ausführen einiger grundlegender Konfigurationsfunktionen.	Klicken Sie in der Steuerzentrale den Ordner **Datenbanken** mit der rechten Maustaste an und wählen Sie **Erstellen —> Datenbank mit Assistent** aus.
Tabelle erstellen	Auswählen eines Basisdatentyps und Erstellen eines Primärschlüssels für die Tabelle.	Klicken Sie in der Steuerzentrale das Symbol **Tabellen** mit der rechten Maustaste an und wählen Sie **Erstellen —> Tabelle mit Assistent** aus.
Tabellenbereich erstellen	Erstellen eines neuen Tabellenbereichs.	Klicken Sie in der Steuerzentrale das Symbol **Tabellenbereiche** mit der rechten Maustaste an und wählen Sie **Erstellen —> Tabellenbereich mit Assistent** aus.
Index erstellen	Hinweise zum Erstellen und Löschen von Indizes für Ihre Abfragen.	Klicken Sie in der Steuerzentrale das Symbol **Index** mit der rechten Maustaste an und wählen Sie **Erstellen —> Index mit Assistent** aus.
Leistungskonfiguration	Optimieren der Leistung einer Datenbank durch Aktualisieren der Konfigurationsparameter, so dass sie den Anforderungen Ihres Unternehmens entsprechen.	Klicken Sie in der Steuerzentrale die Datenbank, die optimiert werden soll, mit der rechten Maustaste an und wählen Sie **Leistung mit Assistent konfigurieren** aus. Klicken Sie in einer Umgebung mit partitionierten Datenbanken in der Sicht für Datenbankpartitionen die erste Datenbankpartition, die optimiert werden soll, mit der rechten Maustaste an und wählen Sie **Leistung mit Assistent konfigurieren** aus.
Datenbank wiederherstellen	Wiederherstellen einer Datenbank nach einem Fehler. Dieser Assistent hilft Ihnen, zu entscheiden, welche Sicherungskopie Sie verwenden und welche Protokolle Sie erneut abarbeiten.	Klicken Sie in der Steuerzentrale die Datenbank, die wiederhergestellt werden soll, mit der rechten Maustaste an und wählen Sie **Wiederherstellen —> Datenbank mit Assistent** aus.

Einrichten eines Dokument-Servers

Die DB2-Informationen werden standardmäßig auf Ihrem lokalen System installiert. Das bedeutet, dass alle Benutzer, die Zugriff auf DB2-Informationen benötigen, dieselben Dateien installieren müssen. Führen Sie folgende Schritte aus, um die DB2-Informationen an einer einzigen Position zu speichern:

1. Kopieren Sie alle Dateien und Unterverzeichnisse aus dem Verzeichnis `\sqllib\doc\html` Ihres lokalen Systems auf einen Web-Server. Jedem Handbuch ist ein Unterverzeichnis zugeordnet, das alle erforderlichen HTML- und GIF-Dateien enthält, aus denen das Handbuch besteht. Stellen Sie sicher, dass die Verzeichnisstruktur erhalten bleibt.
2. Konfigurieren Sie den Web-Server so, dass er die Dateien an der neuen Speicherposition sucht. Informationen hierzu finden Sie im Anhang zu NetQuestion im Handbuch *Installation und Konfiguration Ergänzung*.
3. Wenn Sie die Java-Version von **Information - Unterstützung** verwenden, können Sie eine Basis-URL-Adresse für alle HTML-Dateien angeben. Sie sollten die URL-Adresse für das Bücherverzeichnis verwenden.
4. Wenn Sie die Buchdateien anzeigen können, ist es möglich, bei häufig aufgerufenen Themen Lesezeichen zu setzen. Es empfiehlt sich, folgende Seiten mit einem Lesezeichen zu versehen:
   - Bücherverzeichnis
   - Inhaltsverzeichnis häufig verwendeter Handbücher
   - Themen, auf die häufig verwiesen wird, wie beispielsweise zum Ändern von Tabellen
   - Suchformular

Informationen dazu, wie Sie die DB2 Universal Database-Online-Dokumentationsdateien auf einer zentralen Maschine zur Verfügung stellen können, finden Sie im Anhang zu NetQuestion im Handbuch *Installation und Konfiguration Ergänzung*.

Suchen nach Online-Informationen

Verwenden Sie eine der folgenden Methoden, um nach Informationen in den HTML-Dateien zu suchen:

- Klicken Sie im obersten Rahmen auf **Suchen**. Verwenden Sie das Suchformular, um nach einem bestimmten Thema zu suchen. Diese Funktion steht in Linux-, PIX- oder Silicon Graphics IRIX-Umgebungen nicht zur Verfügung.
- Klicken Sie im obersten Rahmen auf **Index**. Mit Hilfe des Indexes können Sie nach einem bestimmten Thema im Buch suchen.
- Rufen Sie das Inhaltsverzeichnis oder den Index der Hilfe oder des HTML-Buchs auf und verwenden Sie die Suchfunktion des Web-Browsers, um nach einem bestimmten Thema im Buch zu suchen.
- Mit Hilfe der Lesezeichenfunktion des Web-Browsers können Sie schnell zu einem bestimmten Thema zurückkehren.
- Mit Hilfe der Suchfunktion von **Information - Unterstützung** können Sie bestimmte Themen suchen. Weitere Informationen finden Sie in „Zugreifen auf Informationen mit ″Information - Unterstützung″“ auf Seite 562.

Anhang K. 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. Lizenzanfragen sind schriftlich an IBM Europe, Director of Licensing, 92066 Paris La Defense Cedex, France, zu richten. Anfragen an obige Adresse müssen auf englisch formuliert werden.

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
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
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.

COPYRIGHT-LIZENZ:

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.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
IBM System AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational
Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RS/6000
IBM System /370
SP
SQL/DS
SQL/400
System/370
IBM System /390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

Folgende Namen sind in gewissen Ländern Marken oder eingetragene Marken anderer Unternehmen:

Microsoft, Windows und Windows NT sind Marken oder eingetragene Marken von Microsoft Corporation.

Java und alle auf Java basierenden Marken und Logos sowie Solaris sind in gewissen Ländern Marken von Sun Microsystems, Inc.

Tivoli und NetView sind in gewissen Ländern Marken von Tivoli Systems Inc.

UNIX ist eine eingetragene Marke und wird ausschließlich von der X/Open Company Limited lizenziert.

Andere Namen von Unternehmen, Produkten oder Dienstleistungen können Marken anderer Unternehmen sein.

Index

E

F

G

H

I

J

K

L

M

N

T

U

V

W

X

Z

Kontaktaufnahme mit IBM

Bei technischen Problemen lesen Sie bitte die entsprechenden Korrekturmaßnahmen im Handbuch *Troubleshooting Guide*, und führen Sie diese aus, bevor Sie sich mit der IBM Kundenunterstützung in Verbindung setzen. Mit Hilfe dieses Handbuchs können Sie Informationen sammeln, die die DB2-Kundenunterstützung zur Fehlerbehebung verwenden kann.

Wenn Sie weitere Informationen benötigen oder eines der DB2 Universal Database-Produkte bestellen möchten, setzen Sie sich mit einem IBM Ansprechpartner in einer lokalen Geschäftsstelle oder einem IBM Softwarevertriebspartner in Verbindung.

Telefonische Unterstützung erhalten Sie unter der folgenden Nummer:

- Unter 0180 3/313 233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.

Produktinformationen

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3/313 233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0180/55 090 können Sie Handbücher telefonisch bestellen.

http://www.ibm.com/software/data/
Auf den DB2-World Wide Web-Seiten erhalten Sie aktuelle DB2-Informationen wie Neuigkeiten, Produktbeschreibungen, Schulungspläne und vieles mehr.

http://www.ibm.com/software/data/db2/library/
Mit **DB2 Product and Service Technical Library** können Sie auf häufig gestellte Fragen, Berichtigungen, Handbücher und aktuelle technische DB2-Informationen zugreifen.

Anmerkung: Diese Informationen stehen möglicherweise nur auf Englisch zur Verfügung.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
Auf der Website für die Bestellung internationaler Veröffentlichungen (International Publications) finden Sie Informationen zum Bestellverfahren.

http://www.ibm.com/education/certify/
Das **Professional Certification Program** auf der IBM Website stellt Zertifizierungstestinformationen für eine Reihe von IBM Produkten, u. a. auch DB2, zur Verfügung.

ftp.software.ibm.com
Melden Sie sich anonym an. Im Verzeichnis `/ps/products/db2` finden Sie Demoversionen, Berichtigungen, Informationen und Tools zu DB2 und vielen zugehörigen Produkten.

comp.databases.ibm-db2, bit.listserv.db2-l
Über diese Internet-Newsgroups können DB2-Benutzer Ihre Erfahrungen mit den DB2-Produkten austauschen.

Für Compuserve: GO IBMDB2
Geben Sie diesen Befehl ein, um auf IBM DB2 Family-Foren zuzugreifen. Alle DB2-Produkte werden über diese Foren unterstützt.

In Anhang A des Handbuchs *IBM Software Support Handbook* finden Sie Informationen dazu, wie Sie sich mit IBM in Verbindung setzen können. Rufen Sie die folgende Webseite auf, um auf dieses Dokument zuzugreifen: http://www.ibm.com/support/. Wählen Sie anschließend die Verbindung zum IBM Software Support Handbook am unteren Rand der Seite aus.

Anmerkung: In einigen Ländern sollten sich die IBM Vertragshändler an die innerhalb ihrer Händlerstruktur vorgesehene Unterstützung wenden, nicht an die IBM Unterstützungsfunktion.