IBM® DB2 Universal Database™

データ・リカバリーと高可用性 ガイドおよびリファレンス

バージョン 8.2

SC88-9143-01
(英文原典：SC09-4831-01)

IBM® DB2 Universal Database™

データ・リカバリーと高可用性 ガイドおよびリファレンス

バージョン 8.2

SC88-9143-01
(英文原典：SC09-4831-01)

ご注意!

本書および本書で紹介する製品をご使用になる前に、『*特記事項*』 に記載されている情報をお読みください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： SC09–4831–01
IBM® DB2 Universal Database™
Data Recovery and High Availability Guide and Reference
Version 8.2

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.8

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財）日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

本書について

本書では、IBM DB2 Universal Database (UDB) のバックアップ、リストア、およびリカバリー・ユーティリティーに関する詳細と、それぞれの使用方法について説明します。また、高可用性の重要性について説明し、さまざまなプラットフォームでの DB2 フェイルオーバー・サポートについても説明します。

本書の対象読者

このマニュアルは、データベース管理者、アプリケーション・プログラマー、その他 DB2 データベース・システムでバックアップ、リストア、およびリカバリー操作を担当したり理解したいと思っている DB2 UDB ユーザーを対象としています。

読者は DB2 Universal Database、構造化照会言語 (SQL)、および DB2 UDB が稼働するオペレーティング・システム環境に精通しているものと想定されています。本書では、DB2 のインストール方法については説明しません。それはご使用のオペレーティング・システムによって異なります。

本書の構成

以下のトピックについて説明します。

データ・リカバリー

『第 1 章 バックアップおよびリカバリー計画の作成』
データベースおよび表スペースのリカバリー方式 (データベースおよび表スペースのバックアップとリカバリー、およびロールフォワード・リカバリー方式の使用を含む) を選択する場合の考慮事項について説明します。

『第 2 章 データベースのバックアップ』
データベースや表スペースのバックアップ・コピーの作成に使用する、DB2 バックアップ・ユーティリティーについて説明します。

『第 3 章 データベースのリストア』
事前にバックアップを取ったデータベースや表スペースが損傷したり破壊されたりした場合の再作成に使用する、 DB2 リストア・ユーティリティーについて説明します。

『第 4 章 ロールフォワード・リカバリー』
データベース・リカバリー・ログ・ファイルに記録されたトランザクションを適用してデータベースをリカバリーする際に使用する、 DB2 ロールフォワード・ユーティリティーについて説明します。

『第 5 章 データベースのリカバリー』
DB2 リカバリー・ユーティリティーについて説明します。これは、リカバリー履歴ファイルの情報に基づき、必要なリストアおよびロールフォワード操作を実行して、指定した時点までデータベースをリカバリーするものです。

高可用性

『第 6 章 高可用性およびフェイルオーバー・サポートの紹介』
DB2 が提供する高可用性障害リカバリー・サポートの概要を説明します。

『第 7 章 高可用性災害時リカバリー (HADR)』
高可用性災害時リカバリー (HADR) 環境をセットアップして管理するときに必要な概念と手順を説明します。

『第 8 章 AIX でのクラスター・サポート』
AIX での高可用性フェイルオーバー・リカバリーの DB2 サポートについて説明します。このサポートは現在 High Availability Cluster Multi-Processing (HACMP) for AIX の拡張スケーラビリティー (ES) 機能によって実装されています。

『第 9 章 Windows オペレーティング・システムでのクラスター・サポート』
Windows オペレーティング・システムでの高可用性フェイルオーバー・リカバリーの DB2 サポートについて説明します。このサポートは現在 Microsoft Cluster Server (MSCS) によって実装されています。

『第 10 章 Solaris オペレーティング環境でのクラスター・サポート』
Solaris オペレーティング環境での高可用性フェイルオーバー・リカバリーの DB2 サポートについて説明します。このサポートは現在 Sun Cluster 3.0 (SC3.0) または Veritas Cluster Server (VCS) によって実装されています。

付録

『付録 A. 構文図の読み方』
構文図で使用されている表記規則について説明します。

『付録 B. 警告、エラー、および完了メッセージ』
警告またはエラー状態が検出された場合にデータベース・マネージャーが生成するメッセージの解釈に関する情報を提供します。

『付録 C. 追加の DB2 コマンド』
リカバリー関連の DB2 コマンドについて説明します。

『付録 D. 追加の API および関連データ構造』
リカバリー関連の API とそのデータ構造について説明します。

『付録 E. リカバリー・サンプル・プログラム』
リカバリー関連の DB2 API と組み込み SQL 呼び出しを含むサンプル・プログラムのコードのリストと、その使用法に関する情報が記載されています。

『付録 F. Tivoli Storage Manager』
Tivoli Storage Manager (TSM) 製品に関する情報が記載されています。この製品を使用して、データベースや表スペースのバックアップ操作を管理できます。

『付録 G. データベース・リカバリー用のユーザー出口』
ユーザー出口プログラムでデータベース・ログ・ファイルを使用する方法について説明し、いくつかのサンプル・ユーザー出口プログラムを示します。

『付録 H. ベンダー製品用のバックアップおよびリストア API』
DB2 と他のベンダー・ソフトウェアとのインターフェースを確立させるための API の機能および使用方法を説明しています。

第 1 部 データ・リカバリー

第 1 章 バックアップおよびリカバリー計画の作成

このセクションでは、データベースおよび表スペースのリカバリー方式 (データベースおよび表スペースのバックアップとリカバリー、およびロールフォワード・リカバリー方式の使用を含む) を選択する場合の考慮事項について説明します。

以下のトピックについて説明します。

バックアップおよびリカバリー計画の作成

データベースはハードウェア障害またはソフトウェア障害 (あるいはその両方) が原因で使用不能になることがあります。ストレージの問題、電源の停止、またはアプリケーションの障害が同時に起きたり別々に起きたりすることがあり、それぞれの障害で異なったリカバリー処置が必要になります。十分にテストされた適切なリカバリー計画を作成することにより、データが失われる可能性に備えてデータを保護してください。リカバリー計画を作成する際には、以下の要素を考慮する必要があります。

- データベースはリカバリー可能か。
- データベース・リカバリーにどれくらい時間がかかるか。
- バックアップ操作間の間隔。
- バックアップ・コピーおよびアーカイブ・ログのために割り振ることができるストレージ・スペース量。
- 表スペースのレベルのバックアップで十分か、それとも全データベースのバックアップが必要か。
- スタンバイ・システムを、手動で構成するか、高可用性災害時リカバリー (HADR) を使用して構成するか。

データベース・リカバリー計画では、データベース・リカバリーのために必要になった時点ですべての情報を使用できるようにしておく必要があります。データベースのバックアップを取るための定期的なスケジュールを組み込み、パーティション・データベース・システムの場合はシステム規模の変更時 (データベース・パーティション・サーバーまたはノードの追加やドロップによる) のバックアップも組み込む必要があります。またコマンド・スクリプト、アプリケーション、ユーザー定義関数 (UDF)、オペレーティング・システム・ライブラリー中のストアード・プロシージャー・コード、およびロード・コピーのリカバリー手順も計画全体に組み込む必要があります。

以下に、各種のリカバリー方式について説明し、業務環境に最適なリカバリー方式を判別する方法を示します。

データベース・バックアップ の概念は、他のデータ・バックアップの概念と同じです。つまり、オリジナルで障害または損傷が起こる場合のために、データのコピーを取り、異なるメディアに保管します。一番単純なバックアップでは、データベースをシャットダウンして、トランザクションがこれ以上生じないようにしてから、単純にそのバックアップを取ります。その後何らかの原因でデータベースが損傷したり破壊されたりした場合に、そのデータベースを再構築することができます。

このデータベースの再構築のことを*リカバリー* といいます。*バージョン・リカバリー* は、以前のバージョンのデータベースのリカバリーであり、バックアップ操作で作成されたイメージを使用して行われます。*ロールフォワード・リカバリー* では、データベースまたは表スペースのバックアップ・イメージがリストアされた後で、データベース・ログ・ファイル中に記録されているトランザクションが再度適用されます。

クラッシュ・リカバリー では、 1 つまたは複数の作業単位 (トランザクション) の一部となるすべての変更内容が完了しコミットされる前に障害が発生すると、データベースが自動的にリカバリーされます。これは、未完了のトランザクションをロールバックし、故障発生時にメモリーに残っていたコミット済みトランザクションを完了することによって行われます。

データベースを作成すると、ログ・ファイルとリカバリー履歴ファイルが自動的に作成されます (5 ページの図 1)。消失または損傷したデータをリカバリーする必要がある場合には、それらのログ・ファイルは重要になります。

それぞれのデータベースには*リカバリー・ログ* が含まれており、これはアプリケーションまたはシステム・エラーからリカバリーするときに使用します。データベース・バックアップと組み合わせて、これらはデータベースの整合性をエラーが生じた時点までリカバリーするために使用されます。

リカバリー履歴ファイル には、指定した時点までデータベースのすべてまたは一部をリカバリーする必要のある場合に、リカバリー・オプションを判別するために使用できるバックアップ情報のサマリーが含まれています。これは特に、バックアップ操作やリストア操作などのリカバリー関連のイベントを追跡するために使用します。このファイルは、データベース・ディレクトリーにあります。

表スペース変更履歴ファイル (これもデータベース・ディレクトリーにある) は、特定の表スペースのリカバリーにどのログ・ファイルが必要かを判別するために使用できる情報を含んでいます。

リカバリー履歴ファイルや表スペース変更履歴ファイルは直接変更できません。しかし、PRUNE HISTORY コマンドを使用して、ファイルから (履歴) レコードを削除できます。さらに、*rec_his_retentn* データベース構成パラメーターを使用して、これらの履歴ファイルが保存される日数を指定することもできます。

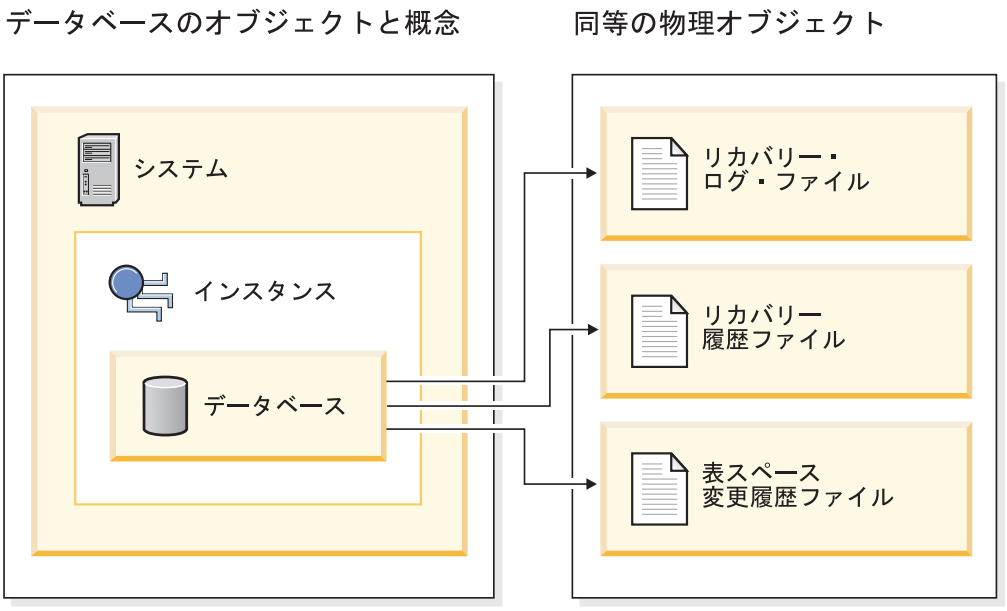


図 1. データベース・リカバリー・ファイル

容易に再作成できるデータは、リカバリー不能データベースに保管できます。これには、読み取り専用アプリケーションに使用される外部ソースからのデータや、頻繁に更新されない表 (ロギングを十分に行っていないため、ログ・ファイルの管理およびリストア操作後のロールフォワードの複雑さに対応できない) が含まれます。*リカバリー不能データベース* では、 *logarchmeth1* および *logarchmeth2* データベース構成パラメーターが「OFF」に設定されます。これは、保管されているログのみがクラッシュ・リカバリーに必要なログであることを意味します。これらのログは、*アクティブ・ログ* と呼ばれ、現在のトランザクション・データを含んでいます。オフライン・バックアップによるバージョン・リカバリーとは、基本的にはリカバリー不能データベースのリカバリーを行うことを意味します。 (オフライン・バックアップは、バックアップ操作の進行中は、他のアプリケーションがこのデータベースを使用できないという意味です。) このようなデータベースは、オフラインでのみリストアできます。リストアすると、バックアップ・イメージが取られたときの状態に戻り、ロールフォワード・リカバリーはサポートさません。

容易に再作成できない データは、リカバリー可能データベースに保管する必要があります。これには、ロード後にソースが破棄されるデータ、表の中に手操作で入力するデータ、およびデータベースにロードした後にアプリケーション・プログラムまたはユーザーによって修正されるデータが含まれます。*リカバリー可能データベース* では、 *logarchmeth1* または *logarchmeth2* データベース構成パラメーターが「OFF」以外の値が設定されたものです。アクティブ・ログをクラッシュ・リカバ

リーで使用できますが、アーカイブ・ログ もあり、これにはコミット済みトランザクション・データが含まれています。このようなデータベースは、オフラインでのみリストアできます。リストアすると、バックアップ・イメージが取られたときの状態に戻ります。ただし、ロールフォワード・リカバリーでは、アクティブ・ログおよびアーカイブ・ログを使用することによって、特定の時点またはアクティブ・ログの最後までデータベースをロール・フォワード する (つまり、バックアップ・イメージが取られたときよりも進める) ことができます。

リカバリー可能データベースのバックアップ操作はオフラインでもオンライン でも実行できます (オンラインとは、バックアップ操作中に他のアプリケーションがそのデータベースに接続できるという意味です)。オンライン表スペースのリストアおよびロールフォワード操作は、データベースがリカバリー可能な場合にのみサポートされます。データベースがリカバリー不能である場合、データベースのリストアおよびロールフォワード操作は、オフラインで実行する必要があります。オンライン・バックアップを作成したときには、ロールフォワード・リカバリーは、そのバックアップがリストアされた後にすべての 表の変更が取り込まれ、再適用されることを保証します。

リカバリー可能データベースがある場合は、データベース全体の代わりに、個々の表スペースをバックアップ、リストア、およびロールフォワードすることができます。表スペースをオンラインでバックアップするとき、その表スペースは依然として使用可能であり、同時に行われる更新がロギングされます。表スペースに対してオンライン・リストアまたはロールフォワード操作を実行するときは、表スペース自体は操作が完了するまで使用できませんが、ユーザーが他の表スペースにアクセスできないということはありません。

自動バックアップ操作

バックアップ操作のような保守活動を実行するかどうか、またいつ実行するかを判別することは時間がかかる場合があるため、「自動保守の構成」ウィザードを使用して代行してもらうことができます。自動保守では、いつ自動保守を実行するかを含む、保守の方針を指定します。 DB2 は、これらの方針を使用して、保守活動を実行する必要があるかどうかを判別し、次回の保守時間枠 (自動保守活動を実行するときのユーザー定義時間枠) 内で必要な保守活動だけを実行します。

注: 自動保守が構成されるときでも、引き続きバックアップ操作を手動で実行できます。 DB2 は、必要な場合にのみ、自動バックアップ操作を実行します。

関連概念:

- 11 ページの『クラッシュ・リカバリー』
- 25 ページの『バージョン・リカバリー』
- 26 ページの『ロールフォワード・リカバリー』
- 201 ページの『高可用性災害時リカバリーの概要』
- 「*DB2 Data Links Manager 管理ガイドおよびリファレンス*」の『データ・リンク・サーバー・ファイルのバックアップ』
- 「*DB2 Data Links Manager 管理ガイドおよびリファレンス*」の『障害とリカバリーの概要』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『rec_his_retentn - 「リカバリー履歴保存期間」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logarchmeth1 - 1 次ログ・アーカイブ方式構成パラメーター』
- 「*DB2 Data Links Manager 管理ガイドおよびリファレンス*」の『DB2 Data Links Manager システムのセットアップとバックアップに関する推奨事項』

バックアップの頻度の決定

データベースのバックアップには時間もシステム・リソースも必要となるため、リカバリー計画では、定期的なバックアップ操作も含める必要があります。全データベースのバックアップと増分バックアップ操作を組み合わせて計画に含めることができます。

ログをアーカイブするとしても (これにより、ロールフォワード・リカバリーが可能になる)、全データベースのバックアップを定期的にとるようにしてください。データベースをリカバリーするには、すべての表スペース・バックアップ・イメージを含む、完全なデータベース・バックアップ・イメージが必要です。表スペースのバックアップ・イメージは、単独のディスク障害やアプリケーション・エラーからリカバリーする場合に有効です。パーティション化されたデータベース環境では、失敗した表スペースをリストアするだけで十分です。すべての表スペースまたはすべてのパーティションをリストアする必要はありません。

また、バックアップ・イメージおよびログを上書きせずに、安全を考慮して全データベースのバックアップ・イメージおよび関連ログを 2 個以上保管することも考慮してください。

更新頻度の高いデータベースのリカバリーとロールフォワードで、アーカイブ・ログを適用するのに必要な時間について心配な場合には、頻繁にデータベースのバックアップをとるためのコストを考慮してください。これにより、ロールフォワード時に適用する必要のあるアーカイブ・ログの数を減らすことができます。

バックアップ操作は、データベースがオンライン でもオフライン でも開始できます。オンラインの場合、他のアプリケーションまたはプロセスは、バックアップ操作の実行中もデータベースに接続したり、データの読み取りや修正を行うことができます。バックアップ操作がオフラインで実行される場合、他のアプリケーションはデータベースに接続できません。

データベースが使用できなくなる時間を短くするには、オンライン・バックアップ操作の使用が考えられます。ロールフォワード・リカバリーが可能である場合にのみ、オンライン・バックアップ操作がサポートされます。ロールフォワード・リカバリーを使用でき、リカバリー・ログの完全セットがある場合は、必要が生じたときにデータベースを再作成できます。バックアップ操作が実行されていた時間に関係しているログがある場合、リカバリーに使用できるのはオンライン・バックアップ・イメージのみです。

オフライン・バックアップ操作は、データ・ファイルの競合がないため、オンライン・バックアップ操作より高速です。

バックアップ・ユーティリティーを使用すると、選択した表スペースのバックアップを取ることができます。 DMS 表スペースを使用すると、その表スペースに異なったタイプのデータを格納でき、バックアップ操作に必要な時間を短縮できます。表データをある表スペースに保持し、長形式フィールドおよび LOB データを別の表スペースに保持し、索引をさらに別の表スペースに保持することができます。このことを行った後にディスク障害が起きた場合には、おそらく 1 つの表スペースしか影響しないでしょう。これらの表スペースの 1 つをリストアまたはロールフォワードするためにかかる時間は、すべてのデータを含む単一の表スペースをリストアするよりも短くてすみます。

さらに、異なる表スペースへの変更が同じものでないなら、それらの表スペースを別の機会にバックアップすることにより時間を節約することもできます。それで、長形式フィールドまたは LOB データが他のデータほど頻繁に変更されない場合には、それらの表スペースのバックアップ頻度を少なくすることができます。長形式フィールドおよび LOB データがリカバリーに必要ではない場合、そのデータを含む表スペースをバックアップしないことも考慮できます。 LOB データが別のソースから再作成可能である場合は、 LOB 列を含む表の作成または変更時には、NOT LOGGED オプションを選択してください。

注: 長形式フィールド・データ、LOB データ、および索引を別々の表スペースに保持し、かつこれらのバックアップを同時に取らない場合には、以下の点を考慮してください。表データの一部が入っていない表スペースをバックアップする場合、その表スペースに対してポイント・イン・タイム指定ロールフォワード・リカバリーは実行できません。表に関する任意のデータ・タイプが含まれているすべての表スペースは、同じ時点まで同時にロールフォワードする必要があります。

表を再編成する場合、操作完了後に関連した表スペースのバックアップを作成する必要があります。これにより、表スペースをリストアしなければならない場合でも、データ再編成によりロールフォワードを実行しなくても済みます。

データベースのリカバリーに必要な時間は、次の 2 つの要素で構成されます。つまり、バックアップのリストアを完了するために必要な時間と、ロールフォワード操作時にログを適用するために必要な時間 (これは、データベースが順方向リカバリーについて使用可能である場合) です。リカバリー計画を公式化するときは、これらのリカバリー費用とそれらが業務操作に与える影響を考慮しなければなりません。全般的なリカバリー計画をテストすることで、データベースをリカバリーするために必要な時間が、業務上の要件を考慮して正当かどうかを判別することができます。各テストを実施した後、バックアップを作成する頻度を増やすことができます。リカバリー計画の一部としてロールフォワード・リカバリーを実行する場合は、これによりバックアップ間でアーカイブされるログ数が減少し、その結果、リストア操作後にデータベースをロールフォワードするための時間が短縮されます。

関連概念:

- 3 ページの『バックアップおよびリカバリー計画の作成』
- 29 ページの『増分バックアップおよびリカバリー』

関連資料:

- 375 ページの『付録 G. データベース・リカバリー用のユーザー出口』

- 40 ページの『データベース・ロギングの構成パラメーター』

リカバリー時のストレージに関する考慮事項

どのリカバリー方式を使用するかを決定する際には、ストレージ・スペースの要件を考慮する必要があります。

バージョン・リカバリー方式の場合は、データベースとリストア後のデータベースを入れるスペースが必要になります。ロールフォワード・リカバリー方式の場合は、データベースまたは表スペースのバックアップ・コピー、リストア後のデータベース、およびアーカイブ・データベース・ログを入れるだけのスペースが必要になります。

表の中に長形式フィールドとラージ・オブジェクト (LOB) の列が含まれている場合は、そのデータを別の表スペースに入れることを考慮してください。このことは、リカバリーの計画に関係するだけでなく、ストレージ・スペースにも影響します。長形式フィールドと LOB データを別の表スペースに取り分けておき、長形式フィールドと LOB データのバックアップにかかる時間が分かっているなら、表スペースのバックアップ頻度を少なくしたリカバリー計画にするよう決定できるかもしれません。表を作成または変更して LOB 列を組み込む際に、これらの列に対する変更内容を記録しないことを選択することもできます。このように選択すると、必要なログ・スペースおよび対応するログ・アーカイブ・スペースのサイズを減らせます。

メディア障害が発生したときにデータベースが破壊され、それを再作成できなくなってしまうことがないようにするため、データベース・バックアップ、データベース・ログ、およびデータベース自身を別々の装置に保持するようにしてください。この理由で、データベース作成時に、データベース・ログは、 *newlogpath* 構成パラメーターを使うことによって、必ず別の装置に保管しておくようにしてください。

データベース・ログは、大量のストレージを使用することがあります。ロールフォワード・リカバリー方式を使用する場合は、アーカイブ・ログをどのように管理するかを決定する必要があります。選択肢は次のとおりです。

- LOGARCHMETH1 または LOGARCHMETH2 構成パラメーターを使用して、ログ・アーカイブ方式を指定する。
- すでにアクティブ・セットではなくなったログを、データベース・ログ・パス・ディレクトリー以外のストレージ・デバイスまたはディレクトリーに手動でコピーする。
- ユーザー出口プログラムを使用して、これらのログを別のストレージ・デバイスにコピーする。

関連概念:

- 53 ページの『ログ・アーカイブを使用したログ・ファイルの管理』

関連資料:

- 40 ページの『データベース・ロギングの構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logarchmeth1 - 1 次ログ・アーカイブ方式構成パラメーター』

- 「*管理ガイド: パフォーマンス*」の『logarchmeth2 - 2 次ログ・アーカイブ方式構成パラメーター』

関連データの一括保持

データベースを設計するとき、各表間の関係が分かります。これらの関係はアプリケーション・レベルで表現することができ、この場合は、トランザクションは複数の表を更新します。またデータベース・レベルで表現することができ、この場合は、表間に参照保全が存在するかまたはある表のトリガーが別の表に影響を与えます。リカバリー計画を作成するときは、これらの関係を考慮に入れる必要があります。関連するデータ・セットは、まとめてバックアップできます。そのようなセットは、表スペース・レベルまたはデータベース・レベルのいずれかで作成できます。関連するデータのセットをまとめて保持することで、すべてのデータの整合性が保証されている時点まで、リカバリー処理を実行できます。これは、表スペースに対し特定の時点のロールフォワード・リカバリーを実行できるようにしたい場合、特に重要です。

異なるオペレーティング・システムの使用

DB2® は、クロスプラットフォームでのバックアップおよびリストア操作をサポートしています。 DB2 バージョン 8、32 ビット Windows® 版で作成されたデータベースを、 DB2 バージョン 8、64 ビット Windows 版にリストアしたり、その逆にリストアしたりすることが可能です。 DB2 バージョン 8、32 ビット Linux (Intel) 版で作成されたデータベースを、 DB2 バージョン 8、64 ビット Linux (Intel) 版にリストアしたり、その逆にリストアしたりすることが可能です。 DB2 バージョン 8、AIX®、HP-UX、または Solaris オペレーティング環境版 (32 ビットまたは 64 ビット) で作成されたデータベースを、 DB2 バージョン 8、 AIX、 HP-UX、または Solaris オペレーティング環境版 (32 ビットまたは 64 ビット) にリストアできます。

さらに、ワード・サイズ (32 ビットまたは 64 ビット) が同じである限り、 DB2 の旧バージョン (2 バージョン前まで) で作成されたバックアップ・イメージをリストアすることもできます。 DB2 の旧バージョンで作成されたバックアップ・イメージのクロスプラットフォーム・リストア操作は、サポートされていません。ターゲット・システムは、ソース・システムとして、同じ (またはより新しい) バージョンの DB2 を持っている必要があります。下位レベルのシステムへのリストア操作は、サポートされていません。

一方のオペレーティング・システムから他のオペレーティング・システムに表を移動しなければならない場合には、 **db2move** コマンドを使用するか、またはエクスポート・ユーティリティーを使用してからインポート・ユーティリティーかロード・ユーティリティーを使用できます。

関連資料:

- 「コマンド・リファレンス」の『db2move - データベース移動ツール・コマンド』
- 「コマンド・リファレンス」の『EXPORT コマンド』

- 「コマンド・リファレンス」の『IMPORT コマンド』
- 「コマンド・リファレンス」の『LOAD コマンド』

クラッシュ・リカバリー

データベースに対するトランザクション (つまり作業単位) は、予期しない割り込みを受けることがあります。たとえば、作業単位の一部となるすべての変更内容が完了しコミットされる前に、障害が発生すると、データベースは矛盾した、または使用不能な状態のままになっています。クラッシュ・リカバリー とは、データベースを整合した使用可能な状態に戻すプロセスのことです。これは、未完了のトランザクションをロールバックし、破損発生時にメモリーに残っていたコミット済みトランザクションを完了することによって行われます (図 2)。データベースが整合性があり使用可能な状態の場合には、これは「整合点」にあることになります。

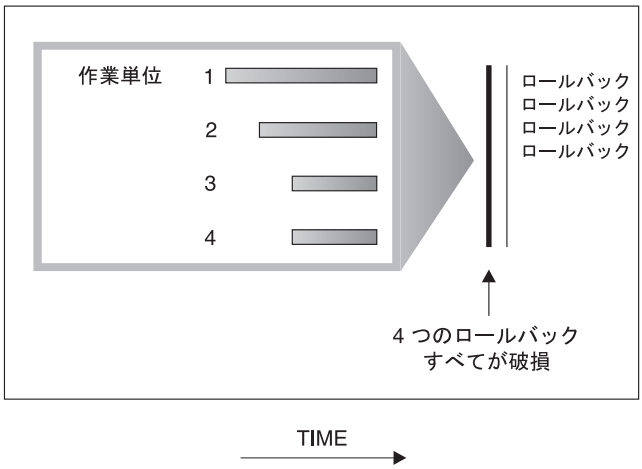


図 2. 作業単位のロールバック (クラッシュ・リカバリー)

トランザクション障害 は、データベースまたはデータベース・マネージャーを異常終了させる重大なエラーまたは状態が起きると発生します。障害が発生した時点で、一部だけ完了している作業単位 (UOW) がディスクにフラッシュされていないと、データベースは矛盾した状態のままになっています。トランザクション障害が発生したら、データベースをリカバリーしなければなりません。以下の条件がトランザクション障害の原因になることがあります。

- マシンの電源障害。そのマシン上のデータベース・マネージャーやデータベース・パーティションがダウンします。
- メモリー破壊、ディスク、CPU、またはネットワーク障害などのハードウェア障害。
- DB2® がダウンするほどの重大なオペレーティング・システム・エラー。

障害発生時に未完了になっている作業単位をデータベース・マネージャーによって自動的にロールバックしたい場合は、自動再始動 (*autorestart*) データベース構成パラメーターを ON に設定し、使用可能にしてください。 (これがデフォルト値です。) 自動再始動を動作したくない場合、 *autorestart* データベース構成パラメーターを「OFF」に設定してください。結果として、データベース障害の発生時に

RESTART DATABASE コマンドを発行する必要があります。破損が発生する前にデータベース入出力が中断された場合には、クラッシュ・リカバリーが継続するように、RESTART DATABASE コマンドの WRITE RESUME オプションを指定する必要があります。データベースが再始動操作を開始すると、管理通知ログに記録されます。

順方向リカバリーが使用可能になっている (つまり *logarchmeth1* 構成パラメーターが OFF にセットされていない) データベースで、クラッシュ・リカバリーを実行する場合に、個別の表スペースが原因でクラッシュ・リカバリー時にエラーが発生すると、その表スペースはオフラインになり、修復されるまでアクセスできなくなります。クラッシュ・リカバリーは続行します。クラッシュ・リカバリーが完了した時点で、データベースに入っている他の表スペースはアクセス可能であり、データベースへの接続は確立できます。しかしながら、オフラインになった表スペースがシステム・カタログを含む表スペースである場合には、その表スペースは、いずれかの接続が許可される前に修復されなければなりません。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『autorestart - 「自動再始動使用可能」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logarchmeth1 - 1 次ログ・アーカイブ方式構成パラメーター』

クラッシュ・リカバリー - 詳細

損傷を受けた表スペースのリカバリー

損傷を受けた表スペースには、アクセスできない 1 つまたは複数のコンテナーがあります。これは、永続的なメディア (たとえば、ディスクに障害がある) または一時的なメディア (ディスクがオフラインになっている、またはファイル・システムがマウントされていない) の問題が原因です。

損傷を受けた表スペースがシステム・カタログ表スペースの場合には、データベースを再始動することはできません。元のデータをそのままの状態にしてコンテナーの問題を修正できない場合には、以下の選択肢しか実行できません。

- データベースをリストアする
- カタログ表スペースをリストアする。

  注:

  1. データベースをロールフォワードしなければならないので、表スペースのリストアはリカバリー可能なデータベースについてのみ有効です。
  2. カタログ表スペースをリストアする場合、ログの最後までロールフォワード操作を実行する必要があります。

損傷を受けた表スペースがシステム・カタログ表スペースでない 場合には、 DB2® はできるかぎりデータベースを使用できるようにします。

損傷を受けた表スペースが唯一の TEMPORARY 表スペースである場合には、データベースに接続できたらすぐに新しい TEMPORARY 表スペースを作成しなければ

なりません。一度作成されると、新しい TEMPORARY 表スペースが使用できるようになり、 TEMPORARY 表スペースが必要な通常のデータベース操作を再開できます。それから、任意でオフライン TEMPORARY 表スペースをドロップします。SYSTEM TEMPORARY 表スペースを使用する表の再編成については、特別な考慮事項があります。

- データベースまたはデータベース・マネージャー構成パラメーター *indexrec* が RESTART に設定されている場合、すべての無効な索引はデータベースの活動化中に再作成されなければなりません。これには、組み立てフェーズで破損した再編成からの索引も含まれます。
- 損傷を受けた TEMPORARY 表スペースで完了していない再編成の要求がある場合には、 *indexrec* 構成パラメーターを ACCESS に設定して、再始動の障害を避ける必要があります。

関連タスク:

- 13 ページの『リカバリー可能データベースの表スペースのリカバリー』
- 14 ページの『リカバリー不能データベースの表スペースのリカバリー』

関連資料:

- 「コマンド・リファレンス」の『RESTART DATABASE コマンド』
- 102 ページの『RESTORE DATABASE』

リカバリー可能データベースの表スペースのリカバリー

クラッシュ・リカバリーが必要な時には、損傷した表スペースはオフラインになり、アクセスができなくなります。それはロールフォワード・ペンディング状態になります。これ以上問題がなければ再始動操作は成功します。

手順:

損傷のある表スペースを使用可能にするには、以下のいずれかの手順を使用します。

- 方法 1
  1. 元のデータを失うことなく損傷のあるコンテナーを修正します。
  2. 表スペースのロールフォワード操作をログの終わりまで完了します。

     注: ロールフォワード操作は、最初に表スペースをオフラインから通常の状態にする操作を行います。
- 方法 2
  1. 損傷を受けているコンテナーを修正します (元のデータは失われる場合があります)。
  2. 表スペースのリストア操作を実行します。
  3. 表スペースのロールフォワード操作をログの終わりまたはポイント・イン・タイムまで完了します。

関連概念:

- 12 ページの『損傷を受けた表スペースのリカバリー』

関連タスク:

- 14 ページの『リカバリー不能データベースの表スペースのリカバリー』

関連資料:

- 「コマンド・リファレンス」の『RESTART DATABASE コマンド』
- 102 ページの『RESTORE DATABASE』

リカバリー不能データベースの表スペースのリカバリー

クラッシュ・リカバリーは必要であり、ログは永久に保持されるわけではないため、ユーザーが損傷を受けた表スペースをドロップしてもかまわない場合にだけリスタート操作が成功します。正常なリカバリー操作とは、損傷を受けた表スペースをリカバリーして整合性のある状態に戻すために必要なログ・レコードがなくなることを意味します。したがって、そのような表スペースで有効なアクションは、これらをドロップすることだけです。

手順:

損傷を受けた表スペースを含むデータベースを再始動するには、以下のようにします。

1. データベース再始動操作を、パラメーター指定なしで呼び出します。損傷を受けた表スペースがない場合には、これは成功します。失敗した (SQL0290N) 場合には、管理通知ログ・ファイルを調べて、現在損傷を受けている表スペースの完全なリストを参照します。
2. 損傷を受けた表をすべてドロップしたい場合は、別のデータベース再始動操作を開始し、 DROP PENDING TABLESPACES オプションを使用して損傷を受けたすべての表スペースをリストします。損傷を受けた表スペースが DROP PENDING TABLESPACES リストにある場合、表スペースはドロップ・ペンディング (DROP PENDING) 状態になっているため、リカバリー操作の完了後に、表スペースをドロップする必要があります。

   再始動操作は、損傷を受けた表スペースをリカバリーすることなく継続されます。損傷を受けた表スペースが DROP PENDING TABLESPACES リストにない場合、データベース再始動の操作は SQL0290N を出して失敗します。

   注: DROP PENDING TABLESPACES リストに表スペース名を入れても、この表スペースが DROP PENDING 状態になったことにはなりません。このような状態になるのは、表スペースが再始動操作中に損傷を受けた場合だけです。
3. データベース再始動操作が成功したら、LIST TABLESPACES コマンドを使用して、どの表スペースがドロップ・ペンディング状態であるかを調べてください。
4. DROP TABLESPACE ステートメントを発行してドロップ・ペンディング状態にある各表スペースをドロップします。これが完了したら、損傷を受けた表スペースが使用していたスペースを再利用するか、表スペースを再作成することができます。
5. これらの表スペースをドロップして損傷を受けた表スペースのデータを失うことを望まない場合は、以下を実行します。
   - 損傷を受けているコンテナーを修正します (元のデータを失わないようにする)。

- RESTART DATABASE コマンドを再発行してください。
- データベースのリストア操作を実行します。

関連概念:

- 12 ページの『損傷を受けた表スペースのリカバリー』

関連タスク:

- 13 ページの『リカバリー可能データベースの表スペースのリカバリー』

関連資料:

- 「コマンド・リファレンス」の『RESTART DATABASE コマンド』
- 102 ページの『RESTORE DATABASE』

メディア障害の影響の緩和

メディア障害の発生率を緩和し、またこの障害タイプからのリカバリー処理を簡単に実行できるようにするためには、以下の操作を実行します。

- 重要なデータベースのデータおよびログが含まれているディスクについて、ミラー処理を実行するか複製を作成する。
- RAID (Redundant Array of Independent Disks) 構成、たとえば RAID レベル 5 などを使用する。
- パーティション・データベース環境では、カタログ・ノードのデータおよびログを操作するための厳密な手順を設定する。データベースの保守にはこのノードが重要なので、以下の点を守ってください。
  - 必ず信頼できるディスクに常駐させる
  - 複製を作成する
  - バックアップを頻繁に取る
  - そこにユーザー・データは入れない

ディスク障害に対する保護

ディスクに障害が発生したためにデータまたはログが損傷を受ける危険性がある場合、考慮するべきことは、ディスク障害に対する何らかの許容度を持つ方策を講じておくことです。通常は、これは、ディスク・アレイ (ディスクのセット) を使用することで行われます。

ディスク・アレイは、単に RAID (新磁気ディスク制御機構) と呼ばれることがあります。ただし、ディスク・アレイはオペレーティング・システムまたはアプリケーション・レベルのソフトウェアによっても提供されています。ハードウェア・ディスク・アレイとソフトウェア・ディスク・アレイの相違点は、入出力 (I/O) 要求を CPU がどのように処理するかという点です。ハードウェア・ディスク・アレイの場合、ディスク・コントローラーが入出力アクティビティーを管理するのに対し、ソフトウェア・ディスク・アレイの場合は、オペレーティング・システムまたはアプリケーションにより実行されます。

ハードウェア・ディスク・アレイ: ハードウェア・ディスク・アレイでは、ディスク・コントローラーにより複数のディスクが使用され管理されていて、独自の CPU も備えています。アレイを構成しているディスクの管理に必要なすべてのロジック

はディスク・コントローラーに含まれています。したがって、この実行はオペレーティング・システムから独立して行われます。

RAID アーキテクチャーには、機能とパフォーマンスが異なる複数の種類がありますが、今日では通常 RAID レベル 1 とレベル 5 だけが使用されます。

RAID レベル 1 は、ディスクのミラーリングまたはデュプレキシングとも呼ばれます。ディスク・ミラーリング は、単一のディスク・コントローラーを使用し、データ (完全なファイル) をあるディスクから別のディスクにコピーします。ディスク・デュプレキシング はディスク・ミラーリングと似ていますが、ディスクは 2 番目のディスク・コントローラーにアタッチされています (2 つの SCSI アダプターと同じ)。データの保護機能は良好です。つまり、どちらのディスクに障害が発生しても、データは他のディスクからアクセス可能です。ディスク・デュプレキシングでは、データ保護を危うくすることなく、ディスク・コントローラーに障害が発生することがあります。パフォーマンスは良好ですが、これをインプリメントすると通常の 2 倍のディスクが必要になります。

RAID レベル 5 は、すべてのディスクのセクター単位のデータ・ストライピングおよびパリティー・ストライピングに関係しています。パリティーは専用ドライブに保管される代わりに、データとインターリーブされます。データの保護機能は良好です。ディスク障害が発生しても、他のディスクからの情報およびストライプされたパリティー情報を使用してアクセス可能です。読み取りパフォーマンスは良好ですが、書き込みパフォーマンスは良好ではありません。 RAID レベル 5 構成では、少なくとも 3 つの同一なディスクが必要です。オーバーヘッドのために必要なディスク・スペースは、アレイに含まれるディスク数により異なります。 5 つのディスクで構成される RAID レベル 5 構成の場合は、スペース・オーバーヘッドは 20% です。

RAID ディスク・アレイ (RAID レベル 0 ではない) を使用する場合、ディスクに障害が発生してもアレイ上のデータにはアクセスできます。常時交換可能または常時スワップ可能ディスクをアレイに使用すると、アレイ使用中に交換ディスクを障害ディスクとスワップすることが可能です。 RAID レベル 5 の場合、2 つのディスクで同時に障害が発生すると、すべてのデータは失われます (しかし、同時にディスク障害が発生する可能性はごくまれです)。

RAID レベル 1 ハードウェア・ディスク・アレイまたはソフトウェア・ディスク・アレイをログに使用できます。これは、障害点までのリカバリー可能性があり、書き込みパフォーマンスが高いためで、これはログにとって重要です。このために、*mirrorlogpath* 構成パラメーターを使用して、 RAID レベル 1 ファイル・システムのミラー・ログ・パスを指定します。 (ディスク障害発生後にただちにデータをリカバリーできるようにする必要があるため) 高信頼性が重要であるが、書き込みパフォーマンスはそれほど重要でない場合は、 RAID レベル 5 ハードウェア・ディスク・アレイの使用を考慮してください。あるいは、書き込みパフォーマンスが重要で、追加ディスク・スペースによるコストが重要でない場合は、データおよびログに RAID レベル 1 ハードウェア・ディスク・アレイを考慮してください。

使用可能な RAID レベルの詳細については、次の Web サイトにアクセスしてください。
http://www.acnc.com/04_01_00.html

ソフトウェア・ディスク・アレイ: ソフトウェア・ディスク・アレイはハードウェア・ディスク・アレイとほぼ同じ操作を実行しますが、ディスク・トラフィックは、オペレーティング・システムまたはサーバーの下で実行されるアプリケーション・プログラムのいずれかが管理します。他のプログラムと同様、ソフトウェア・アレイは CPU およびシステム・リソースを競合して獲得しなければなりません。したがって、CPU 制約システムには適しておらず、ディスク・アレイ全体のパフォーマンスがサーバーの CPU の負荷と容量に依存する点に注意する必要があります。

通常のソフトウェア・ディスク・アレイは、ディスク・ミラーリングを実行します。冗長性ディスクは必要ですが、高価なディスク・コントローラーは不要であるため、ソフトウェア・ディスク・アレイは比較的低価格で実現可能です。

注意:
オペレーティング・システムのブート・ドライブをディスク・アレイに設定すると、そのドライブに障害が発生した場合はシステムが始動しなくなります。ディスク・アレイが実行される前にドライブに障害が発生すると、ディスク・アレイは始動できないため、ドライブにアクセスすることはできません。ブート・ドライブは、ディスク・アレイから分離されていなければなりません。

トランザクション障害の影響の緩和

トランザクション障害の影響を緩和するためには、以下の条件が満たされているかどうか確認してください。

- 各 DB2® サーバーでの中断されない電源供給
- すべてのパーティションでデータベース・ログに十分なディスク・スペース
- パーティション・データベース環境においては、データベース・パーティション・サーバー間の高信頼性通信リンク
- パーティション・データベース環境では、システム・クロックの同期

関連概念:

- 142 ページの『パーティション・データベース・システムにおけるクロックの同期化』

パーティション・データベース環境におけるトランザクション障害のリカバリー

パーティション・データベース環境でトランザクション障害が起きた場合には、通常、障害を引き起こしたデータベース・パーティション・サーバーと、トランザクションに参加していた他のデータベース・パーティション・サーバーとの両方で、データベース・リカバリー処理を実行する必要があります。

- クラッシュ・リカバリーは、障害を引き起こした状態が訂正された後に、障害を引き起こしたデータベース・パーティション・サーバーで実行されます。
- 他の (アクティブのままの) データベース・パーティション・サーバーにおけるデータベース・パーティション・*リカバリー処理* は、障害が検出された直後に行われます。

パーティション・データベース環境では、アプリケーションがサブミットされているデータベース・パーティション・サーバーはコーディネーター・ノードで、最初にアプリケーションの処理を実行するエージェントはコーディネーター・エージェントです。コーディネーター・エージェントは他のデータベース・パーティション・サーバーに対し作業を分配し、どのサーバーがトランザクションに関係するかを追跡します。アプリケーションがトランザクションの COMMIT ステートメントを出すと、コーディネーター・エージェントは 2 フェーズ・コミット・プロトコルを使用してトランザクションをコミットします。最初のフェーズでは、コーディネーター・ノードはトランザクションに関係している他のすべてのデータベース・パーティション・サーバーに対し PREPARE 要求を配布します。これを受け取ると、これらのサーバーは次のいずれかで応答します。

READ-ONLY サーバーではデータの変更は行われなかった。

YES サーバーではデータの変更が行われた。

NO エラーが発生したため、サーバーはコミットの準備ができない。

いずれかのサーバーが NO で応答すると、トランザクションはロールバックされます。そうでない場合は、コーディネーター・ノードは 2 番目のフェーズを開始します。

2 番目のフェーズでは、コーディネーター・ノードは COMMIT ログ・レコードを書き出した後、 YES で応答したすべてのサーバーに対し COMMIT 要求を配布します。他のすべてのデータベース・パーティション・サーバーがコミットを完了すると、それらのサーバーはコーディネーター・ノードに対し COMMIT の肯定応答を送信します。関係するすべてのサーバーからすべての COMMIT 肯定応答をコーディネーター・エージェントが受け取ると、トランザクションは完了します。この時点で、コーディネーター・エージェントは FORGET ログ・レコードを書き出します。

アクティブ・データベース・パーティション・サーバーにおけるトランザクション障害のリカバリー

データベース・パーティション・サーバーが他のサーバーのダウンを検出すると、障害データベース・パーティション・サーバーと関連するすべての作業は、次のように停止されます。

- アクティブ・データベース・パーティション・サーバーがアプリケーションのコーディネーター・ノードで、障害データベース・パーティション・サーバー (ただし COMMIT の準備はできていない) でそのアプリケーションが実行されていた場合は、コーディネーター・エージェントは障害リカバリーを実行するための割り込みが行われます。コーディネーター・エージェントが COMMIT 処理の 2 番目のフェーズにある場合は、アプリケーションに SQL0279N が戻されてから、データベース接続が切断されます。そうでない場合は、コーディネーター・エージェントはトランザクションに関係する他のすべてのサーバーに対して ROLLBACK 要求を配布し、 SQL1229N がアプリケーションに戻されます。
- 障害データベース・パーティション・サーバーがアプリケーションのコーディネーター・ノードであった場合は、アクティブ・サーバーでそのアプリケーションに対し現在でも作業を実行しているエージェントは、障害リカバリーを実行するための割り込みが行われます。現行トランザクションは、サーバーがトランザク

ションの結果を受け取る準備ができていてその結果を待つ状態でない限り、各サーバーでローカルにロールバックされます。この場合は、アクティブ・データベース・パーティション・サーバーでトランザクションが未確定のままとされ、コーディネーター・ノードにはこのことが通知されません (コーディネーター・ノードが使用不可のため)。

- 障害データベース・パーティション・サーバーにアプリケーションが接続されていて (障害発生前)、ローカル・データベース・パーティション・サーバーも障害データベース・パーティション・サーバーもコーディネーター・ノードでない場合は、このアプリケーションの処理を実行しているエージェントは割り込みが行われます。コーディネーター・ノードは、 ROLLBACK または切断メッセージを他のデータベース・パーティション・サーバーに送信します。コーディネーター・ノードが SQL0279 を戻す場合は、トランザクションはデータベース・パーティション・サーバーで未確定になります。

障害サーバーに対し要求を送信しようとするプロセス (エージェントまたはデッドロック検出機能) には、要求が送信できない旨のメッセージが送られます。

障害の発生したデータベース・パーティション・サーバーにおけるトランザクション障害のリカバリー

トランザクション障害が発生しデータベース・マネージャーが異常終了したら、データベース・パーティションを再始動した場合は、 RESTART オプションを指定して **db2start** コマンドを出し、データベース・マネージャーを再始動することができます。データベース・パーティションを再始動できない場合は、**db2start** を出して、別のプロセッサーでデータベース・マネージャーを再始動させることができます。

データベース・マネージャーが異常終了すると、サーバー上のデータベース・パーティションは矛盾状態になることがあります。データベース・パーティションを使用可能にするために、クラッシュ・リカバリーを、データベース・パーティション・サーバー上で次のように起動することができます。

- 明示的に RESTART DATABASE コマンドを使用する。
- *autorestart* データベース構成パラメーターが ON のときは、 CONNECT 要求により暗黙的に開始される。

クラッシュ・リカバリーではアクティブ・ログ・ファイルに含まれるログ・レコードを再適用し、完全に実行されたトランザクションの結果がすべてデータベースに反映されるようにします。変更項目が再適用されると、未確定のトランザクションを除き、コミットされていないすべてのトランザクションがローカルにロールバックされます。パーティション・データベース環境では、2 種類の未確定トランザクションがあります。

- コーディネーター・ノードではないデータベース・パーティション・サーバーでは、準備されていてもまだコミットされていなければ、トランザクションは未確定になります。
- コーディネーター・ノードでは、コミットされていてもログに完了の印が付けられていなければ (つまり、 FORGET レコードがまだ書き出されていない) トランザクションは未確定になります。この状態が発生するのは、コーディネーター・エージェントが、アプリケーションに対して処理実行したすべてのサーバーから、 COMMIT 肯定応答を受け取っていないときです。

クラッシュ・リカバリーでは、以下に述べる処置のいずれかを実行することで、すべての未確定トランザクションの解決を試みます。実行されるアクションは、データベース・パーティション・サーバーがアプリケーションのコーディネーター・ノードであったかどうかにより異なります。

- 再始動されたサーバーがアプリケーションのコーディネーター・ノードでない場合は、そのサーバーはコーディネーター・エージェントに照会メッセージを送信し、トランザクションの結果を見つけます。
- 再始動されたサーバーがアプリケーションのコーディネーター・ノードである 場合、そのサーバーはコーディネーター・エージェントが COMMIT 肯定応答の待ち状態である旨のメッセージを、他のすべてのエージェント (従属エージェント) に送信します。

クラッシュ・リカバリーですべての未確定トランザクションが解決できない場合もあります (たとえば、一部のデータベース・パーティション・サーバーが使用不能の場合)。この場合、SQL 警告メッセージ SQL1061W が戻されます。未確定トランザクションはロックおよびアクティブ・ログ・スペースなどのリソースを保留するので、アクティブ・ログ・スペースが未確定トランザクションにより使用されたままになるため、データベースに対して変更を加えられなくなる場合があります。このため、クラッシュ・リカバリー後に未確定トランザクションが残っているかどうかを判別し、未確定トランザクションを解決しなければならないすべてのデータベース・パーティション・サーバーを、できるだけ早期にリカバリーする必要があります。

未確定トランザクションの解決に必要な 1 つまたは複数のサーバーのリカバリーが間に合わない場合に、他のサーバーのデータベース・パーティションにアクセスしなければならないときは、ヒューリスティックな決定を下すことで未確定トランザクションの解決を手作業で行うことができます。 LIST INDOUBT TRANSACTIONS コマンドを使用し、サーバー上の未確定トランザクションの照会、コミット、およびロールバックを行うことができます。

注: 分散トランザクション環境では、 LIST INDOUBT TRANSACTIONS コマンドも使用されます。 2 種類の未確定トランザクションを区別するために、 LIST INDOUBT TRANSACTIONS コマンドが戻す出力の *originator* フィールドには以下のいずれかが表示されます。

- DB2® Enterprise Server Edition。これは、パーティション・データベース環境で作成されたトランザクションを示しています。
- XA。これは、分散環境で作成されたトランザクションを示しています。

障害のあるデータベース・パーティション・サーバーの識別

データベース・パーティション・サーバーに障害が発生すると、アプリケーションは通常以下のいずれかの SQLCODE を受け取ります。障害が発生したデータベース・マネージャーを検出する方法は、受け取られた SQLCODE により異なります。

SQL0279N

この SQLCODE は、トランザクションに関係するデータベース・パーティション・サーバーが COMMIT 処理時に終了すると受け取られます。

SQL1224N

この SQLCODE は、障害が発生したデータベース・パーティション・サーバーがトランザクションのコーディネーター・ノードであるときに受け取られます。

SQL1229N

この SQLCODE は、障害が発生したデータベース・パーティション・サーバーがトランザクションのコーディネーター・ノードでないときに受け取られます。

どのデータベース・パーティション・サーバーに障害が発生したかの判別は、2 つのステップで構成されます。 SQLCODE SQL1229N と関連する SQLCA には、*sqlerrd* フィールドの 6 番目の配列位置にエラーを検出したサーバーのノード番号が入っています。 (サーバーについて書き出されるノード番号は、 `db2nodes.cfg` ファイルに含まれるノード番号に対応しています。) エラーを検出するデータベース・パーティション・サーバーでは、障害サーバーのノード番号を示すメッセージが管理通知ログに書き込まれます。

注: 複数の論理ノードが 1 つのプロセッサーで使用されている場合は、 1 つの論理ノードに障害が発生すると、同じプロセッサー上の他の論理ノードにも障害が発生します。

関連概念:

- 「*管理ガイド: プランニング*」の『2 フェーズ・コミット』
- 「*管理ガイド: プランニング*」の『2 フェーズ・コミット中のエラー・リカバリー』

関連タスク:

- 「*管理ガイド: プランニング*」の『未確定トランザクションの手動での解決』

関連資料:

- 「コマンド・リファレンス」の『db2start - DB2 の開始コマンド』
- 「コマンド・リファレンス」の『LIST INDOUBT TRANSACTIONS コマンド』

データベース・パーティション・サーバーの障害からのリカバリー

手順:

データベース・パーティション・サーバーの障害からリカバリーするためには、以下の操作を実行します。

1. 障害を引き起こした問題を訂正します。
2. 任意のデータベース・パーティション・サーバーから、 **db2start** コマンドを出してデータベース・マネージャーを再始動します。
3. 障害のあるデータベース・パーティション・サーバー (複数の場合もある) で、RESTART DATABASE コマンドを出してデータベースを再始動します。

関連概念:

- 17 ページの『パーティション・データベース環境におけるトランザクション障害のリカバリー』

関連資料:

- 「コマンド・リファレンス」の『db2start - DB2 の開始コマンド』
- 「コマンド・リファレンス」の『RESTART DATABASE コマンド』

DB2 Connect に DB2 Syncpoint Manager が構成されている場合のホスト上の未確定トランザクションのリカバリー

トランザクションでアプリケーションがホストまたは iSeries データベース・サーバーにアクセスした場合には、未確定トランザクションがリカバリーされる方法は多少異なります。

ホストまたは iSeries データベース・サーバーにアクセスするのに、DB2 Connect が使用されます。 DB2 Connect に DB2 Syncpoint Manager が構成されている場合、リカバリー・ステップは異なります。

手順:

ホストまたは iSeries サーバーにおける未確定トランザクションのリカバリーは、通常、トランザクション・マネージャー (TM) および DB2 Syncpoint Manager (SPM) によって自動的に行われます。ホストまたは iSeries サーバーの未確定トランザクションはローカル DB2 のロケーションにはリソースを保持しませんが、トランザクションがホストまたは iSeries サーバーで未確定である間はその位置にリソースを保持します。ヒューリスティックな決定を行う必要があるとホストまたは iSeries の管理者が判断すると、ホストまたは iSeries でトランザクションをコミットするかロールバックするかを決定するために、管理者はローカル DB2 データベース管理者と連絡をとります (たとえば、電話で)。これが行われると、LIST DRDA INDOUBT TRANSACTIONS コマンドを使用して、ローカル DB2 Connect インスタンスでのトランザクションの状態を判別することができます。 SNA 通信環境を使用している場合は通常、以下のステップを指針として使用することができます。

1. 次のようにして、SPM に接続します。

```
db2 => connect to db2spm

 Database Connection Information

 Database product       = SPM0500
 SQL authorization ID   = CRUS
 Local database alias   = DB2SPM
```

2. LIST DRDA INDOUBT TRANSACTIONS コマンドを出して、 SPM から認識できる未確定トランザクションを表示します。以下の例は、SPM に認識された 1 つの未確定トランザクションを示します。 db_name がホストまたは iSeries サーバーのローカル別名です。 partner_lu がホストまたは iSeries サーバーの完全修飾 LU 名です。これはホストまたは iSeries サーバーを最もよく識別できるもので、ホストまたは iSeries サーバーの呼び出し側が指定してください。 luwid はトランザクションのユニーク ID を提供するもので、すべてのホストまたは iSeries サーバーで使用可能です。今話題にしているトランザクションが表示されている場合には、 uow_status フィールドを用いて、値が C (コミット) か R (ロールバック) の場合のトランザクションの結果を判別することができます。 WITH PROMPTING パラメーターを指定して LIST DRDA INDOUBT TRANSACTIONS コマンドを出す場合は、トランザクションのコミット、ロールバック、無視を対話式に行えます。

```
db2 => list drda indoubt transactions
 DRDA Indoubt Transactions:
 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. partner_lu および luwid の未確定トランザクションが表示されていない場合、または LIST DRDA INDOUBT TRANSACTIONS コマンドが次のようにして戻る場合は、

   ```
   db2 => list drda indoubt transactions
   SQL1251W  No data returned for heuristic query.
   ```

   トランザクションはロールバックされました。

   起こりそうもないが可能性はある、別の状況が生じている場合もあります。正しい luwid と partner_lu を指定した未確定トランザクションが表示されても、uow_status が ″I″ の場合は、SPM はトランザクションがコミットされるのか、ロールバックされるのかを認識しません。この場合、DB2 Connect ワークステーションでトランザクションをコミット、またはロールバックするために、WITH PROMPTING パラメーターを使用する必要があります。その後、DB2 Connect がヒューリスティックな決定に基づいて、ホストまたは iSeries サーバーとの再同期を行えるようにします。

関連タスク:

- 23 ページの『DB2 Connect が DB2 同期点マネージャーを使用しない場合のホスト上の未確定トランザクションのリカバリー』

関連資料:

- 「コマンド・リファレンス」の『db2start - DB2 の開始コマンド』
- 「コマンド・リファレンス」の『LIST INDOUBT TRANSACTIONS コマンド』
- 「コマンド・リファレンス」の『RESTART DATABASE コマンド』

DB2 Connect が DB2 同期点マネージャーを使用しない場合のホスト上の未確定トランザクションのリカバリー

トランザクションでアプリケーションがホストまたは iSeries データベース・サーバーにアクセスした場合には、未確定トランザクションがリカバリーされる方法は多少異なります。

ホストまたは iSeries データベース・サーバーにアクセスするのに、DB2 Connect が使用されます。 DB2 Connect に DB2 Syncpoint Manager が構成されている場合、リカバリー・ステップは異なります。

手順:

DB2 Connect Personal Edition または DB2 Connect Enterprise Server Edition のいずれかからのマルチサイト更新で、 DB2 (z/OS 版) を更新するために TCP/IP 接続が使用されていて、DB2 Syncpoint Manager が使用されない場合は、このセクションの情報を活用してください。この状態での未確定トランザクションのリカバリーは、 DB2 Syncpoint Manager が関係する未確定トランザクションのリカバリーとは

異なります。この環境で未確定トランザクションが発生すると、その問題の検出元に従い、クライアント、データベース・サーバー、またはトランザクション・マネージャー (TM) データベース (あるいはそれらの複数の組み合わせ) でアラート項目が生成されます。アラート項目は、db2alert.log ファイルに保管されます。

TM および関係するデータベースとその接続すべてが再び使用可能になると、未確定トランザクションは自動的に再同期化されます。データベース・サーバーでヒューリスティックな決定を強制するのではなく、自動的に再同期が行われるようにする必要があります。しかし、このようにする場合は、以下のステップをガイドラインとしてください。

注: DB2 Syncpoint Manager は関係していないので、 LIST DRDA INDOUBT TRANSACTIONS コマンドは使用できません。

1. z/OS ホストで、DISPLAY THREAD TYPE(INDOUBT) コマンドを出します。

   このリストから、ヒューリスティックな手法によって完了させたいトランザクションを識別します。 DISPLAY コマンドの詳細については、「*DB2 (z/OS 版) コマンド・リファレンス*」を参照してください。表示される LUWID を、トランザクション・マネージャー・データベースでの同じ luwid に一致させることができます。

2. 行うことに基づいて、 RECOVER THREAD(<LUWID>) ACTION(ABORT|COMMIT) コマンドを出します。

   RECOVER THREAD コマンドの詳細については、「*DB2 (z/OS 版) コマンド・リファレンス*」を参照してください。

関連タスク:

- 22 ページの『DB2 Connect に DB2 Syncpoint Manager が構成されている場合のホスト上の未確定トランザクションのリカバリー』

関連資料:

- 「コマンド・リファレンス」の『LIST INDOUBT TRANSACTIONS コマンド』

災害時リカバリー

災害時リカバリー という用語は、火災、地震、破壊行為、または他の災害が発生した場合にデータベースをリストアするために、実行しなければならない活動の記述に使用されます。災害時リカバリーの計画には、以下の 1 つまたは複数が含まれます。

- 非常事態発生時に使用されるサイト
- データベースをリカバリーするための別のマシン
- データベース・バックアップおよびアーカイブ・ログのオフサイト・ストレージ

災害時リカバリー計画が別のマシンでのデータベース全体のリカバリーを意味する場合は、少なくとも完全なデータベース・バックアップとデータベースに関するすべてのアーカイブ・ログが必要です。ログをアーカイブする際にそれらを予備のデータベースに適用することによって、そのデータベースを最新の状態にしておくことができます。あるいは、データベース・バックアップとログ・アーカイブを予備のサイトに保持し、災害発生後にだけリストアおよびロールフォワードを実行する

ようにもできます。 (この場合、最新のデータベース・バックアップがぜひとも必要です。) しかし、災害時の場合は、すべてのトランザクションを災害発生時までリストアすることは通常は不可能です。

災害時リカバリーに表スペースのバックアップが役に立つかどうかは、障害の範囲によって決まります。通常、災害時リカバリーではデータベース全体をリストアする必要があるので、スタンバイ・サイトに全データベースのバックアップを保持する必要があります。すべての表スペースの別々のバックアップ・イメージがあるとしても、データベースをリストアするためにそれらを使用することはできません。その災害がディスクの損傷である場合には、表スペースのバックアップをそのディスクにある表スペースごとにリカバリーに使用できます。ディスク障害 (または他の理由) によりコンテナーにアクセスできない場合は、コンテナーを別の場所にリストアできます。

部分的または完全なサイト障害からデータを保護するための別の方法は、 DB2® 高可用性災害時リカバリー (HADR) を実装することです。 HADR をセットアップすると、データの変更内容を、1 次データベースと呼ばれるソース・データベースから、スタンバイ・データベースと呼ばれるターゲット・データベースへ複製することにより、データを損失から保護します。

DB2 には、災害時リカバリーを計画する際のオプションがいくつか用意されています。ビジネスのニーズに応じ、データ損失に対する保護手段として、表スペース・バックアップか、データベース全体のバックアップを使用することを決定できますし、それぞれの環境が HADR のようなソリューションによく合っていると判断する場合もあります。どのような選択であっても、リカバリー手順をそれぞれの実稼働環境にインプリメントする前に、テスト環境でそれらをテストする必要があります。

関連概念:

- 100 ページの『リストア操作 (リダイレクト・リストア) 時の表スペース・コンテナーの再定義』
- 201 ページの『高可用性災害時リカバリーの概要』

バージョン・リカバリー

バージョン・リカバリー は、以前のバージョンのデータベースのリカバリーであり、バックアップ操作で作成されたイメージを使用して行われます。このリカバリー方式は、リカバリー不能データベース (つまり、管理者がアーカイブ・ログを持っていないデータベース) に使用します。 RESTORE DATABASE コマンドで WITHOUT ROLLING FORWARD オプションを使用して、リカバリー可能データベースでこの方式を使用することもできます。データベース・リストア操作では、以前に作成されたバックアップ・イメージを使用して、データベース全体が再構築されます。データベースのバックアップにより、データベースを、バックアップをとった時点と同じ状態にリストアすることができます。しかし、バックアップ時点から障害発生時点までのすべての作業単位は失われています (26 ページの図 3 を参照)。

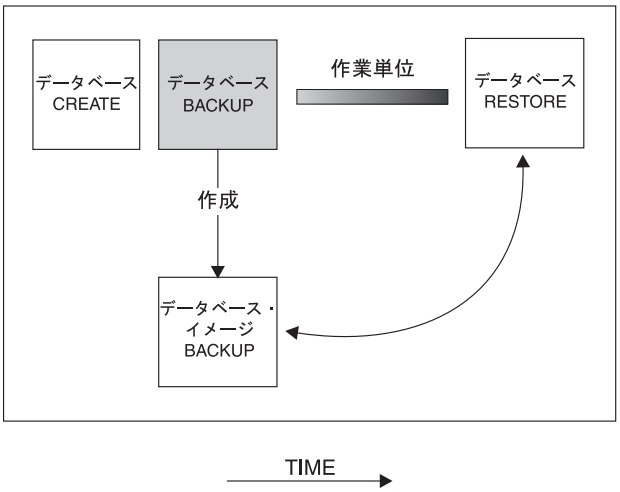


図 3. バージョン・リカバリー: データベースは最新のバックアップ・イメージからリストアされますが、バックアップを取ってから障害が起きるまでの間に処理された作業単位はすべて失われます。

バージョン・リカバリー方式を使用して、データベースのフル・バックアップの計画を立てて、定期的に実行してください。

パーティション・データベース環境では、データベースは多数のデータベース・パーティション・サーバー (またはノード) にまたがって存在します。すべてのパーティションをリストアすることと、データベース・リストア操作に使用するすべてのバックアップ・イメージを同時に作成することが必要です。 (各データベース・パーティションは、別々にバックアップされ、リストアされます。) 同時に作成される各データベース・パーティションのバックアップは、バージョン・バックアップと呼ばれます。

ロールフォワード・リカバリー

ロールフォワード・リカバリー 方式を使用するためには、データベースのバックアップを作成しておき、ログをアーカイブする必要があります (これは、*logarchmeth1* および *logarchmeth2* 構成パラメーターを、 OFF 以外の値に設定することで実行できます)。データベースをリストアして、 WITHOUT ROLLING FORWARD オプションを指定することは、バージョン・リカバリー方式を使用することと同じです。データベースはオフライン・バックアップ・イメージをとった時点と同一の状態にリストアされます。データベースをリストアする際、データベース・リストア操作で WITHOUT ROLLING FORWARD オプションを指定していない 場合、そのデータベースはリストア操作の終了時にロールフォワード・ペンディング状態になります。これで、ロールフォワード・リカバリーを実行できるようになります。

注: データベース・バックアップがオンラインで取られた場合は、WITHOUT ROLLING FORWARD オプションは使用できません。

考慮する 2 つのロールフォワード・リカバリーは次のとおりです。

- *データベースのロールフォワード・リカバリー*。このタイプのロールフォワード・リカバリーでは、データベース・ログに記録されているトランザクションが

データベース・リストア操作の後に適用されます (図 4 を参照)。データベース・ログには、データベースへの変更がすべて記録されています。この方式では、データベースが特定の時点の状態に、または障害が生じる直前 (アクティブ・ログの最後) の状態にリカバリーされます。

パーティション・データベース環境では、データベースは多数のデータベース・パーティションにまたがって存在します。データベースのカタログ表が存在するパーティション (カタログ・パーティション) に対して ROLLFORWARD DATABASE コマンドを発行する必要があります。ポイント・イン・タイム指定ロールフォワード・リカバリーを実行する場合は、すべてのデータベース・パーティションをロールフォワードし、すべてのパーティションが同じレベルになるようにする必要があります。単一のデータベース・パーティションをリストアしなければならない場合は、ログの最後までロールフォワード・リカバリーを実行して、そのパーティションをデータベース内の他のパーティションと同じレベルにすることができます。 1 つのデータベース・パーティションをロールフォワードする場合は、ログの最後までのロールフォワード・リカバリーだけを実行できます。ポイント・イン・タイム指定リカバリーの適用対象は、すべての データベース・パーティションです。

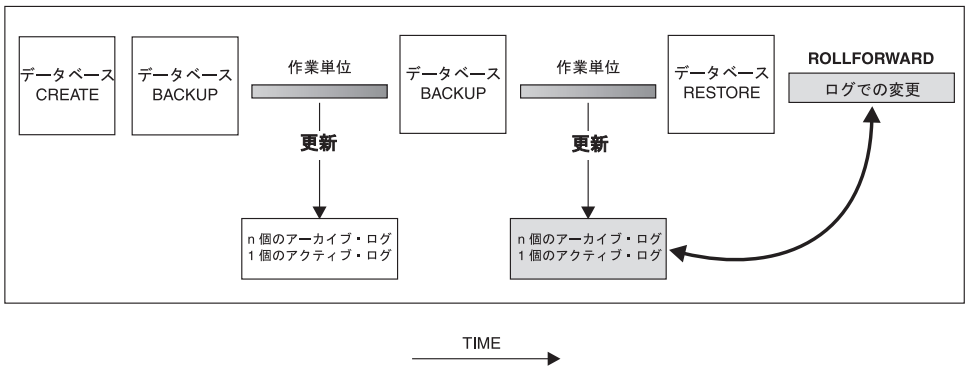


図 4. データベースのロールフォワード・リカバリー: 長時間実行しているトランザクションの場合には、複数のアクティブ・ログが生じる可能性があります。

- *表スペースのロールフォワード・リカバリー*。データベースの順方向リカバリーが可能であれば、表スペースのバックアップ、リストア、およびロールフォワードも可能です (28 ページの図 5 を参照)。表スペースのリストアおよびロールフォワードを行う場合、データベース全体 (つまり、すべての表スペース)、または 1 つまたは複数の個別表スペースのバックアップが必要です。さらに、リカバリーする表スペースに影響を与えるログ・レコードが必要です。以下の 2 つのポイントのうちの 1 つにログをロールフォワードできます。
  - ログの終わった時点。または、
  - 特定の時点 (*ポイント・イン・タイム指定* リカバリーと呼ばれる)。

表スペース・ロールフォワード・リカバリーは、以下の 2 つの場合に使用されます。

- 表スペース・リストア操作の後、表スペースは常にロールフォワード・ペンディング状態で、ロールフォワードする必要があります。 ROLLFORWARD DATABASE コマンドを呼び出し、特定の時点またはログの最後まで表スペースにログを適用してください。

- クラッシュ・リカバリーの後で 1 つまたは複数の表スペースがロールフォワード・ペンディング 状態の場合、まず表スペースの問題を訂正します。場合によっては、表スペースの問題を訂正するのにデータベース・リストア操作が関係しない場合もあります。たとえば、電源が切れると、表スペースはロールフォワード・ペンディング状態になります。この場合にはデータベース・リストア操作は必要ありません。表スペースの問題が解決したら、ROLLFORWARD DATABASE コマンドを使用して、ログの最後までのログを、表スペースに適用することができます。クラッシュ・リカバリーの前にこの問題が訂正された場合、データベースを整合性のある使用可能状態にするのにクラッシュ・リカバリーで十分です。

  注: エラーが発生した表スペースにシステム・カタログ表が含まれている場合は、データベースを開始することはできません。 SYSCATSPACE 表スペースをリストアした後、ログの終わりまでロールフォワード・リカバリーを実行する必要があります。

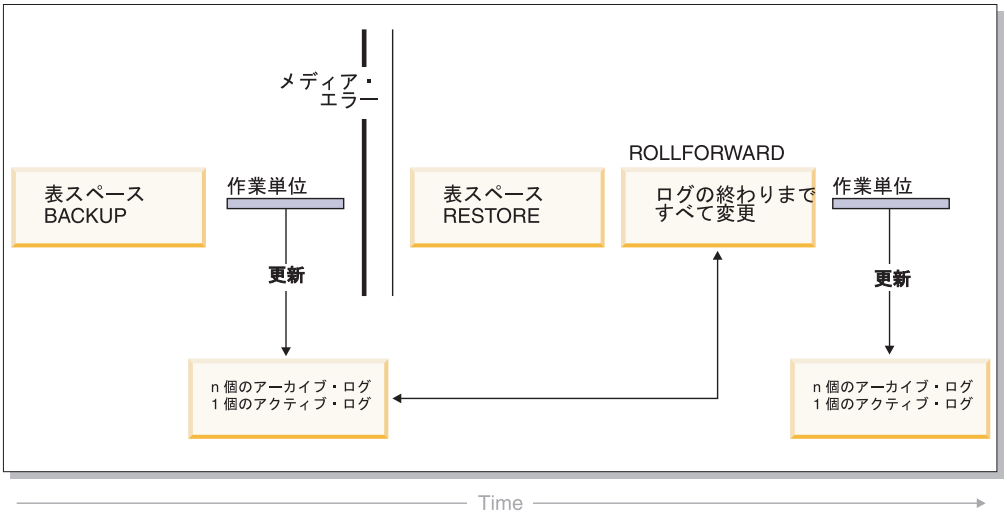


図 5. 表スペースのロールフォワード・リカバリー: 長時間実行しているトランザクションの場合には、複数のアクティブ・ログが生じる可能性があります。

パーティション・データベース環境では、表スペースを *特定の時点まで* ロールフォワードする場合、表スペースが常駐するノード (データベース・パーティション) のリストを指定する必要はありません。 DB2® は、すべてのパーティションにロールフォワード要求をサブミットします。これは、表スペースが常駐するすべてのデータベース・パーティションで表スペースをリストアしなければならないことを意味します。

パーティション・データベース・システムでは、*ログの最後まで* 表スペースをロールフォワードする場合で、すべてのパーティションで表スペースをロールフォワードしたくない 場合は、データベース・パーティションのリストを提供する必要があります。 (すべてのパーティション上にある) ロールフォワード・ペンディング状態のすべての表スペースをログの最後までロールフォワードしたい場合には、データベース・パーティションのリストを指定する必要はありません。デフォルトでは、ロールフォワード要求はすべてのパーティションに送信されます。

関連概念:

- 36 ページの『リカバリー・ログについて』

関連資料:

- 143 ページの『ROLLFORWARD DATABASE』

関連サンプル:

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

増分バックアップおよびリカバリー

データベース (特にウェアハウス) のサイズがテラバイトやペタバイトの範囲で拡張し続けるにつれて、データベースのバックアップとリカバリーに必要なハードウェア・リソースも実質的に大きくなっていきます。大規模なデータベースの場合、その複数コピーのストレージ要件も大きくなるので、このようなデータベースの場合は全データベースや表スペースのバックアップを取るのは必ずしも最善とはいえません。以下の問題を考慮に入れてください。

- ウェアハウス中のデータ変更のパーセンテージが低い場合は、データベース全体のバックアップを取るべきではない。
- 既存のデータベースに表スペースを付加した後に表スペースのバックアップしか行わないのは危険。なぜなら、表スペースのバックアップを行っている間に、バックアップ対象の表スペース以外に変更が加えられなかった保証はないからです。

この問題に取り組むため、DB2® には増分バックアップおよびリカバリーが用意されています。*増分バックアップ* とは、前回のバックアップ以降に更新されたページだけを含むバックアップ・イメージのことです。個々の増分バックアップ・イメージには、更新されたデータ・ページと索引ページに加えて、通常は全バックアップ・イメージに保管されるすべての初期データベース・メタデータ (データベースの構成、表スペースの定義、データベースの履歴など) も含まれます。

注: 表スペースに長形式フィールドまたはラージ・オブジェクト・データが含まれていて、増分バックアップが行われる場合に、その表スペースの任意のページが前のバックアップから変更されていると、長形式フィールドまたはラージ・オブジェクト・データはすべて、バックアップ・イメージにコピーされます。

2 種類の増分バックアップがサポートされています。

- *増分*。増分バックアップ・イメージは、最新の正常実行された全バックアップ操作の後に変更された、すべてのデータベース・データのコピーです。これは累積バックアップ・イメージともいいます。増分バックアップを取るたびに、その前の増分バックアップ・イメージの内容が含まれるからです。増分バックアップ・イメージの先行処理イメージは、常に同じオブジェクトの最新の正常実行された全バックアップになります。
- *差分*。差分、または増分差分のバックアップ・イメージは、当該表スペースの正常実行された最終バックアップ (全、増分、または差分) の後に変更されたすべてのデータベース・データのコピーです。これは差分または非累積バックアップ・

イメージともいいます。差分バックアップ・イメージの先行処理イメージは、その差分バックアップ・イメージ中の個々の表スペースのコピーを含む、正常実行された最新バックアップになります。

増分と差分のバックアップ・イメージの主要な違いは、継続的に変更が加えられるオブジェクトのバックアップを連続して取る場合の動作にあります。増分イメージには、その前の増分イメージの内容がすべて含まれ、前回の全バックアップ作成以降に変更されたデータや新規データも含まれます。差分バックアップ・イメージには、任意のタイプのイメージが前回作成されてから変更されたページだけが含まれます。

オフライン・モードとオンライン・モードの操作の両方で、データベースと表スペースの増分バックアップを組み合わせることができます。データベースと表スペースの増分バックアップを組み合わせた場合、データベース・バックアップ (または複数の表スペースのバックアップ) の先行処理イメージは必ずしも 1 つのイメージとは限らず、過去のさまざまな時点で取られたデータベースと表スペースのバックアップのユニークな集合になる可能性も生じるので、バックアップ計画を作成する際には注意してください。

データベースや表スペースを整合性のある状態に作成し直す場合は、リストア対象のオブジェクト (データベースまたは表スペース) 全体の整合性のあるイメージを使用してリカバリー・プロセスを始めなければならず、開始後下記の順序で個々の該当する増分バックアップ・イメージを適用しなければなりません。

DB2 では、データベースの更新を追跡できるようにするために、新しいデータベース構成パラメーターの *trackmod* がサポートされています。このパラメーターは以下の 2 つの値のいずれかを受け入れることができます。

- NO。この構成で増分バックアップを行えません。データベース・ページの更新を追跡したり記録したりする方法はありません。これはデフォルト値です。
- YES。この構成で増分バックアップを行えます。更新を追跡できるようにすると、変更内容はデータベースに初めて正常に接続した時点で有効になります。特定の表スペースの増分バックアップをとる前に、その表スペースの全バックアップをすることが必要です。

SMS および DMS 表スペースの場合は、この追跡の細分度は表スペース・レベルになります。表スペース・レベルのトラッキングで、それぞれの表スペースのフラグは、その表スペースにバックアップが必要なページがあるかどうかを表します。表スペースの中にバックアップが必要なページがない場合には、バックアップ操作はその表スペースをまとめてスキップできます。

データベースに対する更新の追跡は、データを更新したり挿入したりするトランザクションの実行時パフォーマンスに、最小限とはいえ影響があります。

関連タスク:

- 31 ページの『増分バックアップ・イメージからのリストア』

増分バックアップおよびリカバリー - 詳細

増分バックアップ・イメージからのリストア

手順:

増分バックアップ・イメージからのリストア操作は、常に以下のステップから成ります。

1. 増分ターゲット・イメージを識別します。

   リストアする最終イメージを決定し、DB2 リストア・ユーティリティーによる増分リストア操作を要求しなければなりません。このイメージは、リストアされる最終イメージになるので、増分リストアのターゲット・イメージとして認識されます。増分ターゲット・イメージは RESTORE DATABASE コマンドの TAKEN AT パラメーターを使用して指定します。
2. 最新の全データベースまたは表スペースのイメージをリストアして、以後の増分バックアップ・イメージの適用対象となるベースラインを確立します。
3. ステップ 2 でリストアしたベースライン・イメージの上部に、個々の必要な全データベースまたは表スペース増分バックアップ・イメージを、作成順にリストアします。
4. ステップ 1 のターゲット・イメージが 2 度目に読み取られるまで、ステップ 3 を繰り返します。増分リストア操作が完了するまでの間に、ターゲット・イメージは 2 度アクセスされます。最初のアクセスの際には、初期データだけがイメージから読み取られます。ユーザー・データは読み取られません。イメージが完全に読み取られて処理されるのは 2 度目のアクセス時だけです。

   リストア操作時に作成されるデータベースが、確実に正しい履歴、データベース構成、および表スペース定義で初期構成されるようにするには、増分リストア操作のターゲット・イメージに 2 度アクセスしなければなりません。最初に全データベースのバックアップ・イメージを取った後で表スペースがドロップされた場合、増分リストア処理の際に、バックアップ・イメージからそのイメージの表スペース・データが読み取られますが無視されます。

増分バックアップ・イメージをリストアするには、2 つの方法があります。

- 自動増分リストアでは、RESTORE コマンドを 1 回だけ発行して、使用するターゲット・イメージを指定します。その後 DB2 は、データベース履歴を使用して残っている必要なバックアップ・イメージを判別し、それらをリストアします。
- 手動増分リストアでは、リストアする必要があるバックアップ・イメージごとに、 RESTORE コマンドを 1 回発行する必要があります (上記のステップで説明されています)。

自動増分リストアの例

自動増分リストアを使用して増分バックアップ・イメージの集合をリストアするには、 RESTORE DATABASE コマンドに TAKEN AT timestamp オプションを指定してください。リストアしたい最終イメージのタイム・スタンプを使用してください。次に例を示します。

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

この場合、DB2 リストア・ユーティリティーによって、このセクションの始めにある個々のステップが自動的に実行されます。処理の最初の段階で、タイム・スタンプ 20001228152133 のバックアップ・イメージが読み取られ、リストア・ユーティリティーによってデータベースとその履歴、および表スペース定義があって有効であるか検査されます。

処理の 2 番目の段階で、データベース履歴が照会され、要求されたリストア操作を実行するのに必要なバックアップ・イメージのチェーンが構築されます。何らかの理由で照会できず、必要なイメージのチェーンが完全に構築できない場合、リストア操作は終了し、エラー・メッセージが戻されます。このような場合は、自動増分リストアは行えないので、 RESTORE DATABASE コマンドを INCREMENTAL ABORT オプションとともに発行する必要があります。このようにするとすべての残っているリソースをクリーンアップしますので、手動増分リストアを行うことができます。

注: PRUNE HISTORY コマンドの FORCE オプションを指定して使用しないよう強くお勧めします。このコマンドのデフォルトの操作では、最新の全データベース・バックアップ・イメージからリカバリーする際に必要になることがある履歴項目は削除されませんが、 FORCE オプションを指定すると、自動リストア操作に必要な項目を削除できるようになります。

プロセスの第 3 段階の時に、DB2 は生成されたチェーン内に残っているそれぞれのバックアップ・イメージをリストアします。この段階の時にエラーが発生した場合、RESTORE DATABASE コマンドを INCREMENTAL ABORT オプションとともに発行して、すべての残っているリソースをクリーンアップする必要があります。その後再び RESTORE コマンドを発行したり、手動増分リストアを再び試行する前に、エラーが解決しているかを判別する必要があります。

手動増分リストアの例

増分バックアップ・イメージの集合をリストアするには、手動増分リストアを用いて、 RESTORE DATABASE コマンドの TAKEN AT *timestamp* オプションを使用してターゲット・イメージを指定してください。そして上で説明されているステップに従います。次に例を示します。

1. db2 restore database sample incremental taken at <ts>

   ただし、
   <ts> はリストアされる最後の増分バックアップ・イメージ (ターゲット・イメージ) を指します。

2. db2 restore database sample incremental taken at <ts1>

   ただし、
   <ts1> は最初の全データベース (または表スペース) イメージを指します。

3. db2 restore database sample incremental taken at <tsX>

   ただし、
   <tsX> は作成順の個々の増分バックアップ・イメージを指します。

4. 増分バックアップ・イメージ
   <ts> を含めすべての増分バックアップ・イメージがリストアできるまで、
   ステップ 3 を繰り返します。

データベースのリストア操作を実行しており、表スペースのバックアップ・イメージが作成されている場合に、表スペースのイメージをバックアップのタイム・スタンプの日時順にリストアしなければなりません。

db2ckrst ユーティリティーを使用してデータベース履歴を照会し、増分リストアに必要なバックアップ・イメージのタイム・スタンプのリストを生成することができます。手操作の増分リストアに使用する、単純化されたリストア構文も生成されます。バックアップの完全な記録を保持し、このユーティリティーはガイドとしてのみ使用することをお勧めします。

関連概念:

- 29 ページの『増分バックアップおよびリカバリー』

関連資料:

- 102 ページの『RESTORE DATABASE』
- 285 ページの『db2ckrst - 増分リストア・イメージ順序の検査』

自動増分リストアの制限

1. リストア元にするバックアップ操作をした後で表スペースの名前を変更し、表スペース・レベルのリストア操作を発行する時に新しい名前を使用した場合、データベース履歴からの必要なバックアップ・イメージのチェーンが正しく生成されず、エラーになります (SQL2571N)。

   例:

   ```
   db2 backup db sample -> <ts1>
   db2 backup db sample incremental -> <ts2>
   db2 rename tablespace from userspace1 to t1
   db2 restore db sample tablespace ('t1') incremental automatic taken
   at <ts2>

   SQL2571N Automatic incremental restore is unable to proceed.
   Reason code: "3".
   ```

   推奨されている対処策: 手動増分リストアを使用してください。

2. データベースをドロップすると、データベース履歴は削除されます。ドロップしたデータベースをリストアすると、データベース履歴は、リストア元のバックアップが取られたときの状態にリストアされ、それ以降の履歴項目はすべて失われます。その後、失われた履歴項目を使用する必要のある自動増分リストアを試行すると、 RESTORE ユーティリティーにより不正確なバックアップのチェーンのリストアが試行され、 ″out of sequence″ エラーが戻されます (SQL2572N)。

   例:

   ```
   db2 backup db sample -> <ts1>
   db2 backup db sample incremental -> <ts2>
   db2 backup db sample incremental delta -> <ts3>
   db2 backup db sample incremental delta -> <ts4>
   db2 drop db sample
   db2 restore db sample incremental automatic taken at <ts2>
   db2 restore db sample incremental automatic taken at <ts4>
   ```

   推奨されている対処策:

   - 手動増分リストアを使用してください。

- 最初にイメージ <ts4> から履歴ファイルをリストアしてから、自動増分リストアを実行してください。

3. バックアップ・イメージをあるデータベースから別のデータベースにリストアし、その後増分 (差分) バックアップをする場合、もはや自動増分リストアを使用してこのバックアップ・イメージをリストアできません。

   例:

   ```
   db2 create db a
   db2 create db b

   db2 update db cfg for a using trackmod on

   db2 backup db a -> ts1
   db2 restore db a taken at ts1 into b

   db2 backup db b incremental -> ts2

   db2 restore db b incremental automatic taken at ts2

   SQL2542N No match for a database image file was found based on the source
   database alias "B" and timestamp "ts1" provided.
   ```

   推奨されている対処策:

   - 以下のようにして手動増分リストアを使用してください。

     ```
     db2 restore db b incremental taken at ts2
     db2 restore db a incremental taken at ts1 into b
     db2 restore db b incremental taken at ts2
     ```

   - データベース B への手動リストア操作の後、フル・データベース・バックアップを発行して、新規に増分チェーンを開始してください。

関連概念:

- 29 ページの『増分バックアップおよびリカバリー』

関連タスク:

- 31 ページの『増分バックアップ・イメージからのリストア』

関連資料:

- 102 ページの『RESTORE DATABASE』

バックアップ、リストア、およびリカバリー操作の進行状況をモニターする

LIST UTILITIES コマンドを使用して、データベースでのバックアップ、リストア、およびロールフォワード操作をモニターできます。

手順:

バックアップ、リストア、およびリカバリー操作の進捗モニターを使用するには、次のようにします。

- LIST UTILITIES コマンドを発行して、SHOW DETAIL オプションを指定します。

  ```
  list utilities show detail
  ```

次に示すのは、オフライン・データベース・バックアップ操作でのパフォーマンスをモニターするときの出力例です。

```
LIST UTILITIES SHOW DETAIL

ID                          = 2
Type                        = BACKUP
Database Name               = SAMPLE
Description                 = offline db
      Start Time            = 10/30/2003 12:55:31.786115
Throttling:
Priority                    = Unthrottled
Progress Monitoring:
Estimated Percentage Complete    = 41
      Total Work Units      = 20232453 bytes
      Completed Work Units = 230637 bytes
      Start Time            = 10/30/2003 12:55:31.786115
```

バックアップ操作の場合、処理されるバイト数の初期見積もりが指定されます。バックアップ操作が進行するにつれて、処理されるバイト数が更新されてゆきます。表示されるバイト数は、イメージのサイズに対応するものではないので、バックアップ・イメージ・サイズの見積もりには使用しないようにします。実際のイメージは、増分バックアップか圧縮バックアップかに応じて、はるかに小さい可能性があります。

リストア操作の場合、初期見積もりは示されません。代わりに、UNKNOWN が指定されます。各バッファーがイメージから読み取られると、実際の読み取りバイト数が更新されます。複数のイメージをリストアできる自動増分リストア操作の場合、進行状況はフェーズを使用して追跡されます。各フェーズは、増分チェーンからリストアされるイメージを表します。初めは、1 つのフェーズだけが示されます。最初のイメージがリストアされたら、フェーズの累計が示されます。各イメージがリストアされると、完了したフェーズ数は、処理済みバイト数として更新されます。

クラッシュ・リカバリーおよびロールフォワード・リカバリーの場合、進行状況モニターには、FORWARD と BACKWARD の 2 つのフェーズがあります。FORWARD フェーズでは、ログ・ファイルが読み取られて、そのログ・レコードがデータベースに適用されます。クラッシュ・リカバリーの場合、合計作業量は、最新ログ・ファイルの終わりまでの開始シーケンス番号を使用して見積もられます。ロールフォワード・リカバリーの場合、このフェーズを開始すると、作業合計の見積もりには UNKNOWN が指定されます。バイト単位での処理済み作業量は、処理の進捗と共に更新されます。

BACKWARD フェーズでは、FORWARD フェーズで適用されたコミットされない変更は、ロールバックされます。処理されるログ・データの量のバイト単位での見積もりが示されます。バイト単位での処理済み作業量は、処理の進捗と共に更新されます。

関連概念:

- 11 ページの『クラッシュ・リカバリー』
- 69 ページの『バックアップの概要』
- 95 ページの『リストアの概要』
- 129 ページの『ロールフォワードの概要』

関連資料:

- 「コマンド・リファレンス」の『LIST UTILITIES コマンド』

リカバリー・ログについて

すべてのデータベースには、それと結び付くログがあります。これらのログには、データベース変更の記録が保持されています。データベースを最後の全オフライン・バックアップよりも後の時点にリストアする必要がある場合は、データを障害発生時点までロールフォワードするためにログが必要です。

DB2® ロギングには 2 つのタイプがあります。*循環* およびアーカイブ であり、それぞれは、異なるレベルのフェイルオーバー・リカバリーを提供します。

- *循環* ロギングは、新規のデータベースが作成されるときのデフォルト動作です。(*logarchmeth1* および *logarchmeth2* データベース構成パラメーターは OFF に設定されます。) このタイプのロギングでは、データベースの全オフライン・バックアップのみが許可されます。全バックアップを作成するとき、データベースはオフライン (ユーザーからアクセス不能) でなければなりません。名前から分かるとおり、循環ロギングはオンライン・ロギングの「輪」を使用して、トランザクション障害およびシステム破損からのリカバリーを提供します。ログは、現行トランザクションの保全性を保証するためにのみ使用および保存されます。循環ロギングでは、最後に行った全バックアップ操作より後に実行されたトランザクションを使用してデータベースをロールフォワードすることはできません。最後のバックアップ操作以降に加えられた変更はすべて失われます。このタイプのリストア操作では、全バックアップが取られた時点までデータがリカバリーされるため、これは、バージョン・リカバリー と呼ばれています。

  図 6 は、循環ロギングがアクティブなときにアクティブ・ログがログ・ファイルの輪を使用する様子を示しています。

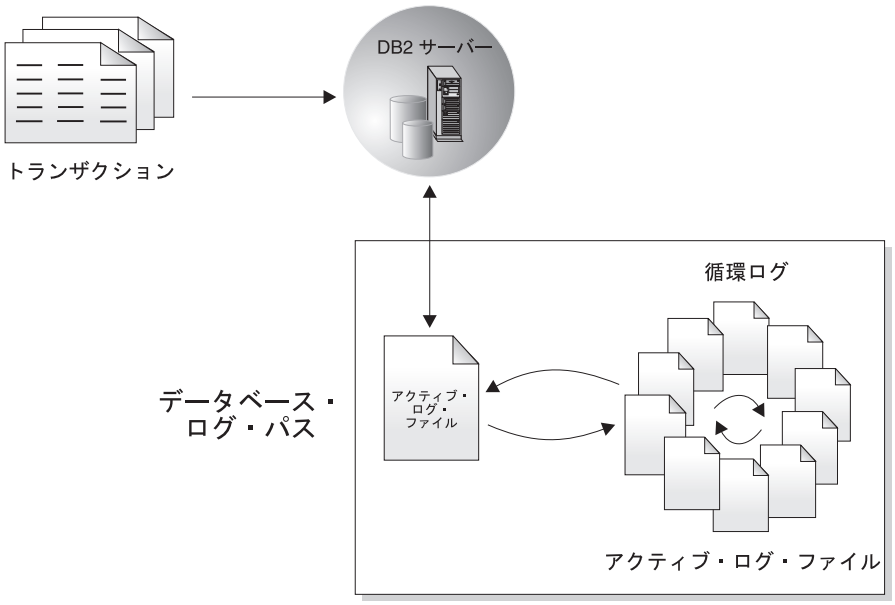


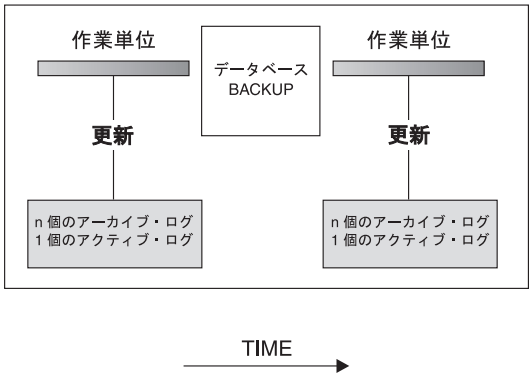
図 6. 循環ロギング

アクティブ・ログは、障害 (システム電源またはアプリケーション・エラー) が原因でデータベースが整合性のない状態になるのを防ぐために、クラッシュ・リカバリー処理中に使用されます。アクティブ・ログは、データベース・ログ・パス・ディレクトリーにあります。

- アーカイブ・ロギングは、ロールフォワード・リカバリーに使用されます。アーカイブ・ログは、アクティブであったものの、もはやクラッシュ・リカバリーに必要のないログです。アーカイブ・ロギングを使用可能にするには、*logarchmeth1* データベース構成パラメーターを使用します。

アーカイブ・ロギングを選択する利点は、ロールフォワード・リカバリーでは、アーカイブ・ロギングとアクティブ・ロギングの両方を使用して、ロギングの終わりまで、または特定の時点まで、データベースを再構築することができるという点です。アーカイブ・ログ・ファイルは、バックアップを取ったあとに変更内容をリカバリーするために使用されます。これは、バックアップ時までの状態しかリカバリーできず、その後のすべての変更内容が失われる循環ロギングとは異なります。

データベースがアーカイブ・ロギング用に構成されている場合にのみ、オンライン・バックアップを取ることがサポートされます。オンライン・バックアップ操作時には、データベースに対するすべてのアクティビティーがログに記録されます。オンライン・バックアップ・イメージがリストアされるとき、これらのログは少なくともバックアップ操作が完了した時点までロールフォワードされなければなりません。このことが生じるようにするために、ログをアーカイブしておき、データベースがリストアされるときに使用可能にしなければなりません。オンライン・バックアップの完了後に、 DB2 により現行のアクティブ・ログが強制的にクローズされることにより、アーカイブされます。これにより、オンライン・バックアップに、リカバリーに使用できるアーカイブ・ログの完全セットが揃うことになります。



データベースへの変更をたどるためにバックアップ間でログが使用される。

図 7. ロールフォワード・リカバリーでのアクティブ・ログおよびアーカイブ・データベース・ログ： 長時間実行しているトランザクションの場合には、複数のアクティブ・ログが生じる可能性があります。

アーカイブ・ログの保管位置は、*newlogpath* パラメーター、 *logarchmeth1*、および *logarchmeth2* パラメーターのデータベース構成パラメーターによって変更できます。 *newlogpath* パラメーターを変更すると、アクティブ・ログの保管位置も変更されます。

データベース・ログ・パス・ディレクトリーの中のどのログ・エクステントがアーカイブ・ログかを判別するには、 *loghead* データベース構成パラメーターの値を調べてください。このパラメーターには、最も低い番号のアクティブ・ログが示されています。 *loghead* よりも小さいシーケンス番号のログはアーカイブ・ログであり、それらは移動可能なログです。このパラメーターの値は、コントロール・センターを使用してチェックできます。あるいは、コマンド行プロセッサーおよび GET DATABASE CONFIGURATION コマンドを使用して、「最初のアクティブ・ログ・ファイル」を参照します。この構成パラメーターについての詳細は、「*管理ガイド: パフォーマンス*」を参照してください。

関連概念:

- 38 ページの『ログのミラーリング』

関連資料:

- 375 ページの『付録 G. データベース・リカバリー用のユーザー出口』
- 「*管理ガイド: パフォーマンス*」の『loghead - 「最初のアクティブ・ログ・ファイル」構成パラメーター』
- 40 ページの『データベース・ロギングの構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logarchmeth1 - 1 次ログ・アーカイブ方式構成パラメーター』

リカバリー・ログの詳細

ログのミラーリング

DB2® でデータベース・レベルのログのミラーリングがサポートされるようになりました。ログ・ファイルをミラーリングすると、以下の事態からデータベースを保護するのに役立ちます。

- アクティブ・ログの不慮の削除
- ハードウェア障害によるデータ破壊

アクティブ・ログが (ディスクが壊れた結果) 損傷するかもしれないことが心配な場合は、 MIRRORLOGPATH 構成パラメーターを使用し、データベースの 2 次パスを指定してアクティブ・ログのコピーを管理することにより、ログの保管先のボリュームをミラーリングすることを考慮してください。

MIRRORLOGPATH 構成パラメーターを使用すると、データベースは、ログ・ファイルの 2 つ目のコピー (同一の内容) を別のパスに書き込めます。物理的に別個のディスク (別のディスク・コントローラー上でもあることが望ましい) に 2 次ログ・パスを作成することをお勧めします。こうすれば、ディスク・コントローラーが Single Point of Failure になることはありません。

MIRRORLOGPATH を初めて使用可能にした場合は、実際には次のデータベース始動時まで使用されません。これは、NEWLOGPATH 構成パラメーターに似ています。

アクティブ・ログ・パスまたはミラー・ログ・パスへのエラーの書き込みが生じると、データベースにより、障害が起きたパスに「不良」のマークが付けられ、管理

通知ログにメッセージが書き込まれ、以後ログ・レコードを残りの「良好な」ログ・パスだけに書き込まれます。現行のログ・ファイルが完了するまで、DB2 により「不良」パスの使用が再試行されることはありません。 DB2 で次のログ・ファイルをオープンする必要がある場合は、このパスが有効かどうか検査され、有効な場合はそのパスが使い始められます。有効でない場合は、次のログ・ファイルに初めてアクセスするまでの間に、 DB2 によりそのパスの使用が再試行されることはありません。ログ・パスの同期化は試行されませんが、生じたアクセス・エラーに関する情報が DB2 により保持されるので、ログ・ファイルがアーカイブされる際には正しいパスが使用されます。残りの「良好な」パスへの書き込み中に障害が起きると、データベースはシャットダウンします。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『mirrorlogpath - 「ミラー・ログ・パス」構成パラメーター』

NOT LOGGED INITIALLY パラメーターによるロギングの低減

アプリケーションでマスター表から作業表を作成してその中にデータを入れる場合に、マスター表から簡単に再作成できるという理由でこれら作業表のリカバリー可能性について心配する必要がない場合、作業表は CREATE TABLE ステートメントに NOT LOGGED INITIALLY パラメーターを指定して作成することができます。NOT LOGGED INITIALLY パラメーターを使用する利点は、表が作成された作業単位と同じ作業単位で表に対し行われた変更 (挿入、削除、更新、または索引作成操作を含む) をロギングしなくてもよい点です。これにより、実行されるロギングが低減されるだけでなく、アプリケーションのパフォーマンスも向上します。 NOT LOGGED INITIALLY パラメーターを指定した ALTER TABLE ステートメントを使用して、既存の表について同じ結果を確保できます。

注:

1. 同じ作業単位で NOT LOGGED INITIALLY パラメーターを使用し複数の表を作成できます。
2. カタログ表および他のユーザー表に対する変更は、ロギングの対象になります。

表に対する変更がロギングされないため、 NOT LOGGED INITIALLY 表属性の使用を決定するときには次の点を考慮する必要があります。

- 表に対するすべての 変更項目は、コミット時に書き出されます。つまり、コミットには時間がかかります。
- NOT LOGGED INITIALLY 属性がアクティブになっており、ログに記録されないアクティビティーが発生した場合、ステートメントが失敗したり、ROLLBACK TO SAVEPOINT が実行された場合には、作業単位全体はロールバックされます (SQL1476N)。
- ロールフォワード時には、これらの表をリカバリーできません。ロールフォワード操作実行時に NOT LOGGED INITIALLY オプションにより作成または変更された表が検出されると、この表には使用不可のマークが付けられます。データベースのリカバリー後にこの表にアクセスしようとすると、 SQL1477N が戻されます。

注: 表が作成されると、 COMMIT が実行されるまでカタログ表には行ロックが保持されます。ロギングが行われないことを利用するためには、作成される作業単位と同じ作業単位の表にデータを入れる必要があります。これは、並行操作に影響を与えます。

宣言済み一時表によるロギングの低減

宣言済み一時表を作業表として使用することを計画している場合は、次の点に注意してください。

- 宣言済み一時表はカタログでは作成されません。したがって、ロックは保留されません。
- 宣言済み一時表に対しては、最初の COMMIT の後でもロギングは実行されません。
- COMMIT の後に表の行を保持するため、 ON COMMIT PRESERVE オプションを使用してください。これを行わないと、すべての行が削除されます。
- 宣言済み一時表を作成したアプリケーションだけが、その表のインスタンスにアクセスできます。
- データベースのアプリケーション接続がドロップされるときに、宣言済み一時表は暗黙的にドロップされます。
- 宣言済み一時表を使用する作業単位の途中で操作にエラーが生じると、作業単位が完全にはロールバックされなくなります。ただし、宣言済み一時表の内容を変更するステートメントで操作にエラーが生じると、表の行がすべて削除されます。作業単位 (または保管ポイント) のロールバックでは、その作業単位 (または保管ポイント) で変更された宣言済み一時表にあるすべての行が削除されます。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『アプリケーションのプロセス、並行性、およびリカバリー』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『表スペースの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DECLARE GLOBAL TEMPORARY TABLE ステートメント』

データベース・ロギングの構成パラメーター

アーカイブ再試行遅延 (archretrydelay)
前の試行が失敗してからの、次にログ・ファイルをアーカイブしようとするまでに待つ時間間隔 (秒数) を指定します。デフォルト値は 20 です。

ログ・ディスクのブロックが満杯 (blk_log_dsk_ful)
このデータベース構成パラメーターを設定して、 DB2 で新しいログ・ファイルをアクティブ・ログ・パス内に作成できない場合に、ディスク満杯エラーが生成されないようにすることができます。代わりに、正常実行されるまで DB2 は 5 分おきにログ・ファイルの作成を試行します。試行するたびに、DB2 はメッセージを管理通知ログに書き込みます。ログ・ディスクが満杯になったためにアプリケーションが停止したことを確認するには、管理通知ログをモニターするしかありません。ログ・ファイルが正常に作成され

るまでは、表データの更新を試行するユーザー・アプリケーションにより、トランザクションをコミットすることができません。読み取り専用の照会には直接の影響がないこともあります。しかしながら照会で、更新要求によりロックされたデータや、更新アプリケーションによってバッファー・プール中に固定されたデータ・ページにアクセスする必要がある場合は、読み取り専用の照会でも停止状態になります。

blk_log_dsk_ful を YES に設定すると、 DB2 にログ・ディスク満杯エラーが発生する時に、アプリケーションがハングする原因になります。その時エラーを解決し、トランザクションを継続することができます。ディスク満杯の状態は、古いログ・ファイルを別のファイル・システムに移動することにより、またはハングを起こしているアプリケーションが完了できるようにファイル・システムのサイズを増やすことにより解決できます。

blk_log_dsk_ful が NO に設定されている場合には、ログ・ディスク満杯エラーを受け取るトランザクションは失敗し、ロールバックが行われます。トランザクションがログ・ディスク満杯エラーを引き起こす場合には、場合によっては、そのデータベースは停止されます。

フェイルオーバー・アーカイブ・パス (failarchpath)
指定されたログ・アーカイブ・メソッドが失敗する場合の、アーカイブ・ログ・ファイル用の代替ディレクトリーを指定します。このディレクトリーは、失敗したログ・アーカイブ・メソッドがもう一度使用可能になるまでの、ログ・ファイル用の一時ストレージ域です。もう一度使用可能になると、ログ・ファイルは、このディレクトリーからログ・アーカイブ・メソッドに移動します。ログ・ファイルをこの一時ロケーションに移動することにより、ログ・ディレクトリーが満杯になる状態を避けられます。このパラメーターは、完全に修飾された既存のディレクトリーでなければなりません。

ログ・アーカイブ・メソッド 1 (logarchmeth1)、ログ・アーカイブ・メソッド 2 (logarchmeth2)
これらのパラメーターを指定すると、データベース・マネージャーは、ログ・ファイルを、アクティブ・ログ・パスではないロケーションにアーカイブできます。この両方のパラメーターが指定される場合、各ログ・ファイルは 2 回アーカイブされます。つまり、2 つの別個のロケーションに、アーカイブ・ログ・ファイルを 2 つ持つことになります。

これらのパラメーターの有効値には、メディア・タイプや (場合によっては) ターゲット・フィールドが含まれます。値を区切るには、コロン (:) を使用してください。有効な値は以下のとおりです。

OFF ログ・アーカイブ方式が使用されないことを指定します。*logarchmeth1* と *logarchmeth2* の両方が OFF に設定される場合、データベースは、循環ロギングを使用していると見なされ、ロールフォワード・リカバリー可能ではなくなります。これはデフォルトです。

LOGRETAIN
この値は、*logarchmeth1* にのみ使用できるもので、 *logretain* 構成パラメーターを RECOVERY に設定することと同じことです。この値を指定する場合、*logretain* 構成パラメーターが自動的に更新されます。

USEREXIT
この値は、*logarchmeth1* の場合にのみ有効で、 *userexit* 構成パラメーターを ON に設定することと同じことです。この値を指定する場合、*userexit* 構成パラメーターが自動的に更新されます。

DISK この値の後に、コロン (:) を付け、その後に、ログ・ファイルがアーカイブされる完全修飾された既存のパス名を続ける必要があります。たとえば、*logarchmeth1* を DISK:/u/dbuser/archived_logs に設定する場合、アーカイブ・ログ・ファイルは /u/dbuser/archived_logs というディレクトリーに置かれます。

注: テープにアーカイブする予定の場合、db2tapemgr ユーティリティーを使用して、ログ・ファイルを保管および検索できます。

TSM 追加の構成パラメーターなしで指定される場合、この値は、ログ・ファイルはデフォルト管理クラスを使用してローカル TSM サーバーにアーカイブされることを示します。その後にコロン (:) と TSM 管理クラスが続けられる場合、ログ・ファイルは、指定された管理クラスを使用してアーカイブされます。

VENDOR
ベンダー・ライブラリーを使用してログ・ファイルをアーカイブすることを指定します。この値の後に、コロン (:) とライブラリーの名前を続ける必要があります。ライブラリーで提供される API は、ベンダー製品のバックアップおよびリストア API を使用する必要があります。

注:

1. *logarchmeth1* または *logarchmeth2* のいずれかが OFF 以外の値に設定される場合、データベースはロールフォワード・リカバリー用に構成されます。
2. *userexit* または *logretain* 構成パラメーターを更新すると、 *logarchmeth1* は自動的に更新されます (その逆もあります)。しかし、*userexit* または *logretain* のいずれかを使用する場合、 *logarchmeth2* は OFF に設定する必要があります。

ログ・アーカイブ・オプション 1 (logarchopt1)、ログ・アーカイブ・オプション 2 (logarchopt2)
TSM サーバーまたはベンダー API に渡されるストリングを指定します。TSM の場合、このフィールドを使用すると、別の TSM ノード上または別の TSM ユーザーによって生成されたログをデータベースで検索できるようになります。このストリングは、次の形式で指定する必要があります。

```
"-fromnode=nodename -fromowner=ownername"
```

ここで、nodename は、ログ・ファイルを元々アーカイブした TSM ノードの名前で、 ownername は、ログ・ファイルを元々アーカイブした TSM ユーザーの名前です。それぞれのログ・アーカイブ・オプション・フィールドには、対応する 1 つのログ・アーカイブ・メソッドがあります。*logarchopt1* は *logarchmeth1* で使用され、*logarchopt2* は *logarchmeth2* で使用されます。

ログ・バッファー (logbufsz)

このパラメーターには、ログ・レコードをディスクに書き込む前に、ログ・レコード・バッファーとして使用する、メモリーの量を指定します。これらのログ・レコードは、以下のイベントが生じるとディスクに書き込まれます。

- トランザクションのコミット
- ログ・バッファーが満杯
- 他の何らかの内部データベース・マネージャー・イベントの結果として

ログ・バッファーのサイズを大きくすると、ログ・レコードがディスクに書き込まれる頻度が少なくなり、 1 度に書き込まれるログ・レコードの数が多くなるので、ロギングに関連した入出力 (I/O) アクティビティーがさらに効率的になります。

ログ・ファイル・サイズ (logfilsiz)

このパラメーターには、構成する各ログのページ数を指定します。ページのサイズは 4 KB です。

構成できるアクティブ・ログ・スペースの合計の論理的な限界は 256 GB です。この値は、 *logfilsiz* の上限 (262144) と *logprimary* + *logsecond* の上限 (256) の計算結果です。

ログ・ファイルのサイズは、パフォーマンスに直接影響します。あるログから別のログに切り替えると、パフォーマンスが犠牲になります。それで、純粋にパフォーマンスだけを考えると、ログ・ファイルのサイズが大きければ大きいほどよいということになります。このパラメーターはさらに、アーカイブ用のログ・ファイル・サイズも表しています。この場合、ログ・ファイルのサイズが大きければ必ずしもよいというわけではありません。ログ・ファイルのサイズが大きくなると、障害の発生が増加したり、ログ・シッピング・シナリオに遅延が発生するかもしれないからです。アクティブ・ログ・スペースを考慮する時には、小さいログ・ファイルを多く持つ方がよいかもしれません。たとえば、2 つの非常に大きなログ・ファイルがあり、トランザクションが、1 つのログ・ファイルの終わりに近いところで開始した場合、ログ・スペースの半分しか使用できないことになります。

データベースが非活動になる (データベースへのすべての接続が終了する) たびに、現在書き込みが行われているログ・ファイルは、切り捨てられます。それで、データベースが頻繁に非活動になる場合には、DB2 が大きなファイルを作成しても切り捨てられてしまうため、大きなログ・ファイル・サイズを選択しないほうが賢明です。 ACTIVATE DATABASE コマンドを使用して、この損失を避けることができ、バッファー・プールをプライム状態にすることもパフォーマンス向上の助けとなります。

データベースをオープンするための処理時間を最小限にするためにデータベースをオープンしたままにするアプリケーションを使用している場合、ログ・ファイルのサイズはオフライン・アーカイブ・ログのコピーの作成にかかる時間によって決めてください。

ログ・ファイルの損失を最小限にすることも、ログ・サイズを設定する際の重要な考慮事項の 1 つです。アーカイブには、ログ全体が必要です。単一の大きなログを使用する場合は、アーカイブの間隔を長くしてください。ログが入っているメディアで障害が発生した場合は、トランザクション情報が

いくらか失われる可能性があります。ログ・サイズを小さくすれば、アーカイブの頻度は高くなりますが、小さいログを使うため、メディア障害が発生しても失う情報量が少なくなります。

ログ保持 (logretain)

この構成パラメーターは、*logarchmeth1* に置き換えられました。現在は、バックレベル互換性のためにサポートされています。

logretain を RECOVERY に設定すると、アーカイブ・ログがデータベース・ロギング・パス・ディレクトリーに保持されるので、データベースはリカバリー可能と見なされます。つまり、ロールフォワード・リカバリーを使用できるようになります。

注: *logretain* データベース構成パラメーターのデフォルト値は、ロールフォワード・リカバリーをサポートしません。ロールフォワード・リカバリーを使用する予定であれば、このパラメーターの値を変更する必要があります。

トランザクションごとの最大ログ (max_log)

このパラメーターは、1 つのトランザクションで消費できる 1 次ログ・スペースのパーセンテージを示します。

値が 0 に設定される場合、トランザクションで消費できる 1 次ログ・スペースの合計のパーセンテージに制限はありません。アプリケーションが *max_log* 構成に違反する場合、そのアプリケーションは強制的にデータベースから切断され、トランザクションはロールバックされ、そしてエラー SQL1224N が戻されます。

DB2_FORCE_APP_ON_MAX_LOG レジストリー変数を *FALSE* に設定することで、この動作をオーバーライドできます。このようにすると、*max_log* 構成に違反するトランザクションは失敗し、エラー SQL0964N が戻されます。アプリケーションは、作業単位内の前のステートメントで完了した作業を引き続きコミットするか、完了した作業をロールバックして、作業単位を取り消すことができます。

注: 以下の DB2 コマンドは、*max_log* 構成パラメーターによって課された制限から除外されます: ARCHIVE LOG、 BACKUP DATABASE、 LOAD、 REORG TABLE (オンライン)、 RESTORE DATABASE、および ROLLFORWARD DATABASE。

ログ・パスのミラー (mirrorlogpath)

1 次ログ・パスのログをディスク障害または不慮の削除から保護するために、同一のログのセットを 2 次 (ミラー) ログ・パスで保守するように指定することができます。このことを行うためには、この構成パラメーターの値を変更して、別のディレクトリーを指すようにします。データベースの構成が現在ロールフォワード・リカバリー用になっている場合、ミラーリングされたログ・パス・ディレクトリーに保管されるアクティブ・ログは新規の保管位置に移動されません。

ログ・パス・ディレクトリーは変更可能なので、ロールフォワード・リカバリーに必要なログが別のディレクトリーに存在している可能性があります。ロールフォワード操作中に、この構成パラメーターの値を変更して、複数の位置のログにアクセスすることができます。

そのログの位置を記録しておく必要があります。

変更内容は、データベースが整合した状態になるまで適用されません。構成パラメーター *database_consistent* はデータベースの状況を戻します。

この構成パラメーターをオフにするには、その値を DEFAULT にしてください。

注:

1. この構成パラメーターは、1 次ログ・パスがロー・デバイスである場合には、サポートされません。
2. このパラメーターのために指定される値は、ロー・デバイスにはなり得ません。

新規ログ・パス (newlogpath)

データベース・ログは、最初、データベース・ディレクトリーのサブディレクトリー SQLOGDIR の中に作成されます。アクティブ・ログと将来のアーカイブ・ログを保管する位置を変更するには、この構成パラメーターの値を変更して別のディレクトリーか装置を指示するようにします。データベースの構成が現在ロールフォワード・リカバリー用になっている場合、データベース・ログ・パス・ディレクトリーに保管されるアクティブ・ログは新規の保管位置に移動されません。

ログ・パス・ディレクトリーは変更可能なので、ロールフォワード・リカバリーに必要なログが別のディレクトリーや別の装置に存在している可能性があります。ロールフォワード操作中に、この構成パラメーターの値を変更して、複数の位置のログにアクセスすることができます。

そのログの位置を記録しておく必要があります。

変更内容は、データベースが整合した状態になるまで適用されません。構成パラメーター *database_consistent* はデータベースの状況を戻します。

グループへのコミット回数 (mincommit)

このパラメーターを使うと、最低限の回数のコミットが実行されるまで、ログ・レコードのディスク書き込みを遅らせることができるようになります。この遅延により、ログ・レコード書き込みに関連するデータベース・マネージャーのオーバーヘッドが少なくなるため、データベースに対して複数のアプリケーションを実行しており、それらのアプリケーションによって非常に短い期間内に多数のコミットが要求される場合のパフォーマンスを向上させることができます。

コミットのグループ化が行われるのは、このパラメーターの値が 1 より大きく、データベースに接続されているアプリケーションの数がこのパラメーターの値より大きい場合だけです。コミットのグループ化が有効な場合、アプリケーションのコミット要求は、1 秒経過するか、またはコミット要求回数がこのパラメーター値に達するまで保留されます。

エラー時のアーカイブ再試行回数 (numarchretry)

ログ・ファイルが *failarchpath* 構成パラメーターで指定されたパスにアーカイブされる前に、指定したログ・アーカイブ方式を使用したログ・ファイルのアーカイブを試行する回数を指定します。このパラメーターは、*failarchpath* 構成パラメーターが設定される場合にのみ使用できます。デフォルト値は 5 です。

ログ・スパン数 (num_log_span)

このパラメーターは、アクティブな 1 つのトランザクションで扱える、アクティブなログ・ファイルの数を示します。値が 0 に設定される場合、1 つのトランザクションで扱えるログ・ファイルの数に制限はありません。

アプリケーションが *num_log_span* 構成に違反する場合、そのアプリケーションは強制的にデータベースから切断され、エラー SQL1224N が戻されます。

注: 以下の DB2 コマンドは、*num_log_span* 構成パラメーターによって課された制限から除外されます: ARCHIVE LOG、 BACKUP DATABASE、LOAD、 REORG TABLE (オンライン)、 RESTORE DATABASE、および ROLLFORWARD DATABASE。

オーバーフロー・ログ・パス (overflowlogpath)

このパラメーターは、実際のロギング要件により、いくつかの機能で使用されます。 DB2 がロールフォワード操作に必要なログ・ファイルを検索するために、場所を指定できます。それは、ROLLFORWARD コマンドの OVERFLOW LOG PATH オプションに似ていますが、発行されるすべての ROLLFORWARD コマンドに OVERFLOW LOG PATH オプションを指定する代わりに、この構成パラメーターを 1 度だけ設定します。もし両方とも使用される場合には、そのロールフォワード操作のために OVERFLOW LOG PATH オプションが *overflowlogpath* 構成パラメーターを上書きします。

logsecond が -1 に設定されている場合には、 DB2 が、アーカイブから検索してきたアクティブ・ログ・ファイルを保管するためにディレクトリーを指定します。 (アクティブ・ログ・ファイルは、それらがもうアクティブ・ログ・パスにない場合には、ロールバック操作のために検索する必要があります。)

overflowlogpath が指定されていない場合には、 DB2 はログ・ファイルを検索してアクティブ・ログ・パスに入れます。このパラメーターを指定することにより、検索したログ・ファイルを DB2 が保管するために、付加的なリソースを提供できます。この利点には、入出力の損失を別のディスクに分散できること、さらにより多くのログ・ファイルをアクティブ・ログ・パスに保管できるようになることが含まれます。

たとえば、複製のために **db2ReadLog** API を使用している場合、*overflowlogpath* を使用して、DB2 がこの API に必要なログ・ファイルを検索するための場所を指定できます。ログ・ファイルが (アクティブ・ログ・パスまたはオーバーフロー・ログ・パスのいずれかで) 検出できず、データベースが *userexit* を使用可能にして構成されている場合には、DB2 はログ・ファイルを検索します。さらに、このパラメーターを使用して、検索したログ・ファイルを DB2 が保管するためにディレクトリーを指定できます。この利点は、アクティブ・ログ・パスの入出力の損失を削減することと、より多くのログ・ファイルをアクティブ・ログ・パスに保管できるようになることによりもたらされます。

アクティブ・ログ・パスのためにロー・デバイスを構成済みの場合、*logsecond* を -1 に設定したい場合や、 **db2ReadLog** API を使用したい場合には、*overflowlogpath* を構成する必要があります。

overflowlogpath を設定するためには、最大 242 バイトまでのストリングを指定してください。そのストリングは、パス名を指している必要があり、それは相対パス名ではなく、完全修飾パス名でなければなりません。パス名は、ロー・デバイスではなく、ディレクトリーでなければなりません。

注: パーティション・データベース環境では、ノード番号は自動的にパスに追加されます。このことは、複数論理ノード構成のパスの固有性を維持するために行われます。

1 次ログ (logprimary)

このパラメーターには、作成するサイズ *logfilsz* の 1 次ログの個数を指定します。

空の場合でも満杯の場合でも、1 次ログには同じ容量のディスク・スペースが必要です。それで、必要以上に多数のログを構成すると、不必要にディスク・スペースを使用することになります。構成するログ数が少なすぎると、ログ満杯条件になる可能性があります。構成するログの個数を選択する際は、それぞれのログのサイズと、アプリケーションでログ満杯条件を処理できるかどうかを考慮する必要があります。アクティブ・ログ・スペースのログ・ファイルの合計サイズの限界は 256 GB です。

既存データベースでロールフォワード・リカバリーを使用可能にしている場合は、1 次ログの数を、 1 次ログと 2 次ログの合計数に 1 を足した数に変更する必要があります。ロールフォワード・リカバリーが可能になっているデータベースでは、 LONG VARCHAR フィールドと LOB フィールドについて追加情報がログに記録されます。

2 次ログ (logsecond)

このパラメーターは、必要に応じてリカバリー用に作成されて使用される 2 次ログ・ファイルの数を指定します。

1 次ログ・ファイルが満杯になると、 *logfilsiz* で指定されたサイズの 2 次ログ・ファイルが必要に応じて 1 度に 1 つずつ割り振られます。このパラメーターは、その最大数を指定します。このパラメーターが -1 に設定されている場合には、データベースは無限のアクティブ・ログ・スペースで構成されます。データベース上で実行中の不完了トランザクションのサイズまたは数には、制限がありません。

注:

1. ログ・アーカイブは、*logsecond* を -1 に設定するために、使用可能になっている必要があります。
2. このパラメーターが -1 に設定されている場合には、 DB2 がアーカイブ・ログ・ファイルを検索する必要があるかもしれないので、クラッシュ・リカバリー時間を増やします。

ユーザー出口 (userexit)

この構成パラメーターは、*logarchmeth1* に置き換えられました。現在は、バックレベル互換性のためにサポートされています。

このパラメーターを使うと、データベース・マネージャーでログのアーカイブと検索のためのユーザー出口プログラムを呼び出せるようになります。ロ

グ・ファイルはアクティブ・ログ・パスでない場所にアーカイブされます。*userexit* を ON に設定すると、ロールフォワード・リカバリーを使用できるようになります。

オフライン・アーカイブ・ログを保管するのに使用している装置のデータ転送速度、およびコピー作成に使用しているソフトウェアのデータ転送速度は、データベース・マネージャーがログに書き込むときの平均速度以上であることが必要です。転送速度が新規ログ・データの生成速度に追いつかない場合、ロギング・アクティビティーがかなり長い時間続行されると、ディスク・スペースを使い切ってしまうおそれがあります。ディスク・スペースを使い切ってしまう時間の長さは、ディスク・スペースの空き容量によって決まります。これが起こると、データベース処理が停止してしまいます。

テープ装置または光メディアを使用している場合には、データ転送速度が最重要になります。テープ装置の中には、ファイルのコピーに、サイズに関係なく同じ長さの時間がかかるものがあります。アーカイブ装置の能力について調べておく必要があります。

テープ装置には、このほかにも考慮事項があります。アーカイブ要求の頻度も重要です。たとえば、コピー操作の完了に要する時間が 5 分間であるとすると、ログは、ワークロードがピークの時の 5 分間分のログ・データを保持できるだけの十分な大きさになっている必要があります。テープ装置には、1 日あたりの操作回数に設計上の限界がある場合があります。ログ・サイズを決定するときは、これらの要因を考慮するようにしてください。

注:

1. この値を ON に設定して、無限アクティブ・ログ・スペースを使用可能にする必要があります。
2. *userexit* データベース構成パラメーターのデフォルト値は、ロールフォワード・リカバリーをサポートしませんので、それを使用する場合には変更する必要があります。

関連概念:

- 49 ページの『ログ・ファイルの管理』
- 66 ページの『リカバリー・パフォーマンスの向上』

関連資料:

- 375 ページの『付録 G. データベース・リカバリー用のユーザー出口』
- 「*管理ガイド: パフォーマンス*」の『logfilsiz - 「ログ・ファイルのサイズ」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logprimary - 「1 次ログ・ファイル数」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logsecond - 「2 次ログ・ファイル数」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logbufsz - 「ログ・バッファー・サイズ」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『mincommit - 「グループへのコミット数」構成パラメーター』

- 「*管理ガイド: パフォーマンス*」の『newlogpath - 「データベース・ログ・パスの変更」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logretain - 「ログ保存使用可能」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『userexit - 「ユーザー出口使用可能」構成パラメーター』
- 381 ページの『Storage Manager に対するバックアップおよびリストアの API』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 「*管理ガイド: パフォーマンス*」の『mirrorlogpath - 「ミラー・ログ・パス」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『blk_log_dsk_ful - 「ログディスクフルによるアプリケーション中断」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『overflowlogpath - 「オーバーフロー・ログ・パス」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『max_log - トランザクション当たりの最大ログ構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『num_log_span - 番号ログ幅構成パラメーター』
- 「コマンド・リファレンス」の『db2tapemgr - Manage Log Files on Tape コマンド』

ログ・ファイルの管理

データベース・ロギングを管理するときは、以下の項目を考慮に入れてください。

- アーカイブ・ログの番号付けスキーマは、S0000000.LOG から始まり、S9999999.LOG まで (最大 10000000 個のログ・ファイルまで対応できる) になります。データベース・マネージャーは、以下の条件で S0000000.LOG にリセットされます。
  - データベース構成ファイルがロールフォワード・リカバリー可能に変更された場合
  - データベース構成ファイルがロールフォワード・リカバリー*不可* に変更された場合
  - S9999999.LOG が使用された場合

  DB2® では、データベース・リストア後、ロールフォワード・リカバリーを実行してもしなくても、ログ・ファイル名を再使用します。データベース・マネージャーでは、ロールフォワード・リカバリー時に誤ったログが適用されないようになっています。リストア操作後に DB2 がログ・ファイル名を再使用する場合、同じ名前の複数のログ・ファイルをアーカイブできるように、新しいログ・ファイルは個別のディレクトリーにアーカイブされます。ログ・ファイルのロケーションは、ロールフォワード・リカバリー時に適用できるように、リカバリー履歴ファイルに記録されます。ロールフォワード・リカバリー時には、必ず正しいログが使用可能になっているようにする必要があります。

ロールフォワード操作が正常完了すると、使用された最後のログは切り捨てられ、その次の順次ログからロギングが開始されます。ログ・パス・ディレクトリー中のログのうち、ロールフォワード・リカバリーに使用された最後のログよりシーケンス番号が大きいログが再使用されます。切り捨てが行われたログ内の、切り捨て位置より後の項目は、ゼロで上書きされます。ロールフォワード・ユーティリティーを呼び出す前に、ログのコピーを必ず作成してください。 (ユーザー出口プログラムを呼び出して、ログを別の位置にコピーすることもできます。)

- データベースが (ACTIVATE DATABASE コマンドによって) アクティブにされていない場合は、すべてのアプリケーションがそのデータベースから切断されると、 DB2 により現行のログ・ファイルが切り捨てられます。その後アプリケーションがデータベースに接続されると、 DB2 により新しいログ・ファイルへのロギングが開始されます。システム上に小さなログ・ファイルが多数作成されるようであれば、 ACTIVATE DATABASE コマンドを使用することも考慮できます。このコマンドを使用すると、アプリケーションの接続時にデータベースを初期設定する必要が生じることによるオーバーヘッドを節約できるだけでなく、大規模なログ・ファイルを割り振って切り捨ててからまた新しい大規模ログ・ファイルを割り振ることによるオーバーヘッドも節約できます。
- ログ名が再使用されてしまうため、アーカイブ・ログは 1 つのデータベースの 2 つ以上の異なるログ・シーケンス に関連している可能性があります (51 ページの図 8 を参照)。たとえば、Backup 2 をリカバリーしたい場合、使用できるログ・シーケンスは 2 通りあります。全データベースのリカバリー時に、特定の時点までロールフォワード操作を実行し、ログの終わりまで達しないうちに処理を停止すると、新しいログ・シーケンスが作成されます。 2 つのログ・シーケンスを結合することはできません。オンライン・バックアップ・イメージが最初のログ・シーケンスの全体にわたっている場合は、ロールフォワード操作を完了するのにこのログ・シーケンスを使用する必要があります。

  リカバリー後に新規のログ・シーケンスを作成した場合は、古いログ・シーケンスの表スペースのバックアップ・イメージはすべて無効になります。このことは通常リストア時に認識されますが、データベースのリストア操作の直後に表スペースのリストア操作を行うと、リストア・ユーティリティーは古いログ・シーケンスの表スペースのバックアップ・イメージを認識できません。データベースが実際にロールフォワードされるまで、使用されるログ・シーケンスは不明です。表スペースが古いログ・シーケンスにある場合は、表スペースのロールフォワード操作によってそれを「収集」しなければなりません。無効なバックアップ・イメージを使用してリストア操作を行うと、正常に完了することもありますが、その表スペースの表スペース・ロールフォワード操作は失敗し、表スペースはリストア・ペンディング状態のままになります。

  たとえば、表スペース・レベルのバックアップ操作 Backup 3 が上部のログ・シーケンスの S0000013.LOG と S0000014.LOG の間で完了するとします (51 ページの図 8 を参照)。データベース・レベルのバックアップ・イメージ Backup 2 を使用してリストアおよびロールフォワードを実行したい場合は、 S0000012.LOG を使用してロールフォワードする必要があります。その後、上部のログ・シーケンスか (新規の) 下部のログ・シーケンスで、ロールフォワードを継続できます。下部のログ・シーケンスでロールフォワードを行う場合は、表スペース・レベルのバックアップ・イメージ Backup 3 を使って表スペースのリストアとロールフォワード・リカバリーを行うことはできません。

表スペース・レベルのバックアップ・イメージ Backup 3 を使ってログ終了時に表スペースのロールフォワード操作を完了するには、データベース・レベルのバックアップ・イメージ Backup 2 をリストアしてから、上部のログ・シーケンスを使ってロールフォワードする必要があります。表スペース・レベルのバックアップ・イメージ Backup 3 のリストアが終了すると、ログ終了時のロールフォワード操作を開始できるようになります。

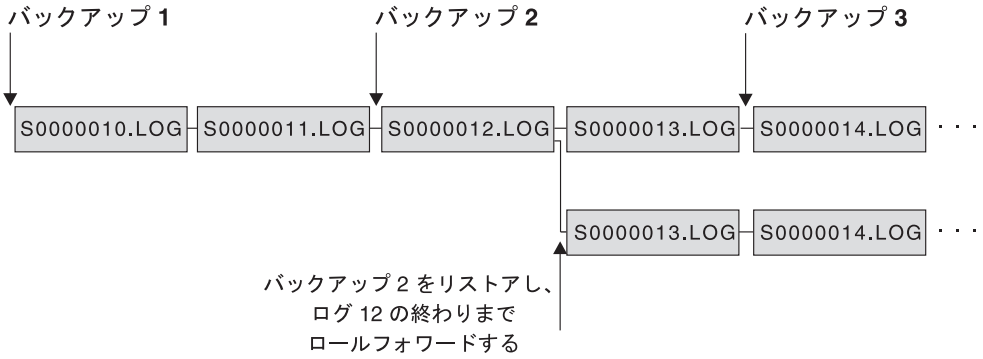


図 8. ログ・ファイル名の再使用

関連資料:

- 375 ページの『付録 G. データベース・リカバリー用のユーザー出口』

ログ・ファイルの割り振りと除去

データベース・ログ・ディレクトリー内のログ・ファイルは、クラッシュ・リカバリーに必要になる可能性がある場合には、決して除去されません。しかし、無限のロギングを使用可能にした場合、ログ・ファイルは、正常にアーカイブされると削除されます。 *logarchmeth1* データベース構成パラメーターが OFF に設定されない場合、満杯のログ・ファイルは、クラッシュ・リカバリーにもはや必要ではなくなってからのみ、除去の候補になります。クラッシュ・リカバリーに必要なログ・ファイルは、アクティブ・ログと呼ばれます。クラッシュ・リカバリーに必要ではないログ・ファイルは、アーカイブ・ログと呼ばれます。

新規ログ・ファイルを割り振り、古いログ・ファイルを除去するプロセスは、*logarchmeth1* データベース構成パラメーターの設定に従属しています。

Logarchmeth1 **および** *Logarchmeth2* **が、OFF に設定されている場合**

循環ロギングが使用されます。循環ロギングでは、クラッシュ・リカバリーはサポートされますが、ロールフォワード・リカバリーはサポートされません。

循環ロギング中、2 次ログ以外の新規ログ・ファイルは生成されず、古いログ・ファイルは削除されません。ログ・ファイルは、循環する仕方で処理されます。すなわち、最後のログ・ファイルが満杯の時、DB2® は最初のログ・ファイルに書き込み始めます。

すべてのログ・ファイルがアクティブで、循環ロギング・プロセスが最初のログ・ファイルに折り返せなかった場合に、ログ満杯の状態が発生する可能性があります。 2 次ログ・ファイルは、すべての 1 次ログ・ファイルがアクティブで満杯の時に作成されます。 2 次ログ・ファイルは、データベー

スが非アクティブ化される場合や、使用しているスペースがアクティブ・ログ・ファイル用に必要な場合に削除されます。

Logarchmeth1 が **LOGRETAIN** に設定される
: アーカイブ・ロギングは、使用されます。データベースは、リカバリー可能です。ロールフォワード・リカバリーおよびクラッシュ・リカバリーの両方は、使用可能になります。ログ・ファイルをアーカイブしたら、アクティブ・ログ・パスからログ・ファイルを削除して、そのディスク・スペースを新しいログ・ファイル向けに再使用できるようにする必要があります。ログ・ファイルが満杯になるたびに DB2 は別のログ・ファイルに書き込み始め、 (1 次および 2 次ログの最大数に達していない場合には) 新規ログ・ファイルを作成します。

Logarchmeth1 が **OFF** または **LOGRETAIN** 以外の値に設定される
: アーカイブ・ロギングは、使用されます。データベースは、リカバリー可能です。ロールフォワード・リカバリーおよびクラッシュ・リカバリーの両方は、使用可能になります。ログ・ファイルが満杯になると、それはユーザーが提供するユーザー出口プログラムを使用して自動的にアーカイブされます。

  通常ログ・ファイルは削除されません。むしろ、新しいログ・ファイルが必要になり、利用できるログ・ファイルがない場合には、アーカイブ・ログ・ファイルがリネームされ、再使用されます。アーカイブ・ログ・ファイルは、それが閉じてログ・アーカイブ・ディレクトリーにコピーされた後は、削除もリネームもされません。 DB2 は新規ログ・ファイルが必要になるまで待った後、最も古いアーカイブ・ログをリネームします。リカバリー中にデータベース・ディレクトリーに移動したログ・ファイルは、それがもはや必要ではなくなった時、リカバリー処理中に除去されます。 DB2 がログ・スペースを使い尽くすまで、古いログ・ファイルがデータベース・ディレクトリーに残っています。

  ログ・ファイルのアーカイブ時にエラーが発生する場合、アーカイブ作業は、 ARCHRETRYDELAY データベース構成パラメーターで指定した時間だけ中断されます。さらに、NUMARCHRETRY データベース構成パラメーターを使用して、DB2 が、ログ・ファイルを (FAILARCHPATH データベース構成パラメーターで指定した) フェイルオーバー・ディレクトリーへアーカイブしようとする前に、そのログ・ファイルを 1 次または 2 次アーカイブ・ディレクトリーへアーカイブしようとする回数も指定できます。 NUMARCHRETRY は、FAILARCHPATH データベース構成パラメーターが設定される場合にのみ使用されます。 NUMARCHRETRY が 0 に設定される場合、DB2 は引き続き 1 次または 2 次ログ・パスからのアーカイブを再試行します。

  古いログ・ファイルを除去する最も簡単な方法はデータベースを再始動することです。データベースが再始動すると、新規ログ・ファイルおよびユーザー出口プログラムがアーカイブに失敗したログ・ファイルのみがデータベース・ディレクトリーに検出されます。

  データベースが再始動すると、データベース・ログ・ディレクトリーにあるログの最小の数は、 *logprimary* データベース構成パラメーターを使用して構成できる 1 次ログの数と等しくなります。 1 次ログの数以上のものがログ・ディレクトリーに検出されることもありえます。これは、データベース

がシャットダウンした時点でのログ・ディレクトリー内の空のログの数が、データベースが再始動した時点での *logprimary* 構成パラメーターの値より大きい場合に起こりえます。これは、*logprimary* 構成パラメーターが、シャットダウンするデータベースと再始動するデータベースとの間で変更された場合、または 2 次ログが割り振られ、決して使用されない場合に発生します。

データベースが再始動する時、空のログの数が *logprimary* 構成パラメーターで指定された 1 次ログの数より小さい場合には、その差を埋め合わせるために余分のログ・ファイルが割り振られます。データベース・ディレクトリーで使用可能な 1 次ログ以上に空のログがある場合には、そのデータベースは、データベース・ディレクトリー内で検出される使用可能な空のログをその数の分だけ使用して再始動できます。データベースのシャットダウンの後、作成された 2 次ログ・ファイルは、データベースが再始動する時にアクティブ・ログ・パスに残ります。

ログ・アーカイブを使用したログ・ファイルの管理

以下の考慮事項が、ログ・ファイルのアーカイブと検索のためのユーザー出口プログラムの呼び出しに適用されます。

- データベース構成パラメーター *logarchmeth1* を USEREXIT に設定することで、データベース・マネージャーがユーザー出口プログラムを呼び出して、ファイルをアーカイブするか、またはデータベースのロールフォワード・リカバリー時にログ・ファイルを検索するかを指定します。ログ・ファイルの検索要求は、ロールフォワード・ユーティリティーで、ログ・パス・ディレクトリーにないログ・ファイルが必要となるときに行われます。

  注: Windows® オペレーティング・システムの場合、 REXX ユーザー出口を使用してログをアーカイブすることはできません。

- アーカイブの際、ログ・ファイルがまだアクティブで通常の処理に必要な場合でも、ログ・ファイルが満杯になると、ユーザー出口に渡されます。このことにより、揮発性メディアからできるだけ速くデータのコピーを移動させることができます。ユーザー出口に渡されたログ・ファイルは、通常の処理に必要でなくなるまで、ログ・パス・ディレクトリーに保持されます。必要なくなった時点で、ディスク・スペースは再使用されます。
- DB2® は、ユーザー出口プログラムが開始してログ・ファイルをアーカイブすると、読み取りモードでこのファイルをオープンします。このため、プラットフォームによっては、ユーザー出口プログラムでログ・ファイルを削除することができません。 AIX® など、ユーザー出口プログラムを含むプロセスでログ・ファイルを削除できるプラットフォームもあります。ログ・ファイルはアーカイブ後も依然としてアクティブで、クラッシュ・リカバリーに必要なので、このようなファイルを決してユーザー出口プログラムで削除してはなりません。ログ・ファイルがアーカイブされると、DB2 によりディスク・スペースの再使用が管理されます。
- ログ・ファイルがアーカイブされ、そこにオープン・トランザクションが含まれない場合、 DB2 はファイルを削除しませんが、そのファイルが必要とされるとき、次のログ・ファイルとして名前を変更します。 (ファイルの名前を変更する代わりに) 新しいログ・ファイルを作成すると、ディスク・スペースを保証する

ためにすべてのページを書き出すことになるので、これはパフォーマンスの向上になります。ディスク上の必要なページを再使用する方が、それを解放して再取得するよりも効率的です。

- DB2 は、*logsecond* データベース構成パラメーターが -1 に設定されていなければ、クラッシュ・リカバリーまたはロールバックの際に、ログ・ファイルを検索*しません*。
- ログ・アーカイブを構成しても、障害の時点までのロールフォワード・リカバリーは保証されませんが、障害期間を短くすることが試行されます。ログ・ファイルが満杯になると、ユーザー出口ルーチン用にキューに入れられます。ログ・ファイルが満杯になる前にログを含むディスクに障害が起きた場合は、そのログ・ファイル内のデータは失われます。また、ファイルはアーカイブ用にキューに入れられるので、すべてのファイルがコピーされる前にディスクに障害が起こり、キュー内のすべてのログ・ファイルが失われることがあります。
- 個々のログ・ファイルの構成サイズは、ログ・アーカイブに直接影響します。個々のログ・ファイルが非常に大きい場合、ディスクに障害が起こると、大量のデータが失われることがあります。小さいログ・ファイルで構成されたデータベースは、より頻繁にデータをユーザー出口ルーチンに渡させます。

  しかし、テープなど比較的低速の装置にデータを移動している場合は、比較的大きいサイズのログ・ファイルを使用して、キューが作成されないようにすることもできます。各ファイルのアーカイブで、テープ装置の巻き戻しやアーカイブ・メディアへの接続の確立など、実際のオーバーヘッドが必要になる場合にも、大きいログ・ファイルを使用することをお勧めします。
- *logarchmeth1* が USEREXIT に設定される場合、ユーザー出口プログラムに対するアーカイブ要求は、アクティブ・ログ・ファイルが満杯になるたびに発生します。データベースからの最後の切断が生じたときにアーカイブ・ログ・ファイルが満杯になっていないことがあります。また、満杯になっていないアクティブ・ログ・ファイルに対してユーザー出口プログラムが呼ばれることもあります。

  注: 未使用のログ・スペースを解放するために、ログ・ファイルは未使用の部分が切り捨てられてからアーカイブされます。
- ログ・ファイルを含む装置にメディア障害が生じた場合に、ロールフォワード・リカバリーでオフライン・ログ・ファイルを使用できるように、別の物理装置に対してログのコピーが行われる必要があります。この装置は、データベース・データ・ファイルのある装置と同じであってはなりません。
- ユーザー出口プログラムを使用可能にし、ログおよびバックアップ・イメージ用のストレージ・デバイスとして磁気テープ・ドライブを使用している場合には、バックアップ・イメージのあて先と、アーカイブ・ログのあて先が同一の磁気テープ・ドライブにないことを確認する必要があります。いくつかのログのアーカイブは、バックアップ操作の進行中に行われる可能性があるため、 2 つのプロセスが同時に同じ磁気テープ・ドライブに書き込もうとすると、エラーが起きるかもしれません。
- 以下を使用している場合には、ユーザー出口プログラムを使用してログ・ファイルを管理するときに、ローカル・アタッチされたテープ・ドライブを使用しないでください。
  - 無限ロギング
  - オンライン表スペース・レベル・リカバリー

  - レプリケーション
  - ログの非同期読み取り API (db2ReadLog)
  - 高可用性災害時リカバリー (HADR)
- 場合によっては、アーカイブ要求に対する肯定応答をユーザー出口プログラムから受け取る前にデータベースがクローズすると、データベース・マネージャーはデータベースがオープンするときに別の要求を送信します。したがって、ログ・ファイルが複数回アーカイブされることがあります。
- ユーザー出口プログラムが、存在しないファイルのアーカイブ要求を受け取ったり (アーカイブ要求が複数あり、最初のアーカイブ操作が正常実行された後にそのファイルが削除されたため)、または、存在しないファイルの探索要求を受け取ったり (別のディレクトリーにあるか、またはログの末尾に達したため) する場合、この要求を無視して正常な戻りコードを渡す必要があります。
- ユーザー出口プログラムは、特定の時点でリカバリーした後で、同じ名前の別のログ・ファイルが存在できるようにする必要があります。その両方のログ・ファイルを保存し、正しいリカバリー・パスと関連付けるように作成する必要があります。
- ログ・ファイルをアーカイブするために同一の磁気テープ装置を使用している複数のデータベース用に、ユーザー出口プログラムが使用可能になっており、そのデータベースの 1 つでロールフォワード操作が行われている場合、それ以外のデータベースをアクティブにしてはいけません。ロールフォワード操作が進行中に別のデータベースがログ・ファイルをアーカイブしようとした場合、ロールフォワード操作に必要なログが見つからなかったり、磁気テープ装置にアーカイブされた新しいログ・ファイルが、その磁気テープ装置に以前に保管したログ・ファイルを上書きしてしまう可能性があります。

  上記のいずれかの状態が生じないようにするために、ユーザー出口プログラムを呼び出すノード上の他のデータベースが、ロールフォワード操作中にオープンしていないようにするか、またはこの状態を処理するようにユーザー出口プログラムを作成することができます。

関連概念:

- 49 ページの『ログ・ファイルの管理』

ログ・ディレクトリー・ファイルが満杯の場合のトランザクションのブロック化

blk_log_dsk_ful データベース構成パラメーターを設定して、 DB2® で新しいログ・ファイルをアクティブ・ログ・パス内に作成できない場合に「ディスク満杯」エラーが生成されないようにすることができます。

代わりに、正常実行されるまで DB2 は 5 分おきにログ・ファイルの作成を試行します。ログ・アーカイブ・メソッドが指定される場合、DB2 によりログ・ファイルのアーカイブが完了したかチェックされます。アーカイブ・ログ・ファイルが正常にアーカイブされたら、非アクティブのログ・ファイルの名前を新しいログ・ファイル名に変更して処理を継続できます。試行するたびに、DB2 はメッセージを管理通知ログに書き込みます。ログ・ディスクが満杯になったためにアプリケーションが停止したことを確認するには、管理通知ログをモニターするしかありません。

ログ・ファイルが正常に作成されるまでは、表データの更新を試行するユーザー・アプリケーションにより、トランザクションをコミットすることができません。読み取り専用の照会には直接の影響がないこともあります。しかしながら照会で、更新要求によりロックされたデータや、更新アプリケーションによってバッファー・プール中に固定されたデータ・ページにアクセスする必要がある場合は、読み取り専用の照会でも停止状態になります。

関連概念:

- 36 ページの『リカバリー・ログについて』
- 53 ページの『ログ・アーカイブを使用したログ・ファイルの管理』

オンデマンドのログのアーカイブ

DB2® では、いつでもリカバリー可能データベースのアクティブ・ログをクローズすることができます (さらに、使用できる場合はアーカイブもできます)。このサポートにより、認識されている時点までのログ・ファイルの完全セットを収集し、これらのログ・ファイルを使用してスタンバイ・データベースを更新できます。

オンデマンドのログのアーカイブを開始するには、ARCHIVE LOG コマンドを呼び出すか、または **db2ArchiveLog** API を呼び出します。

関連概念:

- 53 ページの『ログ・アーカイブを使用したログ・ファイルの管理』

関連資料:

- 295 ページの『ARCHIVE LOG』
- 40 ページの『データベース・ロギングの構成パラメーター』
- 307 ページの『db2ArchiveLog - アクティブ・ログのアーカイブ』

ロー・ログの使用

データベース・ログにロー・デバイスを使用することができます。このようにすることには、利点も欠点もあります。

- この利点は次のとおりです。
  - 1 つのシステムに 26 以上の物理ドライブをアタッチできる。
  - ファイル I/O パスの長さが短い。このため、システムでのパフォーマンスが向上します。ワークロードを軽減するだけの実質的な益があるかどうかを評価するために、ベンチマークを実行する必要があります。
- 欠点は次のとおりです。
  - 他のアプリケーションが装置を共有できない。つまり、装置全体を DB2® に割り当てなければならない。
  - その装置からバックアップまたはコピーを行うオペレーティング・システムまたは他社製のツールが、装置を操作できない。
  - 間違った物理ドライブ名を指定すると、既存のドライブにあるファイル・システムを簡単に削除できてしまう。

ロー・ログの構成には、*newlogpath* データベース構成パラメーターを使用できます。ただし、そのようにする前に、上記の利点と欠点、および下記の追加の考慮事項を考慮してください。

- 1 つの装置だけが許可されます。オペレーティング・システム・レベルで複数のディスクに対して装置を定義できます。 DB2 はオペレーティング・システム呼び出しを行って、 4 KB ページ内で装置のサイズを決定します。

  複数のディスクを使用する場合は、これによってより大きな装置が提供され、結果として生じるストライピングは、より速い入出力スループットによってパフォーマンスを向上させます。
- DB2 は、装置の最後の 4 KB ページに書き込みを行おうとします。装置のサイズが 2 GB を超える場合、 2 GB 以上の装置をサポートしていないオペレーティング・システムでは、最後のページへの書き込みが失敗します。この場合、DB2 はサポートできる限界のすべてのページを使用しようとします。

  装置のサイズに関する情報を使えば、オペレーティング・システムのサポートによって DB2 で使用できる装置のサイズ (4 KB ページ内) を指定できます。DB2 が書き込むことのできるディスク・スペースの量は、*device-size-available* として参照されます。

  DB2 は装置の最初の 4 KB ページを使用しません。 (このスペースは一般にオペレーティング・システムが他の目的で使用します。) そのため、 DB2 で利用できるスペースの合計は、*装置サイズ = 使用可能な装置サイズ - 1* です。
- *logsecond* パラメーターは使用しません。 DB2 は 2 次ログの割り振りを行いません。アクティブ・ログ・スペースのサイズは、 *logprimary* × *logfilsiz* で算出される 4 KB ページの数です。
- ログ・レコードは引き続きログ・エクステントにグループ化され、各レコードのログ・ファイル・サイズ (*logfilsiz*) は 4 KB ページになります。ログ・エクステントは 1 つずつロー・デバイスに入れられます。各エクステントには、エクステント・ヘッダーに使用する余分の 2 ページも入っています。つまり、装置がサポートできる*使用可能なログ・エクステント数* は、*装置サイズ* / (*logfilsiz* + 2) となります。
- 装置には、アクティブ・ログ・スペースをサポートできるだけの十分な容量が求められます。すなわち、*使用可能なログ・エクステント数* は、 *logprimary* 構成パラメーターに使用した値よりも大きい (またはそれに等しい) 値でなければなりません。 (*logarchmeth1* および *logarchmeth2* 構成パラメーターを使用して) ログ・アーカイブが使用可能になっている場合、ロー・デバイスが、*logprimary* 構成パラメーターで指定された値より多くのログを保管ができることを確認してください。このことにより、ログ・ファイルがアーカイブされる時に発生する遅延が埋め合わされます。
- 循環ロギングを使用している場合は、*logprimary* 構成パラメーターが、装置に書き込まれるログ・エクステントの数を決定します。これは、装置に未使用スペースを生じさせることがあります。
- *logarchmeth1* が LOGRETAIN に設定される場合、*使用可能なログ・エクステント数* が使い果たされると、結果として更新される操作はすべて、ログ満杯エラーを受け取ります。この時点で、データベースをシャットダウンして、オフライン・バックアップをとり、リカバリー性を保全します。データベースのバックアップ操

作後、装置に書き込まれたログ・レコードは失われます。これは、データベースのリストアに以前のデータベース・バックアップ・イメージを使用できないという意味なので、データベースをロールフォワードします。*使用可能なログ・エクステント数* が使い果たされる前に、データベースのバックアップを取ると、データベースをリストアしてロールフォワードできます。

- *logarchmeth1* が USEREXIT に設定される場合、ユーザー出口プログラムは、装置を読み取り、アーカイブ・ログをファイルとして保管できなければなりません。
- DB2 は、ロー・デバイスからログ・ファイルを検索しません。しかし、ログ・ファイルが必要なときに、DB2 はエクステント・ヘッダーを読み取って、必要なログ・ファイルがロー・デバイスに入っているかどうかを判別します。必要なログ・ファイルがロー・デバイスに入っていなければ、オーバーフロー・ログ・パスを検索します。それでもログ・ファイルが見つからなければ、ログ・ファイルを検索してオーバーフロー・ログ・パスに入れます。オーバーフロー・ログ・パスが指定されていない場合、 DB2 はログ・ファイルの検索を試行しません。
- ロギングのためにロー・デバイスを構成し、 DataPropagator™ (DPROP)、または **db2ReadLog** API を呼び出す別のアプリケーションを使用している場合には、*overflowlogpath* データベース構成パラメーターを構成する必要があります。 DB2 は、ユーザー出口プログラムを呼び出してログ・ファイルを検索し、**db2ReadLog** API により要求されたログ・データを戻すかもしれません。検索されたログ・ファイルは、*overflowlogpath* データベース構成パラメーターにより指定されたパスに置かれます。

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『ロー I/O の指定』

関連資料:

- 330 ページの『db2ReadLog - ログの非同期読み取り』
- 371 ページの『付録 F. Tivoli Storage Manager』

ログ・ファイルをバックアップ・イメージに含める

オンライン・バックアップ操作を実行する際には、データベースのリストアおよびリカバリーに必要なログ・ファイルをバックアップ・イメージに含めることを指定できます。つまり、災害時リカバリー・サイトにバックアップ・イメージを送る必要がある場合、ログ・ファイルを個別に送信したり、自分でそれらをパッケージする必要はないということです。さらに、オンライン・バックアップの整合性を保つために必要なログ・ファイルを決める必要もなくなります。これにより、正常なリカバリーに必要なログ・ファイルを削除してしまうことをある程度防げます。

この機能を利用するには、BACKUP DATABASE コマンドの INCLUDE LOGS オプションを指定します。このオプションを指定すると、バックアップ・ユーティリティーは、現在アクティブなログ・ファイルを切り捨て、必要なログ・エクステント群をバックアップ・イメージにコピーします。

バックアップ・イメージからログ・ファイルをリストアするには、RESTORE DATABASE コマンドの LOGTARGET オプションを使用し、 DB2® サーバーに存在する完全修飾パスを指定します。そうすると、データベース・リストア・ユーティリティーは、イメージ内のログ・ファイルをターゲット・パスに書き込みます。同じ名前のログ・ファイルがターゲット・パスにすでに存在する場合、リストア操

作は失敗し、エラーが戻されます。 LOGTARGET オプションが指定されない場合、ログ・ファイルはバックアップ・イメージからリストアされません。

LOGTARGET オプションが指定される場合で、バックアップ・イメージにログ・ファイルが含まれていない場合、表スペース・データをリストアしようとする前に、エラーが戻されます。無効パスや読み取り専用パスが指定する場合にも、リストア操作は失敗します。 LOGTARGET オプションが指定されたデータベース・リストアまたは表スペース・リストア時に、 1 つ以上のログ・ファイルを取り出せない場合、リストア操作は失敗してエラーが戻されます。

バックアップ・イメージに保管されたログ・ファイルだけをリストアすることも選択できます。このためには、RESTORE DATABASE コマンドの LOGTARGET オプションに加えて LOGS オプションを指定します。リストア中のログ・ファイルがこのモードのときに、リストア操作に問題が生じる場合、このリストア操作は失敗してエラーが戻されます。

自動増分リストア操作においては、リストア操作のターゲット・イメージに含まれているログだけがバックアップ・イメージから取り出されます。増分リストア処理中に参照される中間イメージに含まれるログが、それらのバックアップ・イメージから取り出されることはありません。手動の増分リストア時に、ログ・ファイルを含むバックアップ・イメージをリストア中に、ログ・ターゲット・ディレクトリーを指定する場合、そのバックアップ・イメージに含まれるログ・ファイルがリストアされます。

注:

1. この機能は、単一パーティション・データベースでのみ使用可能です。
2. この機能は、オフライン・バックアップではサポートされていません。
3. ログがオンライン・バックアップ・イメージに含まれる場合、バージョン 8.2 より前の DB2 Universal Database™ (UDB) リリースでは、結果のイメージをリストアすることはできません。

関連タスク:

- 31 ページの『増分バックアップ・イメージからのリストア』

関連資料:

- 77 ページの『BACKUP DATABASE』
- 102 ページの『RESTORE DATABASE』

ログ・ファイルの消失を防止する方法

データベースをドロップしたり、ポイント・イン・タイム指定ロールフォワード・リカバリーを実行する必要がある状況で、将来のリカバリー操作で必要となる可能性のあるログ・ファイルを消失してしまう可能性があります。そのようなケースでは、現行データベース・ログ・パス・ディレクトリーにあるすべてのログのコピーを作成することが大切です。以下のシナリオを考慮に入れてください。

- リストア操作の前にデータベースをドロップする計画をしている場合、 DROP DATABASE コマンドを実行する前に、アクティブ・ログ・パスにあるログ・ファイルを保管する必要があります。それらのログ・ファイルのうちいくつかは、データベースをドロップする前にアーカイブされていなかった可能性があるた

め、そのデータベースがリストアされた後、それらのログ・ファイルがロールフォワード・リカバリーに必要になるかもしれません。通常、RESTORE コマンドを実行する前にデータベースをドロップする必要はありません。とはいえ、データベースは、RESTORE コマンドが失敗する程度にまで損傷しているため、データベースをドロップする必要があるかもしれません (または DROP DATABASE コマンドの AT NODE オプションを指定することにより、1 つのパーティションのデータベースをドロップします)。さらに、白紙状態から開始するために、リストア操作の前にデータベースをドロップするように決定するかもしれません。

- 特定の時点までのデータベースのロールフォワード操作を実行する場合は、指定するタイム・スタンプの後のログ・データは上書きされます。ポイント・イン・タイム指定ロールフォワード操作を完了し、データベースに再接続した後に、データベースを実際にはさらに後の時点までロールフォワードする必要があったと判断した場合でも、それはできません。ログがすでに上書きされてしまったからです。元のセットのログ・ファイルがアーカイブされている可能性はありますが、DB2® は、ユーザー出口プログラムを呼び出して自動的に新しく生成されたログ・ファイルをアーカイブしているかもしれません。ユーザー出口プログラムがどのように書かれているかにより、アーカイブ・ログ・ディレクトリーにある元のセットのログ・ファイルが上書きされる原因となることがあります。たとえ元のセットおよび新規セットのログ・ファイルの両方が (同じファイルの別のバージョンとして) アーカイブ・ログ・ディレクトリーにあったとしても、将来のリカバリー操作のためにどのセットのログを使用するかを決定する必要があるかもしれません。

関連概念:

- 36 ページの『リカバリー・ログについて』

リカバリー履歴ファイルについて

リカバリー履歴ファイルはデータベースごとに作成され、以下の操作が実行されるたびに自動更新されます。

- データベースまたは表スペースのバックアップ
- データベースまたは表スペースのリストア
- データベースまたは表スペースのロールフォワード
- 表スペースの作成
- 表スペースの変更
- 表スペースの静止
- 表スペースの名前変更
- 表スペースのドロップ
- 表のロード
- 表のドロップ (ドロップされた表のリカバリーが可能な場合)
- 表の再編成
- オンデマンド・ログ・アーカイブの呼び出し
- 新規ログ・ファイルの書き込み (リカバリー可能ログの使用時)
- ログ・ファイルのアーカイブ (リカバリー可能ログの使用時)
- データベースのリカバリー

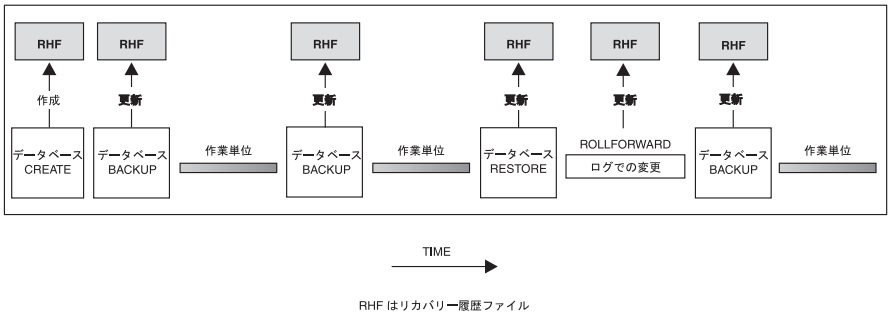


図 9. リカバリー履歴ファイルの作成と更新

このファイルのバックアップ情報の要約を使用して、ある時点までデータベースの全体または一部をリカバリーできます。このファイルには、以下のような情報が含まれています。

- それぞれのレコードを一意的に識別するための識別 (ID) フィールド
- コピーされたデータベースの部分とその方法
- コピーが作成された時刻
- コピーの位置 (装置情報とこのコピーにアクセスする論理方法とを示す)
- リストア操作が行われた最終時刻
- 表スペースの名前が変更された時刻 (その表スペースの以前の名前および現在の名前を示す)
- バックアップ操作の状況: アクティブ、非アクティブ、有効期限切れ、または削除済み
- データベース・バックアップにより保管された、またはロールフォワード・リカバリー操作中に処理された最後のログ・シーケンス番号

リカバリー履歴ファイル内のレコードを参照するには、 LIST HISTORY コマンドを使用してください。

どのバックアップ操作 (データベース、表スペース、または増分) にも、リカバリー履歴ファイルのコピーが含まれます。リカバリー履歴ファイルは、データベースに関連付けられています。データベースを削除すると、リカバリー履歴ファイルも削除されます。データベースを新規の位置にリストアすると、リカバリー履歴ファイルもリストアされます。ディスク上にあるファイルに項目がない時以外は、リストアの際に、既存のリカバリー履歴ファイルは上書きされません。ディスク上にあるファイルに項目がない時には、データベース履歴は、バックアップ・イメージからリストアされます。

現在のデータベースが使用不可で、関連するリカバリー履歴ファイルが壊れていたり削除されていたりする場合は、 RESTORE コマンドのオプションを使って、リカバリー履歴ファイルだけをリストアできます。その後、そのリカバリー履歴ファイルを調べて、データベースのリストアに使用するバックアップの情報を得ることができます。

ファイルのサイズは、*rec_his_retentn* 構成パラメーターで制御されており、そのファイルの項目の保持期間が日数単位で指定されます。このパラメーターの値がゼロ (0)

に設定されていても、最新の全データベース・バックアップ (とそのリストア・セット) は保持されます。 (このコピーを除去する唯一の方法は、 FORCE オプションを指定した PRUNE を使用することです。) 保存期間のデフォルト値は 366 日間です。この期間に -1 を使用すると、不特定の日数を設定できます。その場合、ファイルの明示的な整理が必要になります。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『rec_his_retentn - 「リカバリー履歴保存期間」構成パラメーター』
- 298 ページの『LIST HISTORY』

リカバリー履歴ファイル - ガーベッジ・コレクション

ガーベッジ・コレクション

PRUNE HISTORY コマンドを使用して履歴ファイルからレコードを削除できますが、この種の整理は DB2® に任せることをお勧めします。リカバリー履歴ファイルに記録される DB2 データベース・バックアップの数は、 DB2 *ガーベッジ・コレクション* によって自動的にモニターされます。以下の状況で DB2 ガーベッジ・コレクションが起動します。

- 完全な非増分バックアップ操作が正常に完了した後。
- ロールフォワード操作不要のデータベース・リストア操作が正常に完了した後。
- 正常なデータベース・ロールフォワード操作が正常に完了した後。

構成パラメーター *num_db_backups* は、保持されるアクティブな全 (非増分) データベース・バックアップ・イメージの数を定義します。履歴ファイルを、最後の項目から始めてスキャンするために、このパラメーターの値が使用されます。

すべての全 (非増分) データベース・バックアップ操作の後で、 *rec_his_retentn* 構成パラメーターを使用して、履歴ファイルから有効期限切れの項目が整理されます。

アクティブ・データベース・バックアップ は、データベースの現行の状態をリカバリーするように現行のログを使用してリストアおよびロールフォワードできるものです。*非アクティブ・データベース・バックアップ* は、リストアの際に、データベースを直前の状態に戻します。

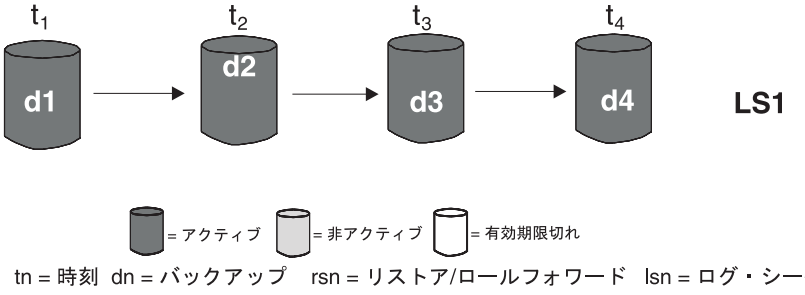


図 10. アクティブ・データベース・バックアップ: *num_db_backups* の値は 4 に設定されています。

必要でなくなったアクティブ・データベース・バックアップには、すべて「有効期限切れ」とマークされます。これらのイメージは、もはや必要のないものと見なされます。さらに新しいバックアップ・イメージがあるためです。この有効期限が切れたデータベース・バックアップ・イメージより前にとった表スペースのバックアップ・イメージおよびロード・バックアップ・コピーも、「有効期限切れ」とマークされます。

「非アクティブ」のマークが付いており、有効期限のすでに切れているデータベース・バックアップが作成された時点よりも前に作成されたデータベース・バックアップ・イメージも、「有効期限切れ」とマークされます。すべての関連した非アクティブの表スペース・バックアップ・イメージおよびロード・バックアップ・コピーにも、「有効期限切れ」のマークが付きます。

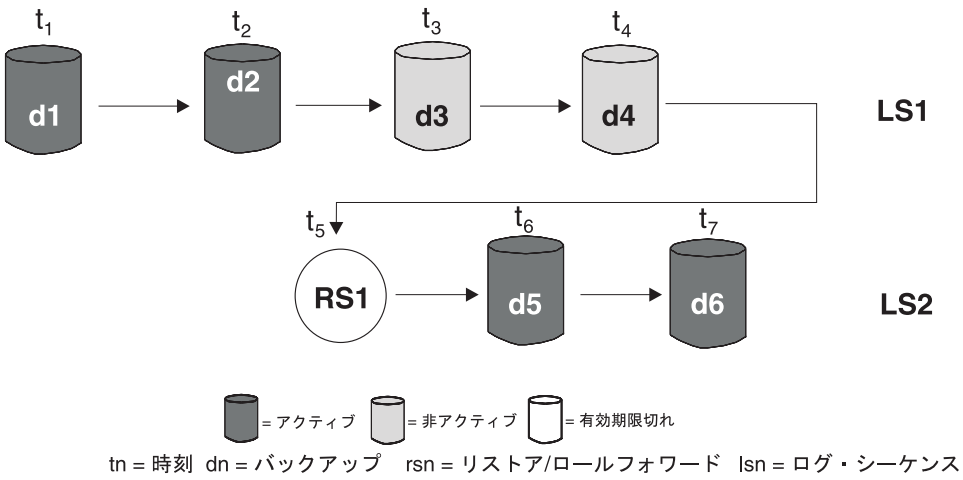


図 11. 非アクティブ・データベース・バックアップ

アクティブのデータベース・バックアップ・イメージがリストアされるものの、履歴ファイルに記録されている最新のデータベース・バックアップではない場合には、これと同じログ・シーケンスに属する後続のデータベース・バックアップ・イメージは「非アクティブ」としてマークされます。

非アクティブのデータベース・バックアップ・イメージがリストアされた場合、現行のログ・シーケンスに属する非アクティブ・データベース・バックアップは、再び「アクティブ」としてマークされるようになります。現行のログ・シーケンスに入っていないすべてのアクティブなデータベース・バックアップ・イメージには、「非アクティブ」のマークが付けられます。

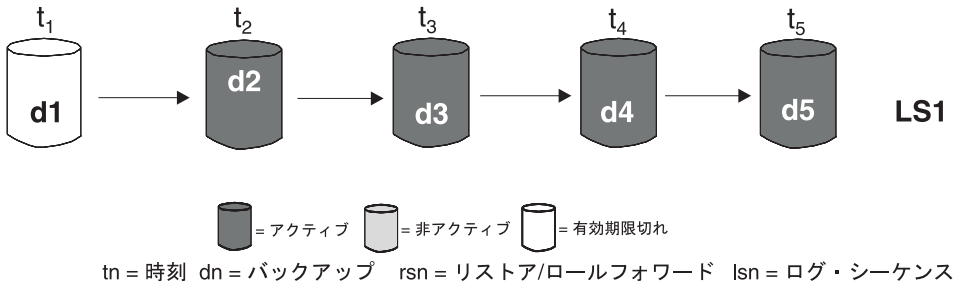


図 12. 有効期限が切れたデータベース・バックアップ

DB2 ガーベッジ・コレクションは、そのバックアップが現行のログ・シーケンス (現行のログ・チェーン とも呼ばれる) に対応していない場合に、 DB2 データベースまたは表スペースのバックアップ・イメージの履歴ファイル項目に、「非アクティブ」のマークを付けます。現行のログ・シーケンスは、リストアされた DB2 データベース・バックアップ・イメージと、処理済みのログ・ファイルにより判別されます。データベース・バックアップ・イメージがリストアされると、以後のデータベース・バックアップ・イメージはすべて「非アクティブ」になります。リストアされたイメージは、新しくログ・チェーンを始めるためです。 (このことは、バックアップ・イメージをロールフォワードせずにリストアした場合も当てはまります。ロールフォワード操作を行うと、ログ・チェーン内の中断後に取られたデータベース・バックアップは、「非アクティブ」のマークが付けられます。ロールフォワード・ユーティリティーにより、損傷した現行バックアップ・イメージを含むログ・シーケンスが調べられるので、古いデータベース・バックアップ・イメージをリストアする必要が生じることが考えられます。)

表スペースをリストアした後で、現行のログ・シーケンスを適用して、データベースの現行の状態に達することができない場合には、その表スペース・レベルのバックアップ・イメージは「非アクティブ」になります。

バックアップ・イメージに DATALINK 列が含まれている場合には、 DB2 Data Links Manager を実行しているすべてのデータ・リンク・サーバーに連絡が取られ、ガーベッジ・コレクションが要求されます。続いて、有効期限切れのバックアップに含まれていたが、次のデータベース・バックアップ操作の前にリンク解除された データ・リンク・サーバーの関連ファイルのバックアップが、 DB2 ガーベッジ・コレクションにより削除されます。

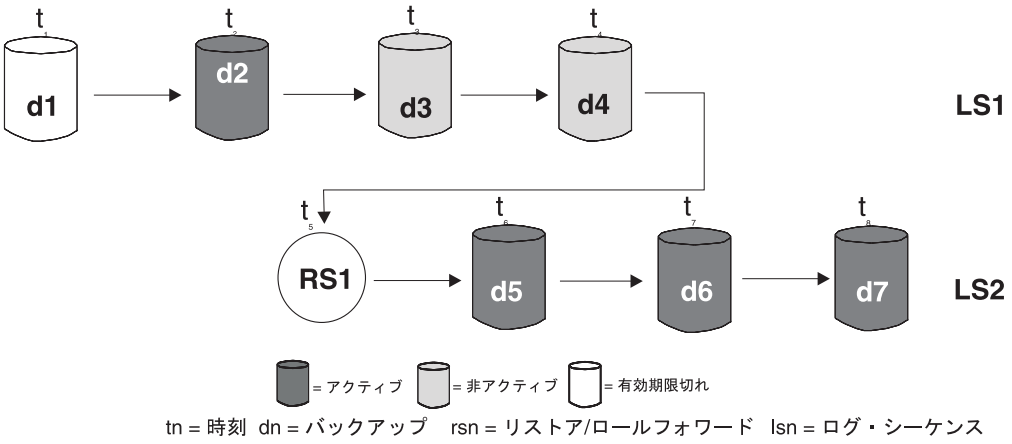


図 13. アクティブ、非アクティブ、有効期限切れのデータベース・バックアップの混合

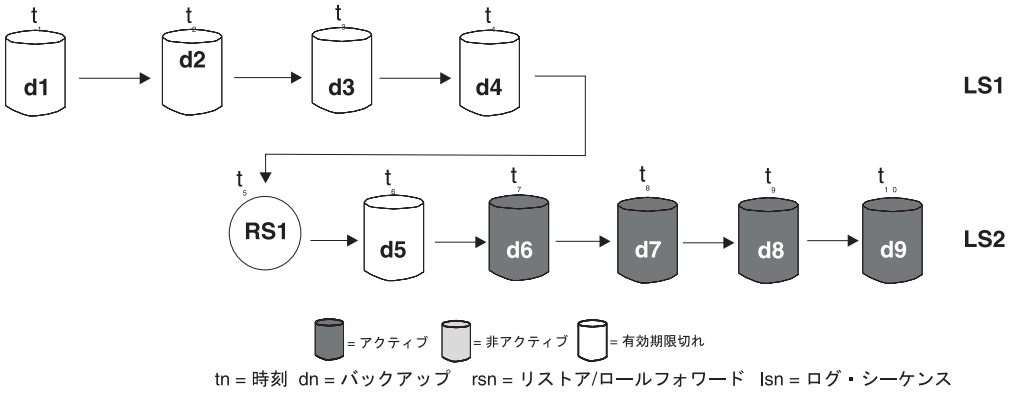


図 14. 有効期限が切れたログ・シーケンス

関連概念:

- 60 ページの『リカバリー履歴ファイルについて』

関連資料:

- 301 ページの『PRUNE HISTORY/LOGFILE』

表スペースの状態について

表スペースの現行の状況は、その *状態* を反映しています。リカバリーに関連した最も一般的な表スペースの状態は、以下のとおりです。

- バックアップ・ペンディング。ロールフォワード操作の特定の時点の後や、コピー・オプションを指定しないロード操作の後に、表スペースはこの状態になります。この場合、その表スペースを使用する前にバックアップを作成する必要があります。 (バックアップが作成されていない場合は、表スペースを更新することはできませんが、読み取り専用操作は行えます。)
- リストア・ペンディング状態。表スペースのロールフォワード操作が取り消された場合や、表スペースのロールフォワード操作中にリカバリー不能エラーが生じた場合 (この場合は、表スペースのリストアとロールフォワードを再度行わなければならない) に、表スペースはこの状態になります。

- *ロールフォワード進行中*。表スペースのロールフォワード操作が進行中の場合に、表スペースはこの状態になります。ロールフォワード操作が正常に完了すると、表スペースはロールフォワード進行中状態ではなくなります。ロールフォワード操作が取り消された場合も表スペースはこの状態ではなくなります。
- *ロールフォワード・ペンディング状態*。表スペースのリストア後、または入出力 (I/O) エラーの後に、表スペースはこの状態になります。表スペースのリストア後の場合は、ログの最後まで、または特定のポイント・イン・タイムまでのロールフォワードを実行できます。入出力エラーの後は、表スペースをログの最後までロールフォワードしなければなりません。

リカバリー・パフォーマンスの向上

リカバリーのパフォーマンスについて考慮する際には、以下の点について考慮に含めてください。

- ログを別の装置に置くことによって、頻繁に更新されるデータベースのパフォーマンスを改善できます。トランザクション処理 (OLTP) 環境では、多くの場合、ログにデータを書き込むには、データ行を保管するよりも入出力が多くなります。ログを別の装置に入れるなら、データベース・ファイルとログとの間で移動するのに必要なディスク・アーム移動量を最小限にとどめることができます。

  他のどのファイルをそのディスクに入れるかについても考慮する必要があります。たとえば、実メモリーが不足しているシステムで、システム・ページングのために使用しているディスクにログを移動すると、調整は台なしになってしまいます。

  DB2® は、バッファー数、バッファー・サイズ、および並列処理設定の最適値を自動的に選択することで、バックアップまたはリストア操作の完了にかかる時間を最小限に抑えようとします。この値は、使用可能なユーティリティー・ヒープ・メモリーの大きさ、使用可能なプロセッサー数、およびデータベース構成に基づきます。
- リストア操作を完了するために必要な時間を短縮するためには、次の方法が有効です。
  - リストア・バッファー・サイズを調整する。バッファー・サイズは、バックアップ操作時に使用されたバッファー・サイズの倍数でなければなりません。
  - バッファー数を増やします。

    複数のバッファーと入出力メディア装置を使用する場合、少なくともメディア装置の 2 倍のバッファーを使用し、メディア装置がデータを待つ状態にならないようにします。使用するバッファーのサイズも、リストア操作のパフォーマンスに影響を与えます。理想的なリストア・バッファーのサイズは、表スペースのエクステント・サイズの倍数です。

    複数の表スペースがありそれぞれエクステント・サイズが異なる場合は、最大エクステント・サイズの倍数の値を指定します。

    推奨する*最小の* バッファー数は、メディア装置の数に、 PARALLELISM オプションに指定される数を加えたものです。
  - 複数のソース装置を使用する。

- リストア操作の PARALLELISM オプションを、ソース装置の数より少なくとも 1 つ多くなるように設定する。

- 表に大量の長形式フィールド・データおよび LOB データが含まれている場合は、それらのリストアには長時間かかります。データベースをロールフォワード・リカバリー可能にしておくと、 RESTORE コマンドでは特定の表スペースをリストアできるようになります。長形式フィールド・データおよび LOB データが業務にとって重要な場合は、これらの表スペースをリストアするときは、これらの表スペースのバックアップ・タスクを完了するために必要な時間を考慮する必要があります。したがって、長形式フィールドのデータおよび LOB データを別々の表スペースに保管させておけば、長形式フィールド・データおよび LOB データが含まれている表スペースのリストアを選択しないことで、リストア操作を完了するための時間を短縮させることができます。 LOB データが別のソースから再作成可能である場合は、 LOB 列を含む表の作成または変更時には、NOT LOGGED オプションを選択してください。長形式フィールドおよび LOB データを含む表スペースはリストアしないが、この表が含まれている表スペースのロールフォワードを希望する場合は、ログの最後までロールフォワードを実行し、表データが含まれるすべての表スペース間で整合性を保証するようにしてください。

  注: 表データが入っている表スペースのバックアップ時に、関連する LONG または LOB フィールドがないと、その表スペースのポイント・イン・タイム・ロールフォワード・リカバリーは実行できません。表に関するすべての表スペースは、同じ時点まで同時にロールフォワードする必要があります。

- バックアップ操作とリストア操作には、以下の事柄が適用されます。
  - 複数の入出力バッファーと装置を使用してください。
  - 使用する装置の少なくとも 2 倍のバッファーを割り振ってください。
  - 入出力装置のコントローラーの処理速度が過負荷にならないようにしてください。
  - いくつかの大きなバッファーを使用する代わりに、より多くの小さなサイズのバッファーを使用してください。
  - システム・リソースに従い、バッファー数とサイズを調整してください。
  - PARALLELISM オプションを使用してください。

- DB2® では、複数のエージェントを使用して、クラッシュ・リカバリーとデータベースのロールフォワード・リカバリーの両方を実行するようになりました。特に、対称マルチプロセッサー (SMP) マシンの場合、これらの操作の際にパフォーマンスが向上することを期待できます。データベースをリカバリーする際に複数のエージェントを使用すると、 SMP マシン上で使用可能な CPU が増えるという利点があります。

  並列リカバリーで導入されたエージェントのタイプは db2agnsc です。 DB2 では、マシン上の CPU の数を基にして、データベースのリカバリーに使用するエージェントの数が選択されます。

  DB2 によりこれらのエージェントにログ・レコードが配布されるので、該当するエージェントで並行して再適用できます。たとえば、挿入、削除、更新、追加キー、および削除キーの操作に関連したログ・レコードの処理をこの方法で並列化できます。ログ・レコードはページ・レベルで並列化される (同じデータ・ペー

ジ上のログ・レコードは同じエージェントによって処理される) ので、すべての作業が 1 つの表で行われる場合でも、パフォーマンスは拡張されます。

関連概念:

- 92 ページの『バックアップのパフォーマンスの最適化』
- 95 ページの『リストアの概要』

第 2 章 データベースのバックアップ

ここでは、DB2 UDB バックアップ・ユーティリティーについて説明します。このユーティリティーは、データベースや表スペースのバックアップ・コピーの作成に使用します。

以下のトピックについて説明します。

バックアップの概要

DB2® BACKUP DATABASE コマンドの最も単純な形式の場合、必要なのは、バックアップしたいデータベースの別名を指定することだけです。たとえば、次のようにします。

```
db2 backup db sample
```

コマンドが正常に完了すると、コマンドを出したパスまたはディレクトリーに新しいバックアップ・イメージが作成されます。このディレクトリーに入れられる理由は、この例のコマンドはバックアップ・イメージの宛先を明示的に指定していないからです。たとえば、Windows® オペレーティング・システムの場合、このコマンドは (ルート・ディレクトリーから出すと)、以下にリストされているディレクトリーにイメージを作成します。

```
 Directory of D:¥SAMPLE.0¥DB2¥NODE0000¥CATN0000¥20010320

03/20/2001  12:26p      <DIR>          .
03/20/2001  12:26p      <DIR>          ..
03/20/2001  12:27p          12,615,680 122644.001
```

注: DB2 クライアントおよびサーバーが同じシステム上にない場合、DB2 は、クライアント・マシンで現行作業ディレクトリーを判別し、そのディレクトリーを、サーバーのバックアップ・ターゲット・ディレクトリーとして使用します。このため、バックアップ・イメージ用のターゲット・ディレクトリーを指定することをお勧めします。

バックアップ・イメージは、バックアップ・ユーティリティーを起動する際に指定された宛先に作成されます。指定できる場所は次のとおりです。

- ディレクトリー (ディスクまたはディスケットへのバックアップの場合)

- 装置 (テープへのバックアップの場合)
- Tivoli® Storage Manager (TSM) サーバー
- 他のベンダーのサーバー

データベースのバックアップ操作を起動すると、履歴ファイルがサマリー情報によって自動的に更新されます。このファイルは、データベース構成ファイルと同じディレクトリーに作成されます。

UNIX® ベースのシステムでは、ディスク上に作成されるバックアップ・イメージのファイル名は、複数のエレメントを連結してピリオドで区切ったものになります。

```
DB_alias.Type.Inst_name.NODEnnnn.CATNnnnn.timestamp.Seq_num
```

たとえば、次のようにします。

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

Windows オペレーティング・システムでは、4 レベルのサブディレクトリー・ツリーが使用されます。

```
DB_alias.Type¥Inst_name¥NODEnnnn¥CATNnnnn¥yyyymmdd¥hhmmss.Seq_num
```

たとえば、次のようにします。

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

データベース別名
バックアップ・ユーティリティーを起動する際に指定した、1 ～ 8 文字のデータベース別名。

タイプ
バックアップ操作のタイプ。 `0` は全データベース・レベルのバックアップ、 `3` は表スペース・レベルのバックアップ、 `4` は LOAD...COPY TO コマンドによって生成されたバックアップ・イメージ

インスタンス名
DB2INSTANCE 環境変数から取られる 1 ～ 8 文字の現行インスタンス名。

ノード番号
ノード番号。非パーティション・データベース・システムでは、この値は常に `NODE0000` です。パーティション・データベース・システムでは、この値は `NODE`*xxxx* です。 *xxxx* は、 `db2nodes.cfg` ファイル中でノードに割り当てられている数です。

カタログ・ノード番号
データベース用のカタログ・ノードのノード番号です。非パーティション・データベース・システムでは、この値は常に `CATN0000` です。パーティション・データベース・システムでは、この値は `CATN`*xxxx* です。 *xxxx* は、 `db2nodes.cfg` ファイル中でノードに割り当てられている数です。

タイム・スタンプ
バックアップ操作が実行された日付と時刻を 14 文字で表記したもの。タイム・スタンプの形式は *yyyymmddhhnnss* です。ただし、

- *yyyy* は年 (1995 ～ 9999)
- *mm* は月 (01 ～ 12)
- *dd* は日 (01 ～ 31)

- *hh* は時 (00 ～ 23)
- *nn* は分 (00 ～ 59)
- *ss* は秒 (00 ～ 59)

シーケンス番号 ファイル拡張子として使用する 3 桁の番号。

バックアップ・イメージをテープに書き込む際には、以下のようになります。

- ファイル名は作成されません。しかし、検査するために、上記の情報がバックアップ・ヘッダーに保管されます。
- テープ装置が標準オペレーティング・システム・インターフェースを介して使用可能でなければなりません。しかし、大規模なパーティション・データベース・システムでは、テープ装置を各データベース・パーティション・サーバーに専用接続することは実際的でないことがあります。この場合は、複数のテープ装置を 1 つまたは複数の TSM サーバーに接続することができます。これは、これらのテープ装置に対するアクセスを、各データベース・パーティション・サーバーから可能にするためです。
- パーティション・データベース・システムでは、 REELlibrarian 4.2 または CLIO/S などの仮想テープ装置機能を提供している製品を使用することもできます。これらの製品を使用し、疑似テープ装置を介して他のノード (データベース・パーティション・サーバー) に接続されているテープ装置にアクセスすることができます。リモート・テープ装置へのアクセスはユーザーが意識せずに実行され、標準オペレーティング・システム・インターフェースを介して疑似テープ装置にアクセスできます。

使用不能状態になっているデータベースのバックアップをとることはできません (バックアップ・ペンディング状態のデータベースを除く)。いずれかの表スペースが異常な状態になっている場合は、その表スペースがバックアップ・ペンディング状態でない限り、そのデータベースまたはその表スペースのバックアップをとることはできません。

同じ表スペースに対する並行バックアップ操作は許可されていません。特定の表スペースに対してバックアップ操作が開始されたら、それ以降の操作は失敗します (SQL2048)。

リストア操作中にシステム障害が起きたために、データベースまたは表スペースが部分的にリストアされた状態になっている場合、そのデータベースまたは表スペースを正常にリストアしてからでなければ、バックアップをとることはできません。

バックアップ操作は、バックアップする表スペースのリストに TEMPORARY 表スペースの名前が含まれていると、失敗します。

バックアップ・ユーティリティーには、異なるデータベースのバックアップ・コピーを作成する複数のプロセスのための並行性を制御する機能が用意されています。この並行制御機能のために、すべてのバックアップ操作が終了するまでバックアップ先の装置はオープンしたままになります。バックアップ操作時にエラーが発生してオープン・コンテナーがクローズできない場合は、同じドライブに対する他のバックアップ操作にはアクセス・エラーが発生することがあります。この種のアクセス・エラーを訂正するには、エラーが発生したバックアップ操作を終了し、バックアップ先装置との接続も切断する必要があります。バックアップ・ユーティリティ

ーを使用してテープへのバックアップの複数の並行操作を実行する場合は、それらのプロセスのバックアップ先を同じテープにしないようにしてください。

バックアップ情報の表示

db2ckbkp を使用して、既存のバックアップ・イメージに関する情報を表示できます。このユーティリティーによって、次のことが可能です。

- バックアップ・イメージの保全性のテスト、およびリストアできるかどうかの判別。
- バックアップ・ヘッダーに保管されている情報の表示。
- バックアップ・イメージのオブジェクトおよびログ・ファイル・ヘッダーに関する情報の表示。

関連概念:

- 3 ページの『バックアップおよびリカバリー計画の作成』
- 60 ページの『リカバリー履歴ファイルについて』
- 58 ページの『ログ・ファイルをバックアップ・イメージに含める』
- 「*管理ガイド: プランニング*」の『自動保守』

関連資料:

- 371 ページの『付録 F. Tivoli Storage Manager』

バックアップの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャー保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスが制御されます。ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

バックアップ・ユーティリティーを使用するには、 SYSADM、SYSCTRL、または SYSMAINT 権限が必要です。

バックアップの使用

前提条件:

バックアップを作成しようとしているデータベースに接続しないでください。指定したデータベースへの接続はデータベース・バックアップ・ユーティリティーにより自動的に確立され、この接続はバックアップ操作が完了すると終了します。バックアップする予定のデータベースに接続している場合、BACKUP DATABASE コマンドが発行されると、接続は切断され、バックアップ操作が進められます。

データベースには、ローカル・エージェントとリモート・エージェントがあります。 Tivoli Storage Manager (TSM) などのストレージ管理製品を使用していない限り、バックアップ・イメージはデータベース・サーバーに残ります。

パーティション・データベース・システムでは、データベース・パーティションのバックアップは個々に行われます。この操作は、ユーティリティーを呼び出すデータベース・パーティション・サーバーでローカルに行われます。しかし、インスタンスのいずれかのデータベース・パーティション・サーバーから **db2_all** を出すことで、サーバーのリスト (ノード番号で識別) についてバックアップ・ユーティリティーを起動することができます。 (ユーザー表のあるノード、つまりデータベース・パーティション・サーバーを識別するには、 LIST NODES コマンドを使用してください。) これを実行する場合は、最初にカタログ・ノードのバックアップを取り、次に、他のデータベース・パーティションのバックアップを取る必要があります。コマンド・エディターを使用してデータベース・パーティションのバックアップを取ることもできます。この方法の場合はロールフォワード・リカバリーはサポートされないので、これらのノード上のデータベースのバックアップを定期的に取ってください。作成する任意のバックアップ・コピーと共に `db2nodes.cfg` ファイルのコピーも保存する必要があります。これは、このファイルに対する損傷の保護です。

分散要求システムでは、バックアップ操作は、当該データベース・カタログに保管されている分散要求データベースおよびメタデータ (ラッパー、サーバー、ニックネームなど) に適用されます。データ・ソース・オブジェクト (表およびビュー) は、分散要求データベースに保管されていないかぎり、バックアップされません。

過去のリリースのデータベース・マネージャーでデータベースを作成し、そのデータベースを移行していない場合は、データベースを移行してからでなければバックアップをとることはできません。

制限:

バックアップ・ユーティリティーには、以下の制限が適用されます。

- 別々の表スペースが関係している場合でも、表スペースのバックアップ操作と表スペースのリストア操作とを同時に実行することはできません。
- パーティション・データベース環境でロールフォワード・リカバリーを使用できるようにしたい場合は、定期的にノード・リストについてデータベースのバックアップをとる必要があり、また、システム内の残りのノードのバックアップ・イメージも少なくとも 1 つは作成する必要があります (該当するデータベースに関するユーザー・データを含んでいない場合でも)。データベースに関するユーザー・データを含んでいないデータベース・パーティション・サーバーで、データベース・パーティションのバックアップ・イメージが必要となるのは、次の 2 つの場合です。
  - 最後のバックアップを作成した後にデータベース・システムにデータベース・パーティション・サーバーを追加し、このデータベース・パーティション・サーバーについて順方向リカバリーを実行する必要がある場合。
  - 特定時点のリカバリーを使用する場合。この場合は、システム内のすべてのデータベース・パーティションがロールフォワード・ペンディング状態でなければなりません。
  - DMS 表スペースのオンライン・バックアップ操作は、以下の操作との互換はありません。
    - ロード
    - 再編成 (オンラインおよびオフライン)

- 表スペースのドロップ
- 表の切り捨て
- 索引の作成
- NOT LOGGED INITIALLY (CREATE TABLE および ALTER TABLE ステートメントと共に使用)

手順:

バックアップ・ユーティリティーは、コマンド行プロセッサー (CLP)、コントロール・センターにあるバックアップ・データベース・ノートブックまたはウィザード、または **db2Backup** アプリケーション・プログラミング・インターフェース (API) を通して起動できます。

CLP によって発行する BACKUP DATABASE コマンドの例を以下に示します。

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

「データベースのバックアップ (Backup Database)」ノートブックまたはウィザードをオープンするには、次のようにします。

1. コントロール・センターから、「データベース (Databases)」フォルダーが見つかるまでオブジェクト・ツリーを展開します。
2. 「データベース (Databases)」フォルダーをクリックします。ウィンドウの右側部分 (目次ペイン) に、既存のデータベースがすべて表示されます。
3. 目次ペイン内で対象となるデータベースをマウスの右ボタンでクリックし、ポップアップ・メニューから「データベースのバックアップ (Backup Database)」または「ウィザードを使用したデータベースのバックアップ (Backup Database Using Wizard)」を選択します。「データベースのバックアップ (Backup Database)」ノートブックまたは「データベースのバックアップ (Backup Database)」ウィザードがオープンします。

詳しい情報については、コントロール・センターのオンライン・ヘルプ機能をご覧ください。

関連概念:

- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『組み込み SQL または DB2 CLI プログラムにおける管理 API』
- 「*管理ガイド: インプリメンテーション*」の『コントロール・センター用のプラグイン・アーキテクチャーの紹介』

関連タスク:

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『データベースの移行』

関連資料:

- 「コマンド・リファレンス」の『LIST DBPARTITIONNUMS コマンド』
- 83 ページの『db2Backup - データベースのバックアップ』

テープへのバックアップ

データベースまたは表スペースのバックアップをとる場合、ブロック・サイズおよびバッファー・サイズを正しく設定しなければなりません。これは、特に可変長のブロック・サイズを使用する場合 (たとえば、AIX® でブロック・サイズがゼロに設定されている場合など) に当てはまります。

バックアップ時に使用できる固定ブロック・サイズ数には制限があります。このような制限があるのは、 DB2® がバックアップ・イメージ・ヘッダーを 4 KB ブロックとして書き出すためです。 DB2 がサポートしている固定ブロック・サイズは 512、1024、2048、および 4096 バイトです。固定ブロック・サイズを使用する場合、任意のバックアップ・バッファー・サイズを指定できます。ただし、固定ブロック・サイズが、DB2 でサポートされているサイズのいずれかでない場合、バックアップ操作が正常に完了しない場合があります。

データベースが巨大な場合、固定ブロック・サイズを使用すると、バックアップ操作の完了に時間がかかる可能性があります。可変長ブロック・サイズを使用することもできます。

注: 可変長ブロック・サイズの使用は、現時点ではサポートされていません。このオプションを使用しなければならない場合は、可変長ブロック・サイズで作成したバックアップ・イメージを使用して正常にリカバリーできる手順を、十分にテストした上で適切な場面で使用するようにしてください。

可変長ブロック・サイズを使用する場合、指定するバッファー・サイズを、使用するテープ装置で許容される上限のサイズ以下にしなければなりません。パフォーマンスを最適にするには、バッファー・サイズを、使用する装置のブロック・サイズの上限に等しくしなければなりません。

磁気テープ装置を Windows® オペレーティング・システム上で使用できるようにする前に、以下のコマンドを発行する必要があります。

```
db2 initialize tape on <device> using <blksize>
```

ここで、

<device>
は、有効な磁気テープ装置名です。 Windows オペレーティング・システムでのデフォルトは、¥¥.¥TAPE0 です。

<blksize>
は、磁気テープのブロック化因数です。係数または 4096 の倍数を指定してください。デフォルト値は、装置のデフォルト・ブロック・サイズです。

可変ブロック・サイズのバックアップ・イメージからリストアすると、エラーが戻されることがあります。エラーが戻された場合、適当なブロック・サイズを使用してイメージを再書き込みしなければならない場合があります。以下に、AIX の場合の例を示します。

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

これにより、 backup_filenam.file というファイルにバックアップ・イメージのダンプが取られます。次に、**dd** コマンドにより、ブロック・サイズ 4096 を使用して、イメージのダンプが再びテープに取られます。

イメージが大き過ぎてファイルにダンプを取ることができない場合、この方法では問題が生じます。解決方法の 1 つは、**dd** コマンドを使用して、テープ装置から別のテープ装置にダンプを行うことです。ただし、イメージが複数のテープにまたがっている場合、この方法は使用できません。 2 つのテープ装置を使用する場合、以下の **dd** コマンドを使用します。

```
dd if=/dev/rmt1 of=/dev/rmt0 obs=4096
```

2 つのテープ装置を使用することができない場合、 **dd** コマンドを使用してロー・デバイスにイメージのダンプを行ってから、そのロー・デバイスからテープにそのイメージのダンプを行うことができます。この方法の問題は、ロー・デバイスにダンプが行われたブロックの数を、 **dd** コマンドが必ず 把握していなければならない点です。この数を、イメージをテープに戻すときに指定しなければなりません。 **dd** コマンドを使用してロー・デバイスからテープにイメージのダンプを行う場合、このコマンドはロー・デバイスの全体の内容のダンプをテープに取ります。 **dd** ユーティリティーでは、イメージを保持するのに使用されるロー・デバイスのサイズを判別することはできません。

バックアップ・ユーティリティーを使用する場合、テープ装置のブロック・サイズの上限を知っていなければなりません。以下は、その例です。

装置	アタッチ	ブロック・サイズの上限	DB2 バッファー・サイズの上限 (4 KB ページ単位)
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

注:

1. 7332 では、ブロック・サイズの上限が設定されていません。 256 KB は単なる推奨値です。ブロック・サイズの上限は親アダプターによって決まります。
2. 3590 は 2 MB のブロック・サイズをサポートしていますが、必要とされるパフォーマンスに応じて、それよりも小さい値 (256 KB など) を使用してみることもできます。
3. ご使用の装置の限度に関する情報は、その装置の資料を参照するか、装置のベンダーに連絡してください。

名前付きパイプへのバックアップ

UNIX ベースのシステムでは、ローカル名前付きパイプにデータベースのバックアップを作成する (および、ローカル名前付きパイプからデータベースをリストアする) ためのサポートを使用できるようになりました。

前提条件:

名前付きパイプの書き込みプログラムと読み取りプログラムは同じマシン上になければなりません。パイプがローカル・ファイル・システム上になければなりません。名前付きパイプはローカル装置として扱われるので、宛先として名前付きパイプを指定する必要があります。

手順:

以下に、AIX の場合の例を示します。

1. 名前付きパイプを作成します。

   ```
   mkfifo /u/dmcinnis/mypipe
   ```

2. リストア・ユーティリティーでこのバックアップ・イメージを使用する計画の場合は、データが失われないように、バックアップ操作の*前に* リストア操作を起動しなければなりません。

   ```
   db2 restore db sample into mynewdb from /u/dmcinnis/mypipe
   ```

3. データベースのバックアップ操作の宛先としてこのパイプを使用します。

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

関連タスク:

- 72 ページの『バックアップの使用』

関連資料:

- 77 ページの『BACKUP DATABASE』
- 102 ページの『RESTORE DATABASE』

BACKUP DATABASE

データベースまたは表スペースのバックアップ・コピーを作成します。

有効範囲:

このコマンドは、それが実行されたデータベース・パーティションに対してだけ影響を与えます。

権限:

以下のいずれかが必要です。

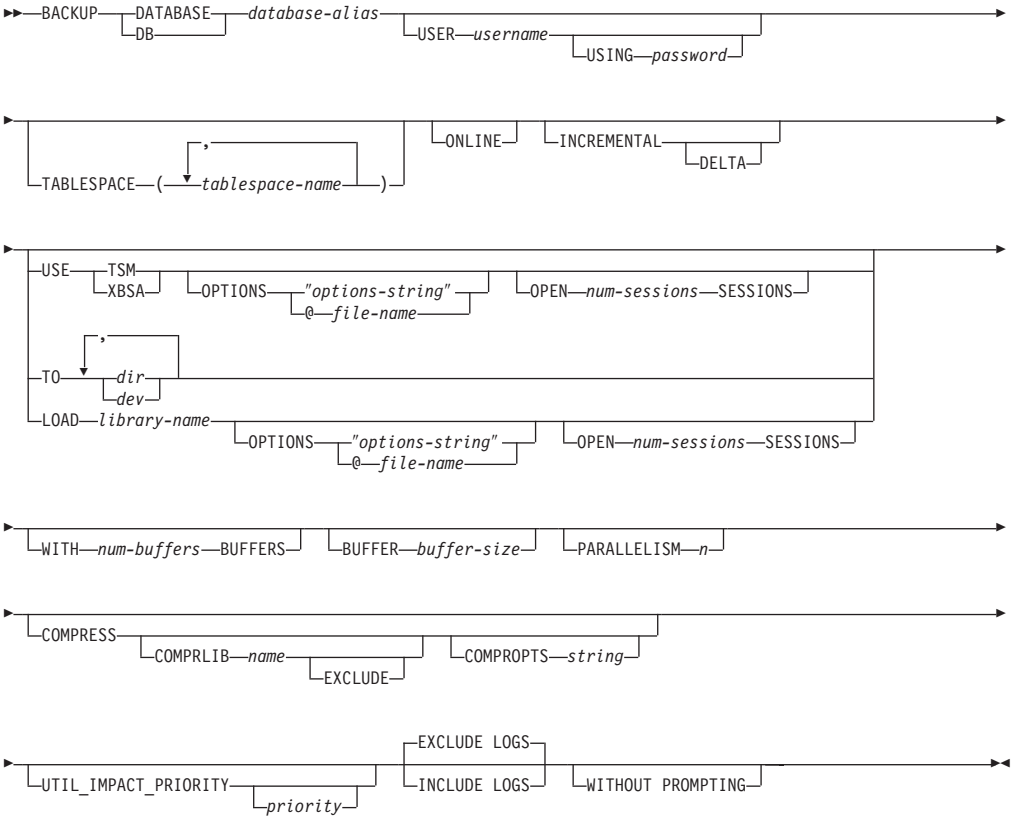
- *sysadm*
- *sysctrl*
- *sysmaint*

必要な接続:

データベース。このコマンドは指定されたデータベースへの接続を自動的に確立します。

注: 指定したデータベースへの接続がすでに存在している場合、その接続は終了して、バックアップ操作のために専用の接続が新規に確立されます。接続は、バックアップ操作の完了時に終了します。

コマンド構文:



コマンド・パラメーター:

DATABASE database-alias
バックアップを取るデータベースの別名を指定します。

USER username
データベースのバックアップを取るユーザー名を識別します。

USING password
ユーザー名を認証するために使用するパスワード。パスワードを省略すると、ユーザーに入力を求めるプロンプトが出ます。

TABLESPACE tablespace-name
バックアップを取る表スペースを指定するときに使用する名前のリスト。

ONLINE
オンライン・バックアップを指定します。デフォルトはオフライン・バックアップです。オンライン・バックアップは、 *logretain* または *userexit* を使用可能にして構成されたデータベースにのみ、使用可能です。オンライン・バックアップでは、 SMS 表スペース内の表が処理される際にそのようなす

べての表に対して DB2 が IN (Intent None) ロックを取得します。また、DB2 は SMS 表スペース中の LOB データに対する S (共用) ロックを取得します。

INCREMENTAL
累積 (増分) バックアップ・イメージを指定します。増分バックアップ・イメージとは、正常に実行された全バックアップ操作のうち最新のものが実行されて以来変更された、すべてのデータベース・データのコピーです。

DELTA
非累積 (差分) バックアップ・イメージを指定します。差分バックアップ・イメージとは、正常に実行された任意のタイプのバックアップ操作のうち最新のものが実行されて以来変更された、すべてのデータベース・データのコピーです。

USE TSM
バックアップに Tivoli Storage Manager 出力を使用することを指定します。

USE XBSA
XBSA インターフェースを使用するように指定します。バックアップ・サービス API (XBSA) は、バックアップやアーカイブの目的でデータ・ストレージ管理を必要とするアプリケーションまたは機能のための、オープン・アプリケーション・プログラミング・インターフェースです。

OPTIONS

"options-string"
バックアップ操作で使用するオプションを指定します。このストリングは、TSM などのベンダー・サポートのライブラリーに、引用符なしで入力された場合とまったく同じように渡されます。

注: このオプションを指定すると、 VENDOROPT データベース構成パラメーターによって指定されている値がオーバーライドされます。

@*file-name*
バックアップ操作で使用するオプションが、 DB2 サーバー上のファイルに含まれていることを指定します。このストリングは、TSM などのベンダー・サポートのライブラリーに渡されます。ファイル名は完全修飾ファイル名でなければなりません。

OPEN num-sessions SESSIONS
DB2 と TSM または他のバックアップ・ベンダー製品との間で作成される入出力セッションの数。

注: このパラメーターは、テープ、ディスク、または他のローカル装置にバックアップする場合には効果はありません。

TO dir/dev
ディレクトリーまたはテープ装置名のリストです。ディレクトリーが常駐する絶対パス名を指定しなければなりません。 USE TSM、TO、および LOAD が省略される場合には、バックアップ・イメージ用のデフォルト・ターゲット・ディレクトリーはクライアント・コンピューターの現行作業ディレクトリーとなります。このターゲット・ディレクトリーまたは装置は、

データベース・サーバー上に存在している必要があります。このパラメーターは、バックアップ・イメージが複数の宛先ディレクトリーや装置にわたる場合に、それらを指定するために繰り返すことができます。宛先が複数指定されている場合 (たとえば、宛先 1、宛先 2、および宛先 3)、宛先 1 が最初にオープンされます。メディア・ヘッダーおよび特殊ファイル (構成ファイル、表スペース表、および履歴ファイルを含む) は、宛先 1 にあります。他の残りの宛先は、オープンされており、これらはバックアップ操作のときに並列で使用されます。 Windows オペレーティング・システムの場合、汎用テープ装置はサポートされていないので、テープ装置のタイプごとにユニークなデバイス・ドライバーが必要です。 Windows オペレーティング・システムの FAT ファイル・システムにバックアップを取るには、ユーザーは 8.3 命名規則に適合するようにしなければなりません。

テープ装置やフロッピー・ディスクを使用することにより、メッセージやユーザー入力のプロンプトを生成できます。有効な応答オプションは、次のとおりです。

c 続行。警告メッセージを生成した装置の使用を続けます (たとえば、新しいテープをマウントしたときなど)。

d 装置の終了。警告メッセージの原因となった装置の使用だけ を停止します (たとえば、これ以上テープがない場合など)。

t 終了。バックアップ操作を打ち切ります。

テープ・システムでバックアップ・イメージを固有に参照する機能をサポートしていない場合は、同じテープに同じデータベースの複数のバックアップ・コピーは作成しないことをお勧めします。

LOAD library-name

使用するバックアップおよびリストア I/O ベンダー関数を含む共有ライブラリー (Windows オペレーティング・システムでは DLL) の名前。絶対パスで指定することができます。絶対パスを指定していない場合、デフォルトはユーザー出口プログラムが常駐しているパスになります。

WITH num-buffers BUFFERS

使用するバッファーの数です。値を明示的に指定しない場合、 DB2 はこのパラメーターのための最適値を自動的に選択します。ただし、バックアップを複数の場所に作成する場合は、パフォーマンスを向上させるために多数のバッファーを使用することができます。

BUFFER buffer-size

4 KB ページごとの単位で表した、バックアップ・イメージを作成する際に使用するバッファーのサイズ。値を明示的に指定しない場合、 DB2 はこのパラメーターのための最適値を自動的に選択します。このパラメーターの最小値は 8 ページです。

さまざまなブロック・サイズのテープを使用する場合は、磁気テープ装置がサポートする範囲内にバッファー・サイズを削減してください。この範囲内でないと、バックアップ操作は正常に実行されることもありますが、作成されたイメージはリカバリー不能になります。

SCO UnixWare 7 上で磁気テープ装置を使用するときは、バッファー・サイズを 16 に指定します。

Linux のほとんどのバージョンでは、SCSI テープ装置でバックアップ操作を行うときに、 DB2 のデフォルトのバッファー・サイズを使用すると、エラー SQL2025N、理由コード 75 が表示されます。 Linux 内部 SCSI バッファーがオーバーフローするのを防ぐには、以下の公式を使用してください。

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

bufferpages は BUFFER パラメーターと共に使用する値であり、 ST_MAX_BUFFERS と ST_BUFFER_BLOCKS は drivers/scsi ディレクトリー中の Linux カーネルで定義されています。

PARALLELISM n
バックアップ・ユーティリティーによって同時に読み取り可能な表スペースの数を決定します。値を明示的に指定しない場合、 DB2 はこのパラメーターのための最適値を自動的に選択します。

UTIL_IMPACT_PRIORITY *priority*
バックアップを、指定した優先順位によりスロットル・モードで実行することを指定します。スロットル・モードでは、バックアップ操作によるパフォーマンスの影響を調整できます。優先順位 (priority) は 1 から 100 までの範囲の任意の数であり、 1 が優先順位最低、100 が優先順位最高を意味します。優先順位の値なしで UTIL_IMPACT_PRIORITY キーワードが指定された場合は、デフォルトの優先順位 50 でバックアップが実行されます。 UTIL_IMPACT_PRIORITY を指定しない場合、バックアップは非スロットル・モードで実行されます。バックアップをスロットル・モードで実行するためには、 *util_impact_lim* 構成パラメーターを設定することによって影響ポリシーが定義されていなければなりません。

COMPRESS
バックアップを圧縮することを指定します。

COMPRLIB *name*
圧縮を実行するために使用するライブラリーの名前。この名前は、サーバー上の 1 個のファイルを参照する完全修飾パスでなければなりません。このパラメーターを指定しない場合、デフォルトの DB2 圧縮ライブラリーが使用されます。指定されたライブラリーをロードできない場合、バックアップは失敗します。

EXCLUDE
圧縮ライブラリーをバックアップ・イメージに格納しないことを指定します。

COMPROPTS *string*
バイナリー・データのうち、圧縮ライブラリーの初期設定ルーチンに渡すブロックを記述します。 DB2 はこのストリングをクライアントからサーバーに直接渡すため、バイト反転やコード・ページ変換の問題がある場合は圧縮ライブラリーで処理する必要があります。データ・ブロックの最初の文字が '@' なら、データの残りの部分は、サーバー上に存在するファイルの名前を指定するものとして解釈されます。その場合 DB2 は、 string の内容をそのファイルの内容で置き換え、そのようにして得られる新しい値を初期設定ルーチンに渡します。 *string* の最大長は 1024 バイトです。

EXCLUDE LOGS
バックアップ・イメージにログ・ファイルをまったく含めないことを指定します。

注: オフライン・バックアップ操作の実行の場合、このオプションが指定されていてもいなくても、ログは除外されます。

INCLUDE LOGS
ログ・ファイルのうち、特定の整合ポイント・イン・タイムまでこのイメージをリストアおよびロールフォワードするために必要な範囲をバックアップ・イメージに含めることを指定します。オフライン・バックアップの場合、このオプションは無効です。

WITHOUT PROMPTING
バックアップは、管理されることなく実行されるため、通常はユーザーの介入を必要とするアクションでエラー・メッセージが戻されるように指定されます。

例:

1. 以下の例で、データベース WSDB は 0 から 3 までの番号が付けられた 4 つのパーティションすべてに定義されています。パス /dev3/backup はすべてのパーティションからアクセスできます。パーティション 0 はカタログ・パーティションであり、これはオフライン・バックアップなので別個にバックアップする必要があります。すべての WSDB データベース・パーティションの /dev3/backup へのオフライン・バックアップを実行するには、データベース・パーティションの 1 つから以下のコマンドを出します。

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

   2 番目のコマンドで、db2_all ユーティリティーは同じバックアップ・コマンドを各データベース・パーティションに順番に出します (パーティション 0 を除く)。 4 つのデータベース・パーティションのバックアップ・イメージはすべて、 /dev3/backup ディレクトリーに保管されます。

2. 以下の例で、データベース SAMPLE は TSM サーバーに 2 つの並行 TSM クライアント・セッションを使用してバックアップされます。この環境に最適なバッファー・サイズが DB2 によって計算されます。

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

3. 次の例では、データベース payroll の表スペース (syscatspace、userspace1) の表スペース・レベル・バックアップがテープに対して実行されます。

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

4. バックアップ操作で使用する TSM 情報を指定するには、 USE TSM OPTIONS キーワードを使用します。次の例は、 USE TSM OPTIONS キーワードを使用して、完全修飾ファイル名を指定する方法を示すものです。

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

   ファイル myoptions.txt には、 -fromnode=bar -fromowner=dmcinnis というストリングが含まれています。

5. 以下は、リカバリー可能データベース用の増分バックアップの週間予定のサンプルです。週 1 回の全データベース・バックアップ操作、1 日 1 回の非累積 (差分) バックアップ操作、および週 2 回の累積 (増分) バックアップ操作が含まれています。

```
(Sun) db2 backup db sample use tsm
(Mon) db2 backup db sample online incremental delta use tsm
(Tue) db2 backup db sample online incremental delta use tsm
(Wed) db2 backup db sample online incremental use tsm
(Thu) db2 backup db sample online incremental delta use tsm
(Fri) db2 backup db sample online incremental delta use tsm
(Sat) db2 backup db sample online incremental use tsm
```

6. 次の例では、データベース SAMPLE のバックアップ操作のために、同一のターゲット・ディレクトリーを 3 回指定しています。ターゲット・ファイル・システムが複数の物理ディスクで構成されている場合には、この方法を使用することができるでしょう。

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

   データは 3 つのターゲット・ディレクトリーに並行してバックアップされ、それら 3 つのバックアップ・イメージは拡張子 .001、 .002、および .003 が付けられて生成されます。

関連資料:

- 102 ページの『RESTORE DATABASE』
- 143 ページの『ROLLFORWARD DATABASE』

db2Backup - データベースのバックアップ

データベースまたは表スペースのバックアップ・コピーを作成します。

有効範囲:

この API は、それが実行されるデータベース・パーティションにのみ影響を与えます。

権限:

以下のいずれかが必要です。

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

必要な接続:

データベース。この API を呼び出せば、指定したデータベースへの接続が自動的に確立されます。

接続はバックアップの完了時に終了します。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

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

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

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

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

汎用 API 構文:

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

typedef SQL_STRUCTURE db2gBackupStruct
{
  char                         *piDBAlias;
  db2Uint32                    iDBAliasLen;
  char                         *poApplicationId;
  db2Uint32                    iApplicationIdLen;
  char                         *poTimestamp;
  db2Uint32                    iTimestampLen;
```

```
  struct db2gTablespaceStruct *piTablespaceList;
  struct db2gMediaListStruct  *piMediaList;
  char                        *piUsername;
  db2Uint32                   iUsernameLen;
  char                        *piPassword;
  db2Uint32                   iPasswordLen;
  void                        *piVendorOptions;
  db2Uint32                   iVendorOptionsSize;
  db2Uint32                   oBackupSize;
  db2Uint32                   iCallerAction;
  db2Uint32                   iBufferSize;
  db2Uint32                   iNumBuffers;
  db2Uint32                   iParallelism;
  db2Uint32                   iOptions;
  db2Uint32                   iUtilImpactPriority;
  char                        *piComprLibrary;
  db2Uint32                   iComprLibraryLen;
  void                        *piComprOptions;
  db2Uint32                   iComprOptionsSize;
} db2gBackupStruct;

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

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

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

API パラメーター:

versionNumber
入力。 2 番目のパラメーター *pDB2BackupStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2BackupStruct
入力。 *db2BackupStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

piDBAlias
入力。バックアップをとるデータベースのデータベース別名 (システム・データベース・ディレクトリーにカタログされている) を含むストリングを指定します。

iDBAliasLen
入力。データベースの別名の長さを示す 4 バイトの符号なし整数 (バイト単位)。

oApplicationId
出力。 API によって、アプリケーションにサービスを提供しているエージェントを識別するストリングが戻されます。データベース・モニターを使用するバックアップ操作の進行状況に関する情報を取得することもできます。

poApplicationId
出力。長さ SQLU_APPLID_LEN+1 (sqlutil.h で定義) のバッファーを提供します。 API によって、アプリケーションにサービスを提供しているエージェントを識別するストリングが戻されます。データベース・モニターを使用するバックアップ操作の進行状況に関する情報を取得することもできます。

iApplicationIdLen
入力。 *poApplicationId* バッファーの長さを示す 4 バイトの符号なし整数 (バイト単位)。 SQLU_APPLID_LEN+1 (sqlutil.h に定義) と等しくなければなりません。

oTimestamp
出力。 API によって、バックアップ・イメージのタイム・スタンプが戻されます。

poTimestamp
出力。長さ SQLU_TIME_STAMP_LEN+1 (sqlutil.h で定義) のバッファーを提供します。 API によって、バックアップ・イメージのタイム・スタンプが戻されます。

iTimestampLen
入力。 *poTimestamp* バッファーの長さを示す 4 バイトの符号なし整数 (バイト単位)。 SQLU_TIME_STAMP_LEN+1 (sqlutil.h で定義) と等しくする必要があります。

piTablespaceList
入力。バックアップをとる表スペースのリストです。表スペース・レベルのバックアップの場合にのみ必要です。データベース・レベルのバックアップの場合は NULL に設定しなければなりません。 DB2TablespaceStruct 構造を参照してください。

piMediaList
入力。この構造を使用することにより、呼び出し側はバックアップ操作の宛先を指定することができます。提供される情報は、*locationType* パラメーターの値によって異なります。 *locationType* に有効な値 (sqlutil.h で定義) は、以下のとおりです。

SQLU_LOCAL_MEDIA
ローカル装置 (テープ、ディスクまたはディスケットの組み合わせ)。

SQLU_TSM_MEDIA
TSM。ロケーション・ポインターが NULL に設定されている場合、 DB2 で提供される TSM 共用ライブラリーが使用されます。別のバージョンの TSM 共用ライブラリーが必要な場合には、SQLU_OTHER_MEDIA を使用し、共用ライブラリー名を入力してください。

SQLU_OTHER_MEDIA
ベンダー製品。ロケーション・フィールド内の共用ライブラリー名を提供します。

SQLU_USER_EXIT
ユーザー出口。追加の入力は必要ありません (サーバーが OS/2 上にある場合のみ使用可能です)。

詳しくは、*db2MediaListStruct* 構造を参照してください。

piUsername
入力。接続の試行時に使用されるユーザー名を含むストリングを指定します。 NULL にすることもできます。

iUsernameLen
入力。ユーザー名の長さを示す 4 バイトの符号なし整数 (バイト単位) です。ユーザー名が提供されていない場合は、ゼロに設定してください。

piPassword
入力。ユーザー名とともに使用されるパスワードを含むストリングを指定します。 NULL にすることもできます。

iPasswordLen
入力。パスワードの長さを示す 4 バイトの符号なし整数 (バイト単位) です。パスワードが提供されていない場合は、ゼロに設定してください。

piVendorOptions
入力。情報をアプリケーションからベンダー関数へ渡すのに使用されます。このデータ構造はフラットでなければなりません。つまり、間接のレベルはサポートされません。このデータについては、バイト反転が行われず、また、コード・ページがチェックされないことに注意してください。

iVendorOptionsSize
入力。 *piVendorOptions* フィールドの長さです。65535 バイト以下でなければなりません。

oBackupSize
出力。バックアップ・イメージのサイズ (MB バイト単位) を示します。

iCallerAction
入力。実行するアクションを指定します。有効な値 (`db2ApiDf.h` で定義) は、以下のとおりです。

DB2BACKUP_BACKUP
バックアップを開始します。

DB2BACKUP_NOINTERRUPT
バックアップを開始します。バックアップを自動実行するよう指定します。通常ユーザーの介入を要求するシナリオは、呼び出し側への最初の戻りなしに試行されるか、エラーを生成します。この呼び出し側アクションは、たとえば、バックアップに必要なメディアがすべてマウントされていることが明らかで、ユーティリティーによるプロンプトが必要とされない場合に使用してください。

DB2BACKUP_CONTINUE
ユーザーがユーティリティーによって要求された何らかのアクション (たとえば、新しいテープのマウント) を実行した後で、バックアップを継続します。

DB2BACKUP_TERMINATE
ユーザーがユーティリティーによって要求された何らかのアクションの実行に失敗した場合、バックアップを終了します。

DB2BACKUP_DEVICE_TERMINATE
バックアップに使用される装置のリストから特定の装置を除外します。特定のメディアがいっぱいになると、バックアップは呼び出し側に警告を戻します (一方、残りの装置を使用して処理を継続します)。その場合、この呼び出し側アクションを指定してバックアップを再び呼び出すことによって、警告が生成される原因となった装置を使用装置のリストから除外してください。

DB2BACKUP_PARM_CHK
バックアップを実行することなく、パラメーターの妥当性を検査します。このオプションは、呼び出しが戻った後でデータベース接続を終了しません。この呼び出しが正常に戻った後、ユーザーが SQLUB_CONTINUE で呼び出しを発行し、処置を進めることが期待されます。

DB2BACKUP_PARM_CHK_ONLY
バックアップを実行することなく、パラメーターの妥当性を検査します。この呼び出しが戻る前に、この呼び出しによって確立したデータベース接続は終了し、後続する呼び出しは必要なくなります。

iBufferSize
入力。バッファー・サイズを 4 KB の割り振り単位 (ページ) でバックアップします。最小値は 8 単位です。

iNumBuffers
入力。使用するバックアップ・バッファーの数を指定します。最小値は 2 です。最大値はメモリーによって制限されます。

iParallelism
入力。並列処理の度合い (バッファー・マニピュレーターの数) を指定します。最小値は 1 です。最大値は 1024 です。

iOptions
入力。バックアップ・プロパティーのビットマップ。オプションは組み合わされて、ビット単位 OR 演算子を使用して *iOptions* の値を生成します。有効な値 (db2ApiDf.h で定義) は、以下のとおりです。

DB2BACKUP_OFFLINE
オフラインで、データベースへの排他的接続が確立されます。

DB2BACKUP_ONLINE
オンラインで、バックアップ操作の実行中に他のアプリケーションがデータベースにアクセスできるようになります。

注: ユーザーが SMS LOB データに対するロックを保持している場合は、オンライン・バックアップ操作は停止しているようにみえる場合があります。

DB2BACKUP_DB
データベースの全バックアップ。

DB2BACKUP_TABLESPACE
表スペース・レベルのバックアップ。表スペース・レベルのバックアップの場合は、 *piTablespaceList* パラメーターに表スペースのリストを提供してください。

DB2BACKUP_INCREMENTAL
累積 (増分) バックアップ・イメージを指定します。増分バックアップ・イメージとは、正常に実行された全バックアップ操作のうち最新のものが実行されて以来変更された、すべてのデータベース・データのコピーです。

DB2BACKUP_DELTA
非累積 (差分) バックアップ・イメージを指定します。差分バックアップ・イメージとは、正常に実行された任意のタイプのバックアップ操作のうち最新のものが実行されて以来変更された、すべてのデータベース・データのコピーです。

DB2BACKUP_COMPRESS
バックアップを圧縮することを指定します。

DB2BACKUP_INCLUDE_COMPR_LIB
バックアップの圧縮に使用するライブラリーがバックアップ・イメージに含まれることを指定します。

DB2BACKUP_EXCLUDE_COMPR_LIB
バックアップの圧縮に使用するライブラリーがバックアップ・イメージに含まれないことを指定します。

DB2BACKUP_INCLUDE_LOGS
ログ・ファイルのうち、特定の整合ポイント・イン・タイムまでこのイメージをリストアおよびロールフォワードするために必要な範囲もバックアップ・イメージに含めることを指定します。オフライン・バックアップまたは複数パーティション・バックアップの場合、このオプションは無効です。

DB2BACKUP_EXCLUDE_LOGS
バックアップ・イメージにログ・ファイルをまったく含めないことを指定します。

注: オフライン・バックアップ操作の実行の場合、このオプションが指定されていてもいなくても、ログは除外されます。

iUtilImpactPriority
バックアップ時に使用される優先順位の値を指定します。優先順位の値は 0 から 100 までの範囲の任意の数であり、 0 が非スロットル、100 が優先順位最高を意味します。

piComprLibrary
入力。バックアップ・イメージの圧縮を実行するために使用する外部ライブラリーの名前を示します。この名前は、サーバー上の 1 個のファイルを参照する完全修飾パスでなければなりません。値が NULL ポインターであるか、空ストリングへのポインターである場合、 DB2 は、圧縮のためにデフォルトのライブラリーを使用します。指定されたライブラリーが見付からない場合、バックアップは失敗します。

piComprLibraryLen
入力。 piComprLibrary で指定したライブラリー名の長さを示す 4 バイトの符号なし整数 (バイト単位) です。ライブラリー名が提供されていない場合は、ゼロに設定してください。

piComprOptions
入力。バイナリー・データのうち、圧縮ライブラリーの初期設定ルーチンに渡すブロックを記述します。 DB2 はこのストリングをクライアントからサーバーに直接渡すため、バイト反転やコード・ページ変換の問題がある場合は圧縮ライブラリーで処理する必要があります。データ・ブロックの最初の文字が '@' なら、データの残りの部分は、サーバー上に存在するファイルの名前を指定するものとして解釈されます。その場合 DB2 は、piComprOptions および iComprOptionsSize の内容をそのファイルの内容およびサイズで置き換え、そのようにして得られる新しい値を初期設定ルーチンに渡します。

iComprOptionsSize
入力。 piComprOptions として渡されるデータ・ブロックのサイズを表す 4 バイトの符号なし整数。 piComprOptions が NULL ポインターである場合に限り、iComprOptionsSize はゼロになります。

tablespaces
バックアップを取る表スペースのリストを指すポインター。 C の場合、リストは NULL で終了するストリングです。一般的には、*db2Char* 構造のリストです。

numTablespaces
tablespaces パラメーター内の項目数。

locations
メディア・ロケーションのリストを指すポインター。 C の場合、リストは NULL で終了するストリングです。一般的には、db2Char 構造のリストです。

numLocations
locations パラメーター内の項目数。

locationType
メディア・タイプを示す文字。有効な値 (`sqlutil.h` で定義) は、以下のとおりです。

SQLU_LOCAL_MEDIA
ローカル装置 (テープ、ディスク、ディスケット、または名前付きパイプ)

SQLU_TSM_MEDIA
Tivoli Storage Manager。

SQLU_OTHER_MEDIA
ベンダー・ライブラリー。

SQLU_USER_EXIT
ユーザー出口 (サーバーが OS/2 上にある場合のみ使用可能です)。

pioData
文字データ・バッファーを指すポインター。

iLength
入力。 *pioData* バッファーのサイズ。

oLength
出力。将来の利用のために予約されています。

関連資料:

- 「*管理 API リファレンス*」の『sqlemgdb - データベースの移行』
- 153 ページの『db2Rollforward - データベースのロールフォワード』
- 「*管理 API リファレンス*」の『SQLCA』
- 113 ページの『db2Restore - データベースのリストア』

関連サンプル:

- 『dbrecov.sqc -- How to recover a database (C)』
- 『dbrecov.sqC -- How to recover a database (C++)』

バックアップ・セッション - CLP の例

例 1

以下の例で、データベース SAMPLE は、2 つの並行 TSM クライアント・セッションを使用して、 TSM サーバーにバックアップされます。バックアップ・ユーティリティーは、4 つのバッファーを使用します。バッファーの最適サイズ (4 KB ページ単位) は、使用可能なメモリー量および宛先装置の数に基づいて、自動的に計算されます。並列処理設定も計算されますが、これは使用可能なプロセッサー数とバックアップ予定の表スペースの数に基づきます。

```
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
```

例 2

以下は、リカバリー可能データベース用の増分バックアップの週間予定のサンプルです。週 1 回の全データベース・バックアップ操作、1 日 1 回の非累積 (差分) バックアップ操作、および週 2 回の累積 (増分) バックアップ操作が含まれています。

```
(Sun) db2 backup db kdr use tsm
(Mon) db2 backup db kdr online incremental delta use tsm
(Tue) db2 backup db kdr online incremental delta use tsm
(Wed) db2 backup db kdr online incremental use tsm
```

```
(Thu) db2 backup db kdr online incremental delta use tsm
(Fri) db2 backup db kdr online incremental delta use tsm
(Sat) db2 backup db kdr online incremental use tsm
```

例 3

Windows 環境で、磁気テープ装置へのバックアップ操作を開始するには、次のコマンドを発行します。

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

関連タスク:

- 72 ページの『バックアップの使用』

バックアップのパフォーマンスの最適化

バックアップ操作を実行すると、DB2® は、バッファー数、バッファー・サイズ、および並列処理設定の最適値を自動的に選択します。この値は、使用可能なユーティリティー・ヒープ・メモリーの大きさ、使用可能なプロセッサー数、およびデータベース構成に基づきます。目的は、バックアップ操作の完了にかかる時間を最小限に抑えることです。次の BACKUP DATABASE コマンド・パラメーターの値を明示的に入力しないと、DB2 側が値を選択します。

- WITH num-buffers BUFFERS
- PARALLELISM n
- BUFFER buffer-size

データベース・マネージャー構成パラメーター BACKBUFSZ および RESTBUFSZ によって指定される値は無視されます。これらの値を使用するのであれば、BACKUP DATABASE コマンドを発行するときに、バッファー・サイズを明示的に指定する必要があります。

さらに、バックアップ操作を完了するために必要な時間を短縮するために、以下のいずれかを実行することを選択できます。

- 表スペース・バックアップを指定します。

  BACKUP DATABASE コマンドの TABLESPACE オプションを使用して、データベースの一部にバックアップを実行し、後でリカバリーすることも可能です。この作業を行うと、表データ、索引、および長形式フィールドやラージ・オブジェクト (LOB) のデータを別個の表スペースで管理しやすくなります。
- バックアップされる表スペースの数を反映するよう、 BACKUP DATABASE コマンドの PARALLELISM パラメーターの値を増やします。

  PARALLELISM パラメーターは、データベースからデータを読み取るときや圧縮バックアップ操作時にデータの圧縮を実行するときに開始されるプロセッサー数またはスレッド数を定義します。各処理またはスレッドは特定の表スペースに割り当てられるため、PARALLELISM パラメーターに、バックアップされる表スペースの数よりも大きい値を指定する利点はありません。処理またはスレッドが、表スペースのバックアップを終了させると、別の表スペースを要求します。しかし、それぞれの処理またはスレッドでは、メモリーと CPU の両方のオーバーヘッドが必要になることに注意してください。

- バックアップ・バッファー・サイズを大きくします。

  理想的なバックアップ・バッファー・サイズは、表スペースのエクステント・サイズの倍数に 1 ページを加えたものです。複数の表スペースがありそれぞれエクステント・サイズが異なる場合は、エクステント・サイズの公倍数に 1 ページを加えた値を指定します。
- バッファー数を増やします。

  少なくともバックアップ先 (またはセッション) の 2 倍のバッファーを使用し、バックアップ先装置がデータを待つ状態にならないようにします。
- 複数のターゲット装置を使用します。

関連概念:

- 69 ページの『バックアップの概要』

関連タスク:

- 72 ページの『バックアップの使用』

第 3 章 データベースのリストア

以下に、DB2 UDB リストア・ユーティリティーについて説明します。このユーティリティーは、事前にバックアップを取ったデータベースや表スペースが損傷したり破壊されたりした場合の再作成に使用します。

以下のトピックについて説明します。

- 『リストアの概要』
- 96 ページの『リストアの使用に必要な特権、権限、および許可』
- 96 ページの『リストアの使用』
- 98 ページの『テストおよび実稼働環境における増分リストアの使用』
- 100 ページの『リストア操作 (リダイレクト・リストア) 時の表スペース・コンテナーの再定義』
- 101 ページの『既存データベースへのリストア』
- 102 ページの『新規データベースへのリストア』
- 102 ページの『RESTORE DATABASE』
- 113 ページの『db2Restore - データベースのリストア』
- 125 ページの『リストア・セッション - CLP の例』
- 128 ページの『リストアのパフォーマンスの最適化』

リストアの概要

DB2® RESTORE DATABASE コマンドの最も単純な形式の場合、必要なのは、リストアしたいデータベースの別名を指定することだけです。たとえば、次のようにします。

```
db2 restore db sample
```

この例では、SAMPLE データベースが存在し、RESTORE DATABASE コマンドを発行すると置き換えられるので、以下のメッセージが戻されます。

```
SQL2539W  Warning!  Restoring to an existing database that is the same as
the backup image database.  The database files will be deleted.
Do you want to continue ? (y/n)
```

y を指定すると、リストア操作は正常に完了するはずです。

データベースをリストア操作する際には、排他モードで接続します。したがって、操作開始時に、データベースに対してアプリケーションが実行されていてはなりません。また、リストア・ユーティリティーを実行すると、リストア操作が正常に完了するまで、他のアプリケーションからデータベースにアクセスできなくなります。ただし、表スペースのリストア操作は、オンラインで行うことができます。

(ロールフォワード・リカバリー後の) リストア操作が正常に完了するまで、表スペースは使用できません。

複数の表スペースにまたがっている表がある場合、その表スペースの集合を一緒にバックアップおよびリストアする必要があります。

部分またはサブセット・リストア操作を実行するときは、表スペース・レベルのバックアップ・イメージを使用するか、全データベース・レベルのバックアップ・イメージを使用してそのイメージから 1 つまたは複数の表スペースを選択できます。バックアップ・イメージ作成時から表スペースに関連付けられているすべてのログ・ファイルが、存在している必要があります。

関連概念:

- 58 ページの『ログ・ファイルをバックアップ・イメージに含める』

リストアの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャー保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスが制御されます。ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

全データベースのバックアップから*既存の* データベースにリストアするときには、SYSADM、SYSCTRL、または SYSMAINT の権限が必要です。*新規の* データベースにリストアするには、 SYSADM または SYSCTRL 権限が必要です。

リストアの使用

前提条件:

既存の データベースにリストアする場合は、リストアしようとしているデータベースに接続しないでください。指定したデータベースへの接続はリストア・ユーティリティーにより自動的に確立され、この接続はリストア操作が完了すると終了します。*新規の* データベースにリストアする場合は、データベースを作成するには、インスタンス・アタッチが必要です。*新規のリモート・データベースにリストアする場合は、まず新規データベースを置くインスタンスにアタッチしなければなりません。次に、サーバーのコード・ページとテリトリーを指定して、新しいデータベースを作成してください。

データベースには、ローカル・エージェントとリモート・エージェントがあります。

制限:

リストア・ユーティリティーには、以下の制限が適用されます。

- リストア・ユーティリティーが使用できるのは、事前に DB2 バックアップ・ユーティリティーを使って、データベースのバックアップをとってある場合に限ります。

- ロールフォワード処理の実行中にデータベースのリストア操作を開始することはできません。
- 表スペースが存在していて、しかも同一表スペースである場合 (「同一」とは、バックアップを取ってから再作成を行うまでの間に表スペースをドロップして再作成していないこと) に限り、表スペースのリストアを行うことができます。
- 新規のデータベースに表スペース・レベルのバックアップをリストアすることはできません。
- システム・カタログ表が関係している表スペース・レベルのリストア操作をオンラインで実行することはできません。

手順:

リストア・ユーティリティーは、コマンド行プロセッサー (CLP)、コントロール・センターにあるデータベース・リストア・ノートブックまたはウィザード、または **db2Restore** アプリケーション・プログラミング・インターフェース (API) を通して起動できます。

CLP によって発行する RESTORE DATABASE コマンドの例を以下に示します。

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

「データベースのリストア (Restore Database)」ノートブックまたはウィザードをオープンするには、次のようにします。

1. コントロール・センターから、「データベース (Databases)」フォルダーが見つかるまでオブジェクト・ツリーを展開します。
2. 「データベース (Databases)」フォルダーをクリックします。ウィンドウの右側部分 (目次ペイン) に、既存のデータベースがすべて表示されます。
3. 目次ペイン内で対象となるデータベースをマウスの右ボタンでクリックし、ポップアップ・メニューから「データベースのリストア (Restore Database)」または「ウィザードを使用したデータベースのリストア (Restore Database Using Wizard)」を選択します。「データベースのリストア (Restore Database)」ノートブックまたは「データベースのリストア (Restore Database)」ウィザードがオープンします。

詳しい情報については、コントロール・センターのオンライン・ヘルプ機能をご覧ください。

関連概念:

-
- 「*管理ガイド: インプリメンテーション*」の『コントロール・センター用のプラグイン・アーキテクチャーの紹介』

関連資料:

- 113 ページの『db2Restore - データベースのリストア』

テストおよび実稼働環境における増分リストアの使用

プロダクション・データベースの増分バックアップとリカバリーを使用可能にした後は、増分または差分バックアップ・イメージを使用してテスト用データベースを作成またはリフレッシュできます。手動または自動増分リストアを使用してこのことを行えます。バックアップ・イメージをプロダクション・データベースからテスト用データベースにリストアするには、 RESTORE DATABASE コマンド上で INTO *target-database-alias* オプションを使用してください。たとえば、以下のバックアップ・イメージを持つプロダクション・データベースがあるとします。

```
backup db prod
Backup successful. The timestamp for this backup image is : <ts1>

backup db prod incremental
Backup successful. The timestamp for this backup image is : <ts2>
```

以下は、手動増分リストアの 1 例です。

```
restore db prod incremental taken at <ts2> into test without
prompting
DB20000I  The RESTORE DATABASE command completed successfully.

restore db prod incremental taken at <ts1> into test without
prompting
DB20000I  The RESTORE DATABASE command completed successfully.

restore db prod incremental taken at <ts2> into test without
prompting
DB20000I  The RESTORE DATABASE command completed successfully.
```

データベース TEST がすでにある場合には、リストア操作により、すでにそこにあるすべてのデータは上書きされます。データベース TEST がない場合には、リストア・ユーティリティーは、そのデータベースを作成した後、バックアップ・イメージからのデータを用いてそれを移植します。

自動増分リストア操作はデータベース履歴に依存しているため、テスト用データベースがあるかどうかによってリストアのステップは多少変わります。データベース TEST への自動増分リストアを実行するためには、その履歴がデータベース PROD のバックアップ・イメージ履歴を含んでいなくてはなりません。バックアップ・イメージのデータベース履歴は、以下の時に、データベース TEST 用にすでにある任意のデータベース履歴を置き換えます。

- RESTORE DATABASE コマンドが出された時にデータベース TEST が存在しない。または、
- RESTORE DATABASE コマンドが出された時にデータベース TEST が存在するが、データベース TEST の履歴にレコードが含まれていない。

以下の例は、存在しないデータベース TEST への自動増分リストアを示しています。

```
restore db prod incremental automatic taken at <ts2> into test without
prompting
DB20000I  The RESTORE DATABASE command completed successfully.
```

リストア・ユーティリティーは、TEST データベースを作成し、それを移植します。

データベース TEST が存在し、データベース履歴が空でないならば、以下のようにして、自動増分リストア操作の前にデータベースをドロップする必要があります。

```
drop db test
DB20000I  The DROP DATABASE command completed successfully.

restore db prod incremental automatic taken at <ts2> into test without
prompting
DB20000I  The RESTORE DATABASE command completed successfully.
```

データベースをドロップしたくない場合には、RESTORE DATABASE コマンドを発行する前に、遠い将来に設定したタイム・スタンプを使用した PRUNE HISTORY コマンドと、WITH FORCE OPTION パラメーターを発行します。

```
connect to test
Database Connection Information

Database server       = <server id>
SQL authorization ID       = <id>
Local database alias       = TEST

prune history 9999 with force option
DB20000I  The PRUNE command completed successfully.

connect reset
DB20000I  The SQL command completed successfully.
restore db prod incremental automatic taken at <ts2> into test without
prompting
SQL2540W  Restore is successful, however a warning "2539" was
encountered during Database Restore while processing in No
Interrupt mode.
```

この場合、RESTORE DATABASE COMMAND は、データベース TEST が存在しなかった時と同様に機能します。

データベース TEST が存在し、データベース履歴が空の場合、自動増分リストア操作の前にデータベース TEST をドロップする必要はありません。

```
restore db prod incremental automatic taken at <ts2> into test without
prompting
SQL2540W  Restore is successful, however a warning "2539" was
encountered during Database Restore while processing in No
Interrupt mode.
```

フル・データベース・バックアップを最初に取らずに、テスト用データベースの増分または差分バックアップを取り続けることができます。とはいえ、増分または差分イメージの 1 つをリストアする必要がある場合には、手動増分リストアを実行する必要があるでしょう。これは、自動増分リストア中にリストアされるそれぞれのバックアップ・イメージが、同じデータベース別名から作成されることを自動増分リストア操作が要求しているためです。

実動バックアップ・イメージを使用してリストア操作を完了した後に、テスト用データベースのフル・データベース・バックアップを取った場合には、増分または差分バックアップを取ることができ、手動または自動モードのいずれかを使用してそれをリストアできます。

関連概念:

- 29 ページの『増分バックアップおよびリカバリー』

関連資料:

- 77 ページの『BACKUP DATABASE』
- 102 ページの『RESTORE DATABASE』
- 298 ページの『LIST HISTORY』

リストア操作 (リダイレクト・リストア) 時の表スペース・コンテナーの再定義

データベースのバックアップ操作時に、バックアップされる表スペースに関連しているすべての表スペース・コンテナーについて記録がとられます。リストア操作時に、バックアップ・イメージ中にリストされているすべてのコンテナーのチェックが行われ、存在していてアクセス可能かどうかが判別されます。メディア障害 (あるいは他の理由) によりこれらのコンテナーのうち 1 つまたは複数がアクセスできない場合は、リストア操作は失敗します。このような場合でもリストア操作を正常実行するには、別のコンテナーにリダイレクトする必要があります。 DB2® では、表スペース・コンテナーの追加、変更、除去がサポートされています。

表スペース・コンテナーを再定義するには、 RESTORE DATABASE コマンドを起動して REDIRECT パラメーターを指定するか、またはコントロール・センターの「データベースのリストア (Restore Database)」ノートブックの「コンテナー (Containers)」ページを使用します。増分バックアップのリダイレクト・リストアを起動するためのプロセスは、非増分バックアップ・イメージのプロセスに類似しています。 REDIRECT パラメーターとともに RESTORE DATABASE コマンドを呼び出し、データベースを増分でリストアするためのバックアップ・イメージを指定します。

リダイレクト・リストア操作中に、ディレクトリー・コンテナーおよびファイル・コンテナーは、存在していなければ自動的に作成されます。データベース・マネージャーは、装置コンテナーを自動的には作成しません。

コンテナーのリダイレクトにより、表スペース・コンテナーを管理するうえで非常に柔軟な対応ができます。たとえば、コンテナーを SMS 表スペースに追加することはサポートされていませんが、リダイレクト・リストア操作の起動時に追加コンテナーを指定すると、このことを行えます。

以下の例には、データベース SAMPLE でリダイレクトされたリストアを実行する方法が示されています。

```
db2 restore db sample redirect without prompting
SQL1277N Restore has detected that one or more table space containers are
inaccessible, or has set their state to 'storage must be defined'.
DB20000I  The RESTORE DATABASE command completed successfully.

db2 set tablespace containers for 2 using (path 'userspace1.0', path
'userspace1.1')
DB20000I The SET TABLESPACE CONTAINERS command completed successfully.

db2 restore db sample continue
DB20000I  The RESTORE DATABASE command completed successfully.
```

関連資料:

- 102 ページの『RESTORE DATABASE』
- 125 ページの『リストア・セッション - CLP の例』

関連サンプル:

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

既存データベースへのリストア

全データベースのバックアップ・イメージを、既存のデータベースにリストアすることもできます。バックアップ・イメージの別名、データベース名、またはデータベース・シードは、既存のデータベースとは異なる場合があります。

データベース・シード とは、データベースの持続期間中変更されない、そのデータベース固有の ID のことです。シードは、データベースの作成時にデータベース・マネージャーによって割り当てられます。 DB2® は、常にバックアップ・イメージからのシードを使用します。

既存データベースにリストアする場合は、リストア・ユーティリティーにより以下の機能が実行されます。

- 既存のデータベースから表、索引、および長形式フィールドのデータを削除し、バックアップのデータに置き換える。
- リストアされる表スペースごとに表項目を置き換える。
- リカバリー履歴ファイルを保持する (損傷していたり、項目がない場合を除く)。リカバリー履歴ファイルが損傷を受けているか、その中に項目がない場合は、データベース・マネージャーはバックアップ・イメージからファイルをコピーします。
- 既存データベースの認証タイプを保持する。
- 既存データベースのデータベース・ディレクトリーを保持する。このディレクトリーは、既存データベースの保存位置とカタログ作成の方法を定義します。
- データベース・シードを比較する。シードが異なる場合は、以下のことが実行されます。
  - 既存データベースに関連するログを削除する。
  - データベース構成ファイルをバックアップ・イメージからコピーする。
  - RESTORE DATABASE コマンドで NEWLOGPATH が指定されている場合、NEWLOGPATH を *logpath* データベース構成パラメーターの値に設定する。

  データベース・シードが同じである場合は、以下のことが実行されます。
  - イメージがリカバリー不能データベースのものである場合はログを削除する。
  - ファイルが壊れていない限り、現在のデータベース構成ファイルを保存する。壊れている場合は、バックアップ・イメージからファイルをコピーする。
  - RESTORE DATABASE コマンドで NEWLOGPATH が指定されている場合、NEWLOGPATH を *logpath* データベース構成パラメーターの値に設定する。それ以外の場合、現在のログ・パスをデータベース構成ファイルにコピーす

る。ログ・パスの妥当性検査が行われ、データベースがそのログ・パスを使用できない場合は、データベース構成を変更して、デフォルトのログ・パスを使用します。

新規データベースへのリストア

新規のデータベースを作成して、全データベースのバックアップ・イメージをそのデータベースにリストアすることもできます。新規データベースを作成しない場合には、リストア・ユーティリティーがそれを作成します。

新規データベースにリストアする場合は、リストア・ユーティリティーにより以下の機能が実行されます。

- ターゲット・データベースの別名パラメーターで指定されたデータベース別名を使用して、新規データベースを作成する。 (ターゲット・データベース別名が指定されていない場合は、リストア・ユーティリティーで、元のデータベース別名パラメーターで指定されているものと同じ別名のデータベースが作成されます。)
- データベース構成ファイルをバックアップ・イメージからリストアする。
- RESTORE DATABASE コマンドで NEWLOGPATH が指定されている場合、NEWLOGPATH を *logpath* データベース構成パラメーターの値に設定する。ログ・パスの妥当性検査が行われ、データベースがそのログ・パスを使用できない場合は、データベース構成を変更して、デフォルトのログ・パスを使用します。
- バックアップ・イメージから認証タイプをリストアする。
- バックアップ・イメージ中のデータベース・ディレクトリーから注釈をリストアする。
- データベースのリカバリー履歴ファイルをリストアする。

RESTORE DATABASE

DB2 バックアップ・ユーティリティーを使用してバックアップされた損傷のある、または破壊されたデータベースを再作成します。リストアされたデータベースは、バックアップ・コピーの作成時と同じ状態になります。さらにこのユーティリティーにより、データベースに異なるイメージを上書きしたり、新しいデータベースにリストアしたりすることもできます。

DB2 バージョン 8、32 ビット Windows 版で作成されたデータベースを、 DB2 バージョン 8、64 ビット Windows 版にリストアしたり、その逆にリストアしたりすることが可能です。 DB2 バージョン 8、32 ビット Linux (Intel) 版で作成されたデータベースを、 DB2 バージョン 8、64 ビット Linux (Intel) 版にリストアしたり、その逆にリストアしたりすることが可能です。 DB2 バージョン 8、AIX、HP-UX、または Solaris オペレーティング環境版 (32 ビットまたは 64 ビット) で作成されたデータベースを、 DB2 バージョン 8、AIX、HP-UX、または Solaris オペレーティング環境版 (32 ビットまたは 64 ビット) にリストアできます。

さらにリストア・ユーティリティーを使用すると、ワード・サイズ (32 ビットまたは 64 ビット) が同じである限り、 DB2 の旧バージョン (2 バージョン前まで) で作成されたバックアップ・イメージをリストアすることもできます。 DB2 の旧バ

ージョンで作成されたバックアップ・イメージのクロスプラットフォーム・リストア操作は、サポートされていません。移行が必要な場合、これはリストア操作の終了時に自動的に起動されます。

バックアップ操作のときに、データベースがすべてのロールフォワード・リカバリーに対して使用可能である場合、リストア操作が正常に完了した後に、ロールフォワード・ユーティリティーを起動することによって、データベースを損傷または破壊が起きる前の状態に戻すことができます。

このユーティリティーは、表スペース・レベルのバックアップをリストアすることもできます。

増分イメージと差分イメージは、オペレーティング・システムまたはワード・サイズ (32 ビットか 64 ビットか) が同じでない場合、リストアできません。

ある環境からそれとは異なる環境へのリストアの後は、非増分バックアップが実施されるまで、増分バックアップまたは差分バックアップは実行できません。 (同じ環境へのリストアの場合、この制限はありません。)

ある環境からそれとは異なる環境へのリストアが成功した場合も、いくらかの注意事項があります。パッケージは、使用する前に再バインドすることが必要です (BIND コマンド、 REBIND コマンド、または db2rbind ユーティリティーを使用)。 SQL プロシージャーは、ドロップしてから再作成する必要があります。また、外部ライブラリーは、新しいプラットフォーム上ですべて再ビルドする必要があります。 (同じ環境にリストアする場合、これらの点は該当しません。)

有効範囲:

このコマンドは、それが実行されたノードに対してだけ影響を与えます。

権限:

既存のデータベースにリストアするには、以下のいずれかが必要です。

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

新規データベースにリストアするには、以下のいずれかが必要です。

- *sysadm*
- *sysctrl*

必要な接続:

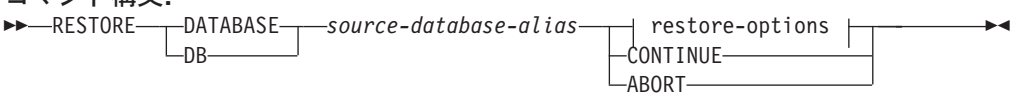
必要な接続は、実行するリストア・アクションの種類によって異なります。

- データベース (既存のデータベースにリストアする場合)。このコマンドは、指定されたデータベースへの排他接続を自動的に確立します。
- インスタンスおよびデータベース (新規データベースにリストアする場合)。データベースを作成するには、インスタンス・アタッチが必要です。

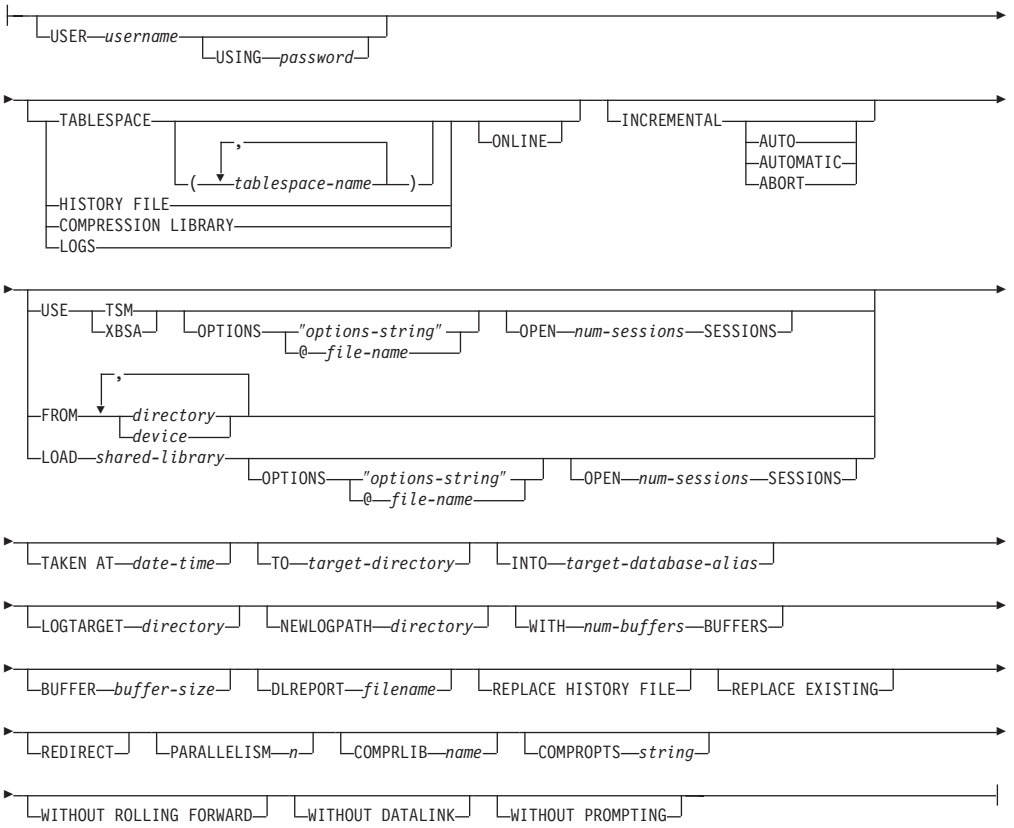
  現行のインスタンスとは異なるインスタンスで新規のデータベースへのリストアを行うには、まず、新規のデータベースを存在させるインスタンスにアタッチす

ることが必要です。新しいインスタンスは、ローカルの場合とリモートの場合が可能です。現在のインスタンスは、DB2INSTANCE 環境変数の値によって定義されます。

コマンド構文:



restore-options:



コマンド・パラメーター:

DATABASE *source-database-alias*
バックアップが取得されるソース・データベースの別名です。

CONTINUE
コンテナーが再定義されていること、およびリダイレクトしたリストア操作の最終ステップを実行する必要があることを指定します。

ABORT
このパラメーターは以下を指定します。

- リダイレクトしたリストア操作を停止します。これは、1 つまたは複数のステップを繰り返す必要があるエラーが発生したときに便利です。ABORT オプションを指定して RESTORE DATABASE を発行した後、REDIRECT オプションを指定した RESTORE DATABASE を含む、リダイレクトしたリストア操作の各ステップを繰り返す必要があります。
- 完了する前に増分リストア操作を終了します。

USER *username*
データベースがリストアされる際のユーザー名を識別します。

USING *password*
ユーザー名を認証するために使用するパスワード。パスワードを省略すると、ユーザーに入力を求めるプロンプトが出ます。

TABLESPACE *tablespace-name*
リストアされる表スペースを指定するときに使用する名前のリストです。

ONLINE
このキーワードは、表スペース・レベルのリストア操作を行う場合のみ適用でき、これを指定するとオンラインでバックアップ・イメージがリストアできます。これは、他のエージェントが、バックアップ・イメージのリストア中にデータベースに接続できることや、指定された表スペースのリストア中に他の表スペースのデータを使用できることを意味します。

HISTORY FILE
このキーワードは、バックアップ・イメージから履歴ファイルのみをリストアするのに指定されます。

COMPRESSION LIBRARY
このキーワードは、バックアップ・イメージから圧縮ライブラリーだけをリストアする場合に指定します。バックアップ・イメージの中にオブジェクトが存在している場合、それはデータベース・ディレクトリーの中にリストアされます。バックアップ・イメージの中にオブジェクトが存在していない場合、リストア操作は失敗します。

LOGS このキーワードは、バックアップ・イメージに含まれている一連のログ・ファイルだけをリストアする場合に指定します。バックアップ・イメージの中にログ・ファイルが含まれていない場合、リストア操作は失敗します。このオプションを指定する場合は、LOGTARGET オプションも指定する必要があります。

INCREMENTAL
INCREMENTAL は、追加のパラメーターを使用しないで手動累積リストア操作を指定します。手動リストアの際、ユーザーはリストアに含まれるイメージごとに各リストア・コマンドを手動で発行する必要があります。以下の順序でこれを行ってください。最後、1 番目、2 番目、3 番目、以下同様に最後のイメージまで。

INCREMENTAL AUTOMATIC/AUTO
自動累積リストア操作を指定します。

INCREMENTAL ABORT
手動累積リストア操作を指定します。

USE TSM
データベースが TSM 管理の出力からリストアされるように指定します。

OPTIONS

"options-string"
リストア操作で使用するオプションを指定します。このストリングは、TSM などのベンダー・サポートのライブラリーに、引用符なしで入力された場合とまったく同じように渡されます。

注: このオプションを指定すると、 VENDOROPT データベース構成パラメーターによって指定されている値がオーバーライドされます。

@*file-name*
リストア操作で使用するオプションが、 DB2 サーバー上のファイルに含まれていることを指定します。このストリングは、TSM などのベンダー・サポートのライブラリーに渡されます。ファイル名は完全修飾ファイル名でなければなりません。

OPEN *num-sessions* **SESSIONS**
TSM またはベンダー製品とともに使用する入出力セッションの数を指定します。

USE XBSA
XBSA インターフェースを使用するように指定します。バックアップ・サービス API (XBSA) は、バックアップやアーカイブの目的でデータ・ストレージ管理を必要とするアプリケーションまたは機能のための、オープン・アプリケーション・プログラミング・インターフェースです。

FROM *directory/device*
バックアップ・イメージがあるディレクトリーまたは装置の完全修飾パス名。 USE TSM、FROM、および LOAD を省略した場合のデフォルト値は、クライアント・マシンの現行作業ディレクトリーです。このターゲット・ディレクトリーまたは装置は、データベース・サーバー上に存在している必要があります。

Windows オペレーティング・システムでは、 DB2 が生成するディレクトリーを指定してはなりません。たとえば、次のようなコマンドを実行するとします。

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

これらのコマンドを使用すると DB2 は、 `c:¥backup` ディレクトリーの下にいくつかのサブディレクトリーを生成し、それにより指定されたトップレベルのディレクトリーの中に複数のバックアップを入れることができるようにします。 DB2 の生成するサブディレクトリーは無視してください。リストアするバックアップ・イメージを正確に指定するためには、 TAKEN AT パラメーターを使用します。複数のバックアップ・イメージを同じパスに保管することもできます。

複数の項目が指定され、項目の最後がテープ装置である場合には、他のテープが要求されます。有効な応答オプションは、次のとおりです。

c 続行。警告メッセージを生成した装置の使用を続けます (たとえば、新しいテープをマウントしたときなど)。

d 装置の終了。警告メッセージの原因となった装置の使用だけ を停止します (たとえば、もうテープがない場合に停止する、など)。

t 終了。ユーザーが、ユーティリティーによって要求された何らかのアクションを実行しなかった場合、リストア操作を異常終了します。

LOAD *shared-library*
使用するバックアップおよびリストア I/O ベンダー関数を含む共有ライブラリー (Windows オペレーティング・システムでは DLL) の名前。名前には絶対パスを含めることができます。絶対パスを指定しない場合、ユーザー出口プログラムが置かれているパスがデフォルト値として使われます。

TAKEN AT *date-time*
データベース・バックアップ・イメージのタイム・スタンプです。タイム・スタンプはバックアップ操作が正常に終了した後に表示され、バックアップ・イメージのパス名の一部になっています。 *yyyymmddhhmmss* の形式で指定されます。タイム・スタンプを部分的に指定することもできます。たとえば、2 つの異なるタイム・スタンプ 20021001010101 および 20021002010101 で指定されるバックアップ・イメージが存在する場合、`20021002` を指定することで、タイム・スタンプ `20021002010101` のイメージが使用できます。このパラメーターに値を指定しない場合は、ソース・メディア上のバックアップ・イメージは 1 つだけでなければなりません。

TO *target-directory*
ターゲット・データベース・ディレクトリーです。ユーティリティーが存在するデータベースへリストアしている場合には、このパラメーターは無視されます。指定するドライブおよびディレクトリーは、ローカルのものでなければなりません。

注: Windows オペレーティング・システムでは、このパラメーターを使用するときに、ドライブ名のみを指定してください。パスを指定すると、エラーが戻されます。

INTO *target-database-alias*
ターゲット・データベースの別名です。ターゲット・データベースが存在しない場合には、作成されます。

データベース・バックアップを既存のデータベースにリストアするとき、リストアされたデータベースは既存のデータベースの別名およびデータベース名を継承します。データベース・バックアップを存在していないデータベースにリストアするとき、新規のデータベースが指定した別名およびデータベース名を使用して作成されます。新規のデータベース名は、リストア先のシステムでユニークなものでなければなりません。

LOGTARGET *directory*
バックアップ・イメージからログ・ファイルを抽出する際のターゲット・ディレクトリーとして使用する、データベース・サーバー上の既存のディレクトリーの絶対パス名。このオプションを指定する場合、バックアップ・イメージ内のログ・ファイルは、そのターゲット・ディレクトリー内に抽出されます。このオプションを指定しない場合、バックアップ・イメージ内のログ・ファイルは抽出されません。バックアップ・イメージからログ・ファイルだけを抽出する場合は、 LOGS オプションを指定してください。

NEWLOGPATH *directory*
リストア操作後にアクティブ・ログに使用されるディレクトリーの絶対パス名。このパラメーターの機能は *newlogpath* データベース構成パラメーターと同じです。ただし、これが影響するのは、これを指定したリストア操作に限定されます。このパラメーターは、バックアップ・イメージのログ・パス

が、リストア後の使用に適していない場合に使用することができます。たとえば、パスがもはや有効でない、または別のデータベースによって使用されている、という場合などです。

WITH *num-buffers* **BUFFERS**
使用するバッファーの数です。値を明示的に指定しない場合、 DB2 はこのパラメーターのための最適値を自動的に選択します。複数のソースが読み取られる場合や、PARALLELISM の値が増やされている場合は、パフォーマンスを向上させるために複数のバッファーを使用することができます。

BUFFER *buffer-size*
リストア操作に使用するバッファーのサイズ (ページ数)。値を明示的に指定しない場合、 DB2 はこのパラメーターのための最適値を自動的に選択します。このパラメーターの最小値は 8 ページです。

リストア・バッファー・サイズは、バックアップ操作中に指定したバックアップ・バッファー・サイズに正の整数を乗算したサイズでなければなりません。誤ったバッファー・サイズを指定すると、許容可能な最小のサイズで割り振られます。

DLREPORT *filename*
ファイル名を指定する場合は、絶対パスとして指定しなければなりません。リストア操作中に高速調整が行われたためにリンク解除されたファイルを報告します。このオプションが使用されるのは、リストアする表に DATALINK 列タイプとリンク・ファイルが含まれている場合だけです。

REPLACE HISTORY FILE
リストア操作において、ディスク上の履歴ファイルを、バックアップ・イメージの履歴ファイルで置換することを指定します。

REPLACE EXISTING
ターゲット・データベースの別名と同じ別名を持つデータベースがすでに存在している場合、このパラメーターは、リストア・ユーティリティーが既存のデータベースをリストアしたデータベースに置換するように指定します。これはリストア・ユーティリティーを起動するスクリプトで便利です。コマンド行プロセッサーは、ユーザーに既存のデータベースの削除を検証するよう求めるプロンプトを出さないためです。 WITHOUT PROMPTING パラメーターが指定された場合、REPLACE EXISTING を指定する必要はありませんが、その場合、ユーザー介入を標準的に必要とするイベントが起こるとこの操作は失敗します。

REDIRECT
リダイレクトしたリストア操作を指定します。リダイレクトしたリストア操作を完了するには、このコマンドの後に 1 つまたは複数の SET TABLESPACE CONTAINERS コマンドを続け、次に CONTINUE オプションを指定して RESTORE DATABASE コマンドを続ける必要があります。

注: 同一のリダイレクトしたリストア操作に関連したコマンドはすべて、同じウィンドウまたは CLP セッションから起動しなければなりません。

WITHOUT ROLLING FORWARD
データベースを、正常にリストアされた後ロールフォワード・ペンディング状態にしないように指定します。

正常なリストアに続いて、データベースがロールフォワード・ペンディング状態にある場合には、データベースが使用できるようになる前に、ROLLFORWARD コマンドを起動する必要があります。

オンライン・バックアップ・イメージからのリストアでこのオプションを指定した場合、エラー SQL2537N が戻されます。

WITHOUT DATALINK

DATALINK 列を持つ任意の表を DataLink_Reconcile_Pending (DRP) 状態にし、リンクされたファイルの調整を実行しないように指定します。

PARALLELISM *n*

リストア操作中に作成されるバッファー・マニピュレーターの数を指定します。値を明示的に指定しない場合、 DB2 はこのパラメーターのための最適値を自動的に選択します。

COMPRLIB *name*

解凍を実行するために使用するライブラリーの名前。この名前は、サーバー上の 1 個のファイルを参照する完全修飾パスでなければなりません。このパラメーターを指定しない場合、 DB2 はイメージ内に格納されているライブラリーの使用を試みます。バックアップが圧縮されていなかった場合、このパラメーターの値は無視されます。指定されたライブラリーをロードできない場合、リストアは失敗します。

COMPROPTS *string*

バイナリー・データのうち、解凍ライブラリーの初期設定ルーチンに渡すブロックを記述します。 DB2 はこのストリングをクライアントからサーバーに直接渡すため、バイト反転やコード・ページ変換の問題がある場合は解凍ライブラリーで処理する必要があります。データ・ブロックの最初の文字が '@' なら、データの残りの部分は、サーバー上に存在するファイルの名前を指定するものとして解釈されます。その場合 DB2 は、 *string* の内容をそのファイルの内容で置き換え、そのようにして得られる新しい値を初期設定ルーチンに渡します。 string の最大長は 1024 バイトです。

WITHOUT PROMPTING

リストア操作を無人で実行するように指定します。通常はユーザー介入を必要とするアクションでは、エラー・メッセージが戻されます。テープやディスケットなどの取り外し可能メディア装置を使用している場合、このオプションを指定していても、その装置が終わるとプロンプトが出されます。

例:

1. 以下の例で、データベース WSDB は 0 から 3 までの番号が付けられた 4 つのパーティションすべてに定義されています。パス /dev3/backup はすべてのパーティションからアクセスできます。以下のオフライン・バックアップ・イメージは、/dev3/backup から入手可能です。

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

最初にカタログ・パーティションをリストアしてから WSDB データベースの他のすべてのデータベース・パーティションを /dev3/backup ディレクトリーからリストアするには、データベース・パーティションの 1 つから以下のコマンドを出します。

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

db2_all ユーティリティーは、指定された各データベース・パーティションにリストア・コマンドを出します。

2. 以下は、別名が MYDB であるデータベースの典型的なリダイレクト・リストアのシナリオです。
   a. 次のように、REDIRECT オプションを指定して RESTORE DATABASE コマンドを発行する。

      ```
      db2 restore db mydb replace existing redirect
      ```

      ステップ 1 が正常終了した後でステップ 3 が完了する前に、次を発行してリストア操作を打ち切ることができる。

      ```
      db2 restore db mydb abort
      ```

   b. 再定義する必要があるコンテナーを持つ表スペースごとに、 SET TABLESPACE CONTAINERS コマンドを発行する。たとえば、

      ```
      db2 set tablespace containers for 5 using
         (file 'f:¥ts3con1' 20000, file 'f:¥ts3con2' 20000)
      ```

      リストアしたデータベースのコンテナーが、このステップで指定したものであることを検査するために、 LIST TABLESPACE CONTAINERS コマンドを発行する。

   c. ステップ 1 および 2 が正常終了した後、次を発行する。

      ```
      db2 restore db mydb continue
      ```

      これはリダイレクト・リストア操作の最終ステップです。

   d. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアはステップ 1 から再始動できる。
3. 以下は、リカバリー可能データベース用の増分バックアップの週間予定のサンプルです。週 1 回の全データベース・バックアップ操作、1 日 1 回の非累積 (差分) バックアップ操作、および週 2 回の累積 (増分) バックアップ操作が含まれています。

   ```
   (Sun) backup db mydb use tsm
   (Mon) backup db mydb online incremental delta use tsm
   (Tue) backup db mydb online incremental delta use tsm
   (Wed) backup db mydb online incremental use tsm
   ```

```
(Thu) backup db mydb online incremental delta use tsm
(Fri) backup db mydb online incremental delta use tsm
(Sat) backup db mydb online incremental use tsm
```

金曜日の午前中に作成されたイメージを自動データベース・リストアするには、次のようにします。

```
restore db mydb incremental automatic taken at (Fri)
```

金曜日の午前中に作成されたイメージを手動データベース・リストアするには、次のようにします。

```
restore db mydb incremental taken at (Fri)
restore db mydb incremental taken at (Sun)
restore db mydb incremental taken at (Wed)
restore db mydb incremental taken at (Thu)
restore db mydb incremental taken at (Fri)
```

4. リモート・サイトに移動することを意図したバックアップ・イメージを作成し、それにログを含めるには、次のようにします。

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

   このバックアップ・イメージをリストアするには、 LOGTARGET パスを指定し、ROLLFORWARD でそのパスを指定します。

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

5. ログを含むバックアップ・イメージから、ログ・ファイルだけを取り出すには、次のようにします。

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

6. リストア操作で使用する TSM 情報を指定するには、 USE TSM OPTIONS キーワードを使用します。 Windows プラットフォームでは、-fromowner オプションを指定しないでください。

   - 区切り文字付きストリングを指定する場合:

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

   - 完全修飾ファイル名を指定する場合:

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

     ファイル `myoptions.txt` には、 -fromnode=bar -fromowner=dmcinnis というストリングが含まれています。

使用上の注意:

- `db2 restore db <name>` という形式の RESTORE DATABASE コマンドは、表スペース・イメージ内に検出される表スペースの表スペース・リストアを実行します。 `db2 restore db <name> tablespace` という形式のすべての RESTORE DATABASE コマンドは、イメージ内で検出される表スペースの表スペース・リストアを実行します。表スペースのリストが提供されるすべての RESTORE DATABASE コマンドは、明示的にリストされているすべての表スペースのリストアを実行します。
- オンライン・バックアップのリストアを実行したなら、その後にロールフォワード・リカバリーを実行してください。
- バックアップ・イメージが圧縮されているなら、そのことは DB2 によって検出され、リストア前に自動的にデータが解凍されます。 db2Restore API でライブラ

リーが指定されている場合、データの解凍にはそれが使用されます。そうでない場合、バックアップ・イメージにライブラリーが格納されているなら、それが使用されます。そうでない場合、データは解凍できず、リストアは失敗します。

- バックアップ・イメージから圧縮ライブラリーをリストアする場合 (リストアのタイプとして DB2RESTORE_COMPR_LIB を指定して明示的に、または圧縮バックアップの通常のリストアを実行することにより暗黙的に)、そのリストア操作は、バックアップが作成されたのと同じプラットフォームおよびオペレーティング・システム上で実行する必要があります。バックアップ作成時のプラットフォームとリストア操作実行時のプラットフォームが違っていると、それらの 2 つのシステムの間のクロスプラットフォーム・リストアが DB2 で通常にサポートされている場合でも、リストア操作は失敗します。
- ログ・ファイルを含むバックアップ・イメージからログ・ファイルをリストアする場合は、 LOGTARGET オプションを指定し、それに DB2 サーバー上に存在する有効な完全修飾パス名を指定する必要があります。それらの条件が満たされている場合、リストア・ユーティリティーは、イメージ内のログ・ファイルをターゲット・パスに書き込みます。ログを含まないバックアップ・イメージのリストアで LOGTARGET を指定した場合、リストア操作で表スペース・データのリストアが試行される前にエラーが戻されます。また、LOGTARGET に無効なパスや読み取り専用パスが指定された場合も、リストアからエラーが戻されます。
- RESTORE DATABASE コマンド発行時点に LOGTARGET パス内にログ・ファイルが存在している場合、ユーザーに対して警告プロンプトが戻されます。WITHOUT PROMPTING が指定されている場合、この警告は戻されません。
- LOGTARGET を指定したリストア操作において、なんらかの理由で抽出できないログ・ファイルがあった場合、リストアは失敗し、エラーが戻されます。バックアップ・イメージから抽出するログ・ファイルの中に、 LOGTARGET パス内に既存のファイルと同じ名前のものが 1 つでもあると、リストア操作は失敗し、エラーが戻されます。データベース・リストア・ユーティリティーは、LOGTARGET ディレクトリー内に既存のログ・ファイルを上書きしません。
- 保管されているログ・セットだけをバックアップ・イメージからリストアすることも可能です。ログ・ファイルだけをリストアすることを指定するには、LOGTARGET パスに加えて LOGS オプションを指定します。 LOGTARGET パスを指定しないで LOGS オプションを指定すると、エラーになります。この操作モードでログ・ファイルをリストアしようとして問題が発生した場合、そのリストア操作は即座に終了し、エラーが戻されます。
- 自動増分リストア操作においては、リストア操作のターゲット・イメージに含まれているログだけがバックアップ・イメージから取り出されます。増分リストア処理中に参照される中間イメージに含まれるログが、それらの中間バックアップ・イメージから抽出されることはありません。手動増分リストア操作の場合、LOGTARGET パスは、最終リストア・コマンドを発行する場合にのみ指定してください。

関連資料:

- 77 ページの『BACKUP DATABASE』
- 143 ページの『ROLLFORWARD DATABASE』
- 「コマンド・リファレンス」の『db2move - データベース移動ツール・コマンド』

db2Restore - データベースのリストア

db2Backup (データベースのバックアップ) を使用して、バックアップが取られていたデータベースが損傷または破壊された場合に、そのデータベースを再作成します。リストアされたデータベースは、バックアップ・コピーの作成時と同じ状態になります。このユーティリティーでは、(新しいデータベースへのリストアが可能であることに加えて) バックアップ・イメージ内のデータベース名とは異なる名前でデータベースをリストアすることが可能です。

このユーティリティーは、以前の 2 つのリリースで作成した DB2 データベースをリストアするためにも使用できます。

このユーティリティーは、表スペース・レベルのバックアップからリストアすることもできますし、データベース・バックアップ・イメージ内から表スペースをリストアすることもできます。

有効範囲:

この API は、それが呼び出されたデータベース・パーティションにのみ影響を与えます。

権限:

既存のデータベースにリストアするには、以下のいずれかが必要です。

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

新規データベースにリストアするには、以下のいずれかが必要です。

- *sysadm*
- *sysctrl*

必要な接続:

データベース (既存のデータベースにリストアする場合)。この API を呼び出せば、指定したデータベースへの接続が自動的に確立され、リストア操作が終了すると接続が解放されます。

インスタンスおよびデータベース (新規データベースにリストアする場合)。データベースを作成するには、インスタンス・アタッチが必要です。

現行のインスタンス (DB2INSTANCE 環境変数の値で定義) とは異なるインスタンスで新規のデータベースへのリストアを行うには、まず、新規のデータベースを存在させるインスタンスにアタッチすることが必要です。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

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

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

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

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

汎用 API 構文:

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

typedef SQL_STRUCTURE db2gRestoreStruct
{
  char                         *piSourceDBAlias;
  db2Uint32                    iSourceDBAliasLen;
  char                         *piTargetDBAlias;
  db2Uint32                    iTargetDBAliasLen;
```

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

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

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

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

API パラメーター:

versionNumber

入力。 2 番目のパラメーター *pDB2RestoreStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2RestoreStruct

入力。 *db2RestoreStruct* 構造を指すポインター。

pSqlca

出力。*sqlca* 構造へのポインター。

piSourceDBAlias
入力。ソース・データベース・バックアップ・イメージのデータベース別名を含むストリング。

iSourceDBAliasLen
入力。ソース・データベースの別名の長さを示す 4 バイトの符号なし整数 (バイト単位)。

piTargetDBAlias
入力。宛先データベースの別名を含むストリング。このパラメーターが NULL の場合、*piSourceDBAlias* が使用されます。

iTargetDBAliasLen
入力。宛先データベースの別名の長さを示す 4 バイトの符号なし整数 (バイト単位)。

oApplicationId
出力。 API によって、アプリケーションにサービスを提供しているエージェントを識別するストリングが戻されます。データベース・モニターを使用するバックアップ操作の進行状況に関する情報を取得することもできます。

poApplicationId
出力。長さ SQLU_APPLID_LEN+1 (`sqlutil` で定義) のバッファーを提供します。 API によって、アプリケーションにサービスを提供しているエージェントを識別するストリングが戻されます。データベース・モニターを使用するバックアップ操作の進行状況に関する情報を取得することもできます。

iApplicationIdLen
入力。 poApplicationId バッファーの長さを示す 4 バイトの符号なし整数 (バイト単位) です。 SQLU_APPLID_LEN+1 (`sqlutil` に定義) と等しくなければなりません。

piTimestamp
入力。バックアップ・イメージのタイム・スタンプを示すストリング。指定したソース内にバックアップ・イメージが 1 つしかない場合、このフィールドはオプションです。

iTimestampLen
入力。 piTimestamp バッファーの長さを示す 4 バイトの符号なし整数 (バイト単位)。

piTargetDBPath
入力。サーバー上の宛先データベースのディレクトリーの相対または完全修飾名を含むストリング。リストアされるバックアップのために新規のデータベースを作成する場合に使用します。そうでない場合は使用しません。

piReportFile
入力。ファイル名が指定されている場合、完全修飾名にしなければなりません。リストア時にリンク解除されるデータ・リンク・ファイル (高速調整の結果) が報告されます。

iReportFileLen
入力。 piReportFile バッファーの長さを示す 4 バイトの符号なし整数 (バイト単位) です。

piTablespaceList

入力。リストアされる表スペースのリストです。データベースまたは表スペース・バックアップ・イメージから、表スペースのサブセットをリストアする場合に使用されます。詳細については、*DB2TablespaceStruct* 構造を参照してください。以下の制限が適用されます。

- データベースはリカバリー可能でなければなりません。つまり、ログ保存またはユーザー出口が使用可能になっていなければなりません。
- リストアされるデータベースは、バックアップ・イメージの作成に使用されたのと同じデータベースでなければなりません。つまり、表スペース・リストア機能を使用して、データベースに表スペースを追加することはできません。
- ロールフォワード・ユーティリティーは、パーティション・データベース環境でリストアされる表スペースが、同じ表スペースを含む他のデータベース・パーティションと必ず同期化されるようにします。表スペースのリストア操作が要求され、*piTablespaceList* が NULL の場合、リストア・ユーティリティーはバックアップ・イメージ内のすべての表スペースのリストアを試みます。

バックアップ後に名前が変更された表スペースをリストアする場合、リストア・コマンドでは新しい表スペース名を使用しなければなりません。古い表スペース名を使用すると、その表スペースを見つけることができません。

piMediaList

入力。バックアップ・イメージのソース・メディア。提供される情報は、locationType フィールドの値によって異なります。 locationType に有効な値 (`sqlutil` で定義) は、以下のとおりです。

SQLU_LOCAL_MEDIA

ローカル装置 (テープ、ディスクまたはディスケットの組み合わせ)。

SQLU_XBSA_MEDIA

XBSA インターフェース。バックアップ・サービス API (XBSA) は、バックアップやアーカイブの目的でデータ・ストレージ管理を必要とするアプリケーションまたは機能のための、オープン・アプリケーション・プログラミング・インターフェースです。

SQLU_TSM_MEDIA

TSM。ロケーション・ポインターが NULL に設定されている場合、 DB2 で提供される TSM 共用ライブラリーが使用されます。別のバージョンの TSM 共用ライブラリーが必要な場合には、SQLU_OTHER_MEDIA を使用し、共用ライブラリー名を入力してください。

SQLU_OTHER_MEDIA

ベンダー製品。ロケーション・フィールド内の共用ライブラリー名を提供します。

SQLU_USER_EXIT

ユーザー出口。追加の入力は必要ありません (サーバーが OS/2 上にある場合のみ使用可能です)。

piUsername
入力。接続の試行時に使用されるユーザー名を含むストリングを指定します。 NULL にすることもできます。

iUsernameLen
入力。 piUsername の長さを示す 4 バイトの符号なし整数 (バイト単位)。ユーザー名が提供されていない場合は、ゼロに設定してください。

piPassword
入力。ユーザー名とともに使用されるパスワードを含むストリングを指定します。 NULL にすることもできます。

iPasswordLen
入力。 piPassword の長さを示す 4 バイトの符号なし整数 (バイト単位)。パスワードが提供されていない場合は、ゼロに設定してください。

piNewLogPath
入力。リストア完了後、ロギングに使用されるパスを表すストリング。このフィールドが NULL の場合、デフォルトのログ・パスが使用されます。

iNewLogPathLen
入力。 *piNewLogPath* の長さを示す 4 バイトの符号なし整数 (バイト単位)。

piVendorOptions
入力。情報をアプリケーションからベンダー関数へ渡すのに使用されます。このデータ構造はフラットでなければなりません。つまり、間接のレベルはサポートされません。このデータについては、バイト反転が行われず、また、コード・ページがチェックされないことに注意してください。

iVendorOptionsSize
入力。 *piVendorOptions* の長さです。 65535 バイト以下でなければなりません。

iParallelism
入力。並列処理の度合い (バッファー・マニピュレーターの数) を指定します。最小値は 1 です。最大値は 1024 です。

iBufferSize
入力。バッファー・サイズを 4 KB の割り振り単位 (ページ) でバックアップします。最小値は 8 単位です。リストア用に入力するサイズは、バックアップ・イメージを作成するのに使用するバッファー・サイズと同じか、または整数倍でなければなりません。

iNumBuffers
入力。使用するリストア・バッファーの数を指定します。

iCallerAction
入力。実行するアクションを指定します。有効な値 (db2ApiDf で定義) は、以下のとおりです。

DB2RESTORE_RESTORE
リストア操作を開始します。

DB2RESTORE_NOINTERRUPT
リストアを開始します。リストアを自動実行するよう指定します。

通常ユーザーの介入を要求するシナリオは、呼び出し側への最初の戻りなしに試行されるか、エラーを生成します。この呼び出し側アクションは、たとえば、リストアに必要なメディアがすべてマウントされていることが明らかで、ユーティリティーによるプロンプトが必要とされない場合に使用してください。

DB2RESTORE_CONTINUE
ユーザーがユーティリティーによって要求された何らかのアクション (たとえば、新しいテープのマウント) を実行した後で、リストアを継続します。

DB2RESTORE_TERMINATE
ユーザーがユーティリティーによって要求された何らかのアクションの実行に失敗した場合、リストアを終了します。

DB2RESTORE_DEVICE_TERMINATE
リストアによって使用される装置のリストから特定の装置を除外します。特定の装置が入力でいっぱいになると、リストア・ユーティリティーは呼び出し側に警告を戻します。その場合、この呼び出し側アクションを指定してリストアを再び呼び出すことによって、警告が生成される原因となった装置を使用装置のリストから除外してください。

DB2RESTORE_PARM_CHK
リストアを実行することなく、パラメーターの妥当性を検査します。このオプションは、呼び出しが戻った後でデータベース接続を終了しません。この呼び出しが正常に戻った後、ユーザーが DB2RESTORE_CONTINUE で呼び出しを発行し、処置を進めることが期待されます。

DB2RESTORE_PARM_CHK_ONLY
リストアを実行することなく、パラメーターの妥当性を検査します。この呼び出しが戻る前に、この呼び出しによって確立したデータベース接続は終了し、後続する呼び出しは必要なくなります。

DB2RESTORE_TERMINATE_INCRE
完了前に増分リストア操作を終了します。

DB2RESTORE_RESTORE_STORDEF
最初の呼び出し。表スペース・コンテナーの再定義が要求されます。

DB2RESTORE_STORDEF_NOINTERRUPT
最初の呼び出し。リストアは割り込まれずに実行されます。表スペース・コンテナーの再定義が要求されます。

iOptions
入力。リストア・プロパティーのビットマップ。オプションは組み合わされて、ビット単位 OR 演算子を使用して *iOptions* の値を生成します。有効な値 (db2ApiDf で定義) は、以下のとおりです。

DB2RESTORE_OFFLINE
オフライン・リストア操作を実行します。

DB2RESTORE_ONLINE
オンライン・リストア操作を実行します。

DB2RESTORE_DB
データベースにあるすべての表スペースをリストアします。これはオフラインで実行する必要があります。

DB2RESTORE_TABLESPACE
バックアップ・イメージから、*piTablespaceList* パラメーターにリストされた表スペースのみリストアします。これはオンラインまたはオフラインで実行できます。

DB2RESTORE_HISTORY
履歴ファイルだけをリストアします。

DB2RESTORE_COMPR_LIB
圧縮ライブラリーをリストアすることを指定します。このオプションは、他のリストア・タイプと同時に使用することはできません。バックアップ・イメージの中にオブジェクトが存在している場合、それはデータベース・ディレクトリーの中にリストアされます。バックアップ・イメージの中にオブジェクトが存在していない場合、リストア操作は失敗します。

DB2RESTORE_LOGS
バックアップ・イメージに含まれている一連のログ・ファイルだけをリストアすることを指定します。バックアップ・イメージの中にログ・ファイルが含まれていない場合、リストア操作は失敗します。このオプションを指定する場合は、*piLogTarget* パラメーターも指定する必要があります。

DB2RESTORE_INCREMENTAL
手動累積リストア操作を実行します。

DB2RESTORE_AUTOMATIC
自動累積 (増分) リストア操作を実行します。
DB2RESTORE_INCREMENTAL とともに指定しなければなりません。

DB2RESTORE_DATALINK
調整操作を実行します。定義された DATALINK 列を含む表では、RECOVERY YES オプションを指定する必要があります。

DB2RESTORE_NODATALINK
調整操作を実行しません。 DATALINK 列を持つ表は、DataLink_Roconcile_pending (DRP) 状態に入れられます。定義された DATALINK 列を含む表では、RECOVERY YES オプションを指定する必要があります。

DB2RESTORE_ROLLFWD
データベースのリストアが成功した後、データベースをロールフォワード・ペンディング状態にします。

DB2RESTORE_NOROLLFWD
データベースのリストアが成功した後、データベースをロールフォワード・ペンディング状態にしません。バックアップがオンライン

で実行される場合、または表スペース・レベルのリストアの場合は、この値は指定できません。リストアが成功した後で、データベースがロールフォワード・ペンディング状態にある場合は、db2Rollforward (データベースのロールフォワード) を実行してからでなければ、データベースを使用できません。

piComprLibrary
入力。バックアップ・イメージが圧縮されている場合、バックアップ・イメージの解凍を実行するために使用する外部ライブラリーの名前を示します。この名前は、サーバー上の 1 個のファイルを参照する完全修飾パスでなければなりません。値が NULL ポインターであるか、空ストリングへのポインターである場合、 DB2 は、イメージに保管されたライブラリーを使用しようとします。バックアップが圧縮されていなかった場合、このパラメーターの値は無視されます。指定されたライブラリーが見付からない場合、リストアは失敗します。

piComprLibraryLen
入力。 piComprLibrary で指定したライブラリー名の長さを示す 4 バイトの符号なし整数 (バイト単位) です。ライブラリー名が提供されていない場合は、ゼロに設定してください。

piComprOptions
入力。バイナリー・データのうち、解凍ライブラリーの初期設定ルーチンに渡すブロックを記述します。 DB2 はこのストリングをクライアントからサーバーに直接渡すため、バイト反転やコード・ページ変換の問題がある場合は圧縮ライブラリーで処理する必要があります。データ・ブロックの最初の文字が '@' なら、データの残りの部分は、サーバー上に存在するファイルの名前を指定するものとして解釈されます。その場合 DB2 は、piComprOptions および iComprOptionsSize の内容をそのファイルの内容およびサイズで置き換え、そのようにして得られる新しい値を初期設定ルーチンに渡します。

iComprOptionsSize
入力。 piComprOptions として渡されるデータ・ブロックのサイズを表す 4 バイトの符号なし整数。 piComprOptions が NULL ポインターである場合に限り、iComprOptionsSize はゼロになります。

piLogTarget
入力。バックアップ・イメージからログ・ファイルを抽出する際のターゲット・ディレクトリーとして使用する、データベース・サーバー上の既存のディレクトリーの絶対パス。このパラメーターを指定する場合、バックアップ・イメージ内のログ・ファイルは、そのターゲット・ディレクトリー内に抽出されます。このパラメーターを指定しない場合、バックアップ・イメージ内のログ・ファイルは抽出されません。バックアップ・イメージからログ・ファイルだけを抽出する場合は、DB2RESTORE_LOGS パラメーターを使用してください。

iLogTargetLen
入力。 *piLogTarget* 内のパスの長さを示す 4 バイトの符号なし整数 (バイト単位) です。

tablespaces
バックアップを取る表スペースのリストを指すポインター。 C の場合、リストは NULL で終了するストリングです。一般的には、*db2Char* 構造のリストです。

numTablespaces
tablespaces パラメーター内の項目数。

locations
メディア・ロケーションのリストを指すポインター。 C の場合、リストは NULL で終了するストリングです。一般的には、db2Char 構造のリストです。

numLocations
locations パラメーター内の項目数。

locationType
メディア・タイプを示す文字。有効な値 (`sqlutil` で定義) は、以下のとおりです。

SQLU_LOCAL_MEDIA
ローカル装置 (テープ、ディスク、ディスケット、または名前付きパイプ)

SQLU_XBSA_MEDIA
XBSA インターフェース。

SQLU_TSM_MEDIA
Tivoli Storage Manager。

SQLU_OTHER_MEDIA
ベンダー・ライブラリー。

SQLU_USER_EXIT
ユーザー出口 (サーバーが OS/2 上にある場合のみ使用可能です)。

pioData
文字データ・バッファーを指すポインター。

iLength
入力。 *pioData* バッファーのサイズ

oLength
出力。将来の利用のために予約されています。

使用上の注意:

オフラインのリストアの場合、このユーティリティーは、排他モードでデータベースに接続します。リストアされるデータベースにアプリケーション (呼び出し側のアプリケーションを含む) がすでに接続している場合、このユーティリティーは失敗します。さらに、リストア・ユーティリティーが、リストアの実行に使用されており、アプリケーション (呼び出し側のアプリケーションを含む) が同じワークステーション上の任意のデータベースにすでに接続されている場合、要求は失敗します。接続が成功すると、 API はリストアが完了するまで他のアプリケーションを締め出します。

現行のデータベース構成ファイルは、それが使用不能でない限り、バックアップ・コピーによって置換されません。ファイルが置換される場合には、警告メッセージが戻されます。

db2Backup (データベースのバックアップ) を使用して、データベースまたは表スペースをバックアップしなければなりません。

呼び出し側アクションが DB2RESTORE_NOINTERRUPT の場合、リストアはアプリケーションにプロンプトを出すことなく継続されます。呼び出し側のアクションが DB2RESTORE_RESTORE で、ユーティリティーが既存のデータベースにリストアしようとしている場合、ユーティリティーは、何らかのユーザー介入を要求するメッセージとともに、アプリケーションに制御を戻します。ユーザーとの対話の処理が終了した後、後続の呼び出しにおいて処理が継続される (DB2RESTORE_CONTINUE) か、終了 (DB2RESTORE_TERMINATE) されるかを示す呼び出し側アクションを設定して、RESTORE DATABASE を再び呼び出します。このユーティリティーは処理を終了させ、 *sqlca* で SQLCODE を戻します。

終了時に装置をクローズするには、呼び出し側アクションを DB2RESTORE_DEVICE_TERMINATE に設定してください。たとえば、2 つのテープ装置を使用して 3 つのテープ・ボリュームからリストアを行う場合、テープの 1 つがリストアされると、アプリケーションは、API からテープの終わりを示す SQLCODE とともに制御を受け取ります。ここで、アプリケーションはユーザーに別のテープをマウントするよう要求しますが、テープが「もうない」ことをユーザーが示すと、メディア装置の終了を通知する呼び出し側アクション SQLUD_DEVICE_TERMINATE を指定して API に戻ります。これで、デバイス・ドライバーは終了しますが、リストアに関連する残りの装置は、リストア・セット内の全セグメントがリストアされるまで処理済みの入力を保持し続けます (バックアップ処理中に、リストア・セット内のセグメントの数が最後のメディア装置に置かれます)。この呼び出し側アクションは、テープ以外の装置 (ベンダーによってサポートされる装置) でも使用できます。

アプリケーションに戻る前にパラメーター・チェックを実行したい場合には、呼び出し側アクションを SQLUD_PARM_CHECK に設定してください。

宛先変更したリストアを実行する場合には、呼び出し側アクションを DB2RESTORE_RESTORE_STORDEF に設定し、「sqlbstsc - 表スペース・コンテナーの設定」との組み合わせで使用してください。

データベース・リストアの重要な段階でシステム障害が発生した場合、正常なリストアが実行されるまで、ユーザーはデータベースに正常に接続することができません。接続を試みたときにエラー・メッセージが戻されることによって、この状態であることが検出されます。バックアップされたデータベースがロールフォワード・リカバリーに使用できるよう構成されておらず、さらにログ保存パラメーターとユーザー出口パラメーターのいずれかが使用可能になっている、使用可能な現行の構成ファイルがある場合、ユーザーは、リストアに続いて、データベースの新しいバックアップをとるか、またはこれらのパラメーターを使用不能にしてから、データベースに接続することが必要です。

リストアが失敗すると、リストアされたデータベースはドロップされませんが (既存のデータベース以外へのリストアの場合を除き)、使用不能になります。

バックアップ上の履歴ファイルをリストアするようにリストア・タイプで指定されている場合、それはデータベースの既存の履歴ファイル上にリストアされます。事実上、リストアされるバックアップの後に履歴ファイルに加えられた変更は、すべて消去されることになります。このことが望ましくない場合は、履歴ファイルを新規またはテスト・データベースにリストアさせることにより、実行された更新を破棄することなく、履歴ファイルの内容を表示できるようにしてください。

バックアップ操作時にデータベースのロールフォワード・リカバリーが使用可能だった場合には、 db2Restore の実行が成功した後に db2Rollforward を発行することによって、データベースを損傷または破壊の発生前の状態に戻すことができます。データベースがリカバリー可能な場合、リストアの完了後に、デフォルトでペンディング状態がロールフォワードされます。

データベース・バックアップ・イメージがオフラインで作成されており、呼び出し側がリストア後のデータベースの ロールフォワードを必要としない場合、リストア用に DB2RESTORE_NOROLLFWD オプションを使用できます。これにより、リストア後にデータベースがすぐに使用可能になります。バックアップ・イメージがオンラインで作成されている場合は、呼び出し元はリストアが完了した時点で、対応するログ・レコードを使用してロールフォワードを行わなければなりません。

ログ・ファイルを含むバックアップ・イメージからログ・ファイルをリストアする場合は、 LOGTARGET オプションを指定し、それに DB2 サーバー上に存在する有効な完全修飾パス名を指定する必要があります。それらの条件が満たされている場合、リストア・ユーティリティーは、イメージ内のログ・ファイルをターゲット・パスに書き込みます。ログを含まないバックアップ・イメージのリストアで LOGTARGET を指定した場合、リストア操作で表スペース・データのリストアが試行される前にエラーが戻されます。また、LOGTARGET に無効なパスや読み取り専用パスが指定された場合も、リストアからエラーが戻されます。

リストア・コマンド発行時点に LOGTARGET パス内にログ・ファイルが存在している場合、ユーザーに対して警告プロンプトが戻されます。 WITHOUT PROMPTING が指定されている場合、この警告は戻されません。

LOGTARGET を指定したリストア時に、なんらかの理由で抽出できないログ・ファイルがあった場合、リストアは失敗し、エラーが戻されます。バックアップ・イメージから抽出するログ・ファイルの中に、 LOGTARGET パス内に既存のファイルと同じ名前のものが 1 つでもあると、リストア操作は失敗し、エラーが戻されます。リストア・ユーティリティーは、LOGTARGET ディレクトリー内に既存のログ・ファイルを上書きしません。

保管されているログ・セットだけをバックアップ・イメージからリストアすることも可能です。ログ・ファイルだけをリストアすることを指定するには、 LOGTARGET パスに加えて LOGS オプションを指定します。 LOGTARGET パスを指定しないで LOGS オプションを指定すると、エラーになります。この操作モードでログ・ファイルをリストアしようとして問題が発生した場合、そのリストアは即座に終了し、エラーが戻されます。

自動増分リストア時には、リストア操作のターゲット・イメージに含まれているログだけがバックアップ・イメージから取り出されます。増分リストア処理中に参照される中間イメージに含まれるログが、それらの中間バックアップ・イメージから

抽出されることはありません。手動増分リストア時には、LOGTARGET パスは、最終リストア・コマンドを発行する場合にのみ指定してください。

バックアップが圧縮されているなら、そのことは DB2 によって検出され、リストア前に自動的にデータが解凍されます。 db2Restore API でライブラリーが指定されている場合、データの解凍にはそれが使用されます。そうでない場合、バックアップ・イメージにライブラリーが格納されているなら、それが使用されます。そうでない場合、データは解凍できず、リストアは失敗します。

バックアップ・イメージから圧縮ライブラリーをリストアする場合 (リストアのタイプとして DB2RESTORE_COMPR_LIB を指定して明示的に、または圧縮バックアップの通常のリストアを実行することにより暗黙的に)、そのリストア操作は、バックアップが作成されたのと同じプラットフォームおよびオペレーティング・システム上で実行する必要があります。バックアップ作成時のプラットフォームとリストア操作実行時のプラットフォームが違っていると、それらの 2 つのシステムの間のクロスプラットフォーム・リストアが DB2 で通常にサポートされている場合でも、リストア操作は失敗します。

関連資料:

- 「*管理 API リファレンス*」の『sqlemgdb - データベースの移行』
- 153 ページの『db2Rollforward - データベースのロールフォワード』
- 「*管理 API リファレンス*」の『SQLCA』
- 83 ページの『db2Backup - データベースのバックアップ』
- 「*管理 API リファレンス*」の『db2CfgGet - 構成パラメーターの入手』

関連サンプル:

- 『dbrecov.sqc -- How to recover a database (C)』
- 『dbrecov.sqC -- How to recover a database (C++)』

リストア・セッション - CLP の例

例 1

以下は、別名が MYDB であるデータベースの典型的な非増分リダイレクト・リストアのシナリオです。

1. 次のように、REDIRECT オプションを指定して RESTORE DATABASE コマンドを発行する。

   ```
   db2 restore db mydb replace existing redirect
   ```

2. 再定義したいコンテナーを持つ表スペースごとに、SET TABLESPACE CONTAINERS コマンドを発行する。たとえば、Windows 環境では次のようにします。

   ```
   db2 set tablespace containers for 5 using
          (file 'f:¥ts3con1'20000, file 'f:¥ts3con2'20000)
   ```

   リストアしたデータベースのコンテナーが、このステップで指定したものであることを検査するために、コンテナーの場所が再定義されているすべての表スペースのために LIST TABLESPACE CONTAINERS コマンドを発行する。

3. ステップ 1 および 2 が正常終了した後、次を発行する。

```
db2 restore db mydb continue
```

これはリダイレクト・リストア操作の最終ステップです。

4. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアはステップ 1 から再始動できる。

注:

1. ステップ 1 が正常終了した後でステップ 3 が完了する前に、次を発行してリストア操作を打ち切ることができる。

```
db2 restore db mydb abort
```

2. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアはステップ 1 から再始動できる。

例 2

以下は、別名が MYDB であり、以下のバックアップ・イメージを持つデータベースの、典型的な手動増分リダイレクト・リストアのシナリオです。

```
backup db mydb
Backup successful. The timestamp for this backup image is : <ts1>

backup db mydb incremental
Backup successful. The timestamp for this backup image is : <ts2>
```

1. 次のように、INCREMENTAL および REDIRECT オプションを指定して RESTORE DATABASE コマンドを発行する。

```
db2 restore db mydb incremental taken at <ts2> replace existing redirect
```

2. 再定義する必要があるコンテナーを持つ表スペースごとに、 SET TABLESPACE CONTAINERS コマンドを発行する。たとえば、Windows 環境では次のようにします。

```
db2 set tablespace containers for 5 using
       (file 'f:¥ts3con1'20000, file 'f:¥ts3con2'20000)
```

リストアしたデータベースのコンテナーが、このステップで指定したものであることを検査するために、 LIST TABLESPACE CONTAINERS コマンドを発行する。

3. ステップ 1 および 2 が正常終了した後、次を発行する。

```
db2 restore db mydb continue
```

4. 残りの増分リストア・コマンドは、以下のように発行することができるようになります。

```
db2 restore db mydb incremental taken at <ts1>
db2 restore db mydb incremental taken at <ts2>
```

これはリダイレクト・リストア操作の最終ステップです。

注:

1. ステップ 1 が正常終了した後でステップ 3 が完了する前に、次を発行してリストア操作を打ち切ることができる。

```
db2 restore db mydb abort
```

2. ステップ 3 が正常終了した後でステップ 4 のすべての必要なコマンドを実行する前に、次を発行してリストア操作を打ち切ることができる。

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

3. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアはステップ 1 から再始動できる。
4. いずれかのリストア・コマンドがステップ 4 で失敗した場合、失敗したコマンドを再実行してリストア・プロセスを継続できます。

例 3

以下は、同じデータベースでの典型的な自動増分リダイレクト・リストア・シナリオです。

1. 次のように、INCREMENTAL AUTOMATIC および REDIRECT オプションを指定して RESTORE DATABASE コマンドを発行する。

```
db2 restore db mydb incremental automatic taken at <ts2>
       replace existing redirect
```

2. 再定義する必要があるコンテナーを持つ表スペースごとに、 SET TABLESPACE CONTAINERS コマンドを発行する。たとえば、Windows 環境では次のようにします。

```
db2 set tablespace containers for 5 using
       (file 'f:¥ts3con1'20000, file 'f:¥ts3con2'20000)
```

   リストアしたデータベースのコンテナーが、このステップで指定したものであることを検査するために、 LIST TABLESPACE CONTAINERS コマンドを発行する。

3. ステップ 1 および 2 が正常終了した後、次を発行する。

```
db2 restore db mydb continue
```

   これはリダイレクト・リストア操作の最終ステップです。

注:

1. ステップ 1 が正常終了した後でステップ 3 が完了する前に、次を発行してリストア操作を打ち切ることができる。

```
db2 restore db mydb abort
```

2. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアは以下を実行した後ステップ 1 から再始動できる。

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

関連資料:

- 102 ページの『RESTORE DATABASE』
- 「コマンド・リファレンス」の『LIST TABLESPACE CONTAINERS コマンド』
- 「コマンド・リファレンス」の『SET TABLESPACE CONTAINERS コマンド』

リストアのパフォーマンスの最適化

リストア操作を実行すると、DB2 は、バッファー数、バッファー・サイズ、および並列処理設定の最適値を自動的に選択します。この値は、使用可能なユーティリティー・ヒープ・メモリーの大きさ、使用可能なプロセッサー数、およびデータベース構成に基づきます。目的は、リストア操作の完了にかかる時間を最小限に抑えることです。次の RESTORE DATABASE コマンド・パラメーターの値を明示的に入力しないと、DB2 側が値を選択します。

- WITH num-buffers BUFFERS
- PARALLELISM n
- BUFFER buffer-size

リストア操作の場合、バックアップ操作によって使用されるバッファー・サイズの倍数が常に使用されます。データベース・マネージャー構成パラメーター BACKBUFSZ および RESTBUFSZ によって指定される値は無視されます。これらの値を使用するのであれば、RESTORE DATABASE コマンドを発行するときに、バッファー・サイズを明示的に指定する必要があります。

さらに、リストア操作を完了するために必要な時間を短縮するために、以下のいずれかを実行することを選択できます。

- リストア・バッファー・サイズを大きくする。

  リストア・バッファー・サイズは、バックアップ操作中に指定したバックアップ・バッファー・サイズに正の整数を乗算したサイズでなければなりません。誤ったバッファー・サイズを指定すると、割り振られるバッファーは、許容可能な最小のサイズになります。
- バッファー数を増やします。

  指定する値は、バックアップ・バッファーに指定したページ数の倍数でなければなりません。ページ数の最小値は 8 です。
- PARALLELISM パラメーターの値を増やす。

  これにより、リストア操作中にデータベースへの書き込みのために使用されるバッファー・マニピュレーター (BM) の番号が増加します。

関連概念:

- 95 ページの『リストアの概要』

関連タスク:

- 96 ページの『リストアの使用』

第 4 章 ロールフォワード・リカバリー

ここでは、DB2 UDB ロールフォワード・リカバリー・ユーティリティーについて説明します。このユーティリティーは、データベース・リカバリー・ログ・ファイルに記録されているトランザクションを適用することによりデータベースをリカバリーします。

以下のトピックについて説明します。

ロールフォワードの概要

最も単純な形式の DB2® ROLLFORWARD DATABASE コマンドで必要なのは、ロールフォワード・リカバリーしたいデータベースの別名を指定することだけです。たとえば、次のようにします。

```
db2 rollforward db sample to end of logs and stop
```

この場合、コマンドの戻りは以下のようになります。

```
                                Rollforward Status

 Input database alias                   = sample
 Number of nodes have returned status   = 1

 Node number                            = 0
 Rollforward status                     = not pending
 Next log file to be read               =
 Log files processed                    =  -
 Last committed transaction             = 2001-03-11-02.39.48.000000

DB20000I  The ROLLFORWARD command completed successfully.
```

ロールフォワード・リカバリーの一般的なアプローチは、以下のとおりです。

1. STOP オプションは指定せずにロールフォワード・ユーティリティーを起動する。

2. QUERY STATUS オプションを指定してロールフォワード・ユーティリティーを起動する

   リカバリーをログの最後まで指定する場合、戻されたポイント・イン・タイムが予定より早いと、 QUERY STATUS オプションは 1 つ以上のログ・ファイルが欠落していると示すことがあります。

   ポイント・イン・タイム指定リカバリーを指定する場合、 QUERY STATUS オプションはロールフォワード操作が確実に正しいポイント・イン・タイムに完了するようにするのに役立ちます。
3. STOP オプションを指定してロールフォワード・ユーティリティーを起動する。操作の停止後は、追加変更項目をロールフォワードすることはできません。

データベースは (restore ユーティリティーを使用し) ロールフォワードする前に正常にリストアしておく必要があります。ただし、表スペースについてはその必要はありません。表スペースを一時的にロールフォワード・ペンディング状態にすることができます。ただし、その状態を取り消すためにリストア操作を行う必要はありません (たとえば、電源割り込みの後など)。

ロールフォワード・ユーティリティーが起動されると、次のようになります。

- データベースがロールフォワード・ペンディング状態の場合、データベースはロールフォワードされます。表スペースもロールフォワード・ペンディング状態にある場合、データベースのロールフォワード操作が完了した後に再びロールフォワード・ユーティリティーを起動して、表スペースをロールフォワードする必要があります。
- データベースはロールフォワード・ペンディング状態にない が、表スペースはロールフォワード・ペンディング状態にある 場合は以下のようになります。
  - 表スペースのリストを指定すると、それらの表スペースだけがロールフォワードされます。
  - 表スペースのリストを指定しないと、ロールフォワード・ペンディング状態の表スペースすべてがロールフォワードされます。

データベース・ロールフォワード操作は、オフラインで実行されます。ロールフォワード操作が正常に完了するまでは、データベースは使用不可です。また、ユーティリティーの起動時に STOP オプションを指定していないと操作は完了できません。

表スペース・ロールフォワード操作は、オフラインで実行できます。ロールフォワード操作が正常に完了するまでは、データベースは使用不可です。ログの最後まで行われた場合、またはユーティリティーの起動時に STOP オプションを指定していた場合、正常に完了します。

SYSCATSPACE が含まれていない限り、オンライン で表スペースのロールフォワード操作を実行することができます。表スペースに対してオンラインでロールフォワード操作を実行するときは、その表スペース自体は使用できませんが、データベース内の他の表スペースは使用できます。

データベースを初めて作成した時点では、循環ロギングだけが使用可能になります。つまり、ログは保管やアーカイブされるのではなく再使用されます。循環ロギ

ングでは、ロールフォワード・リカバリーは不可能です。クラッシュ・リカバリーまたはバージョン・リカバリーだけが行えます。アーカイブ・ログは、バックアップを取った後に生じるデータベースへの変更を文書化します。ログ・アーカイブ (およびロールフォワード・リカバリー) を使用可能にするには、 *logarchmeth1* データベース構成パラメーターを、デフォルトの OFF 以外の値に設定します。*logarchmeth1* を OFF 以外の値に設定すると、データベースはバックアップ・ペンディング状態になり、データベースを再び使用するにはデータベースのオフライン・バックアップを作成する必要があります。

注: 項目は、ロールフォワード操作で使用されるログ・ファイルごとに、リカバリー履歴ファイルに作成されます。

関連概念:

- 140 ページの『ロード・コピー・ロケーション・ファイルを使用したデータのリカバリー』
- 36 ページの『リカバリー・ログについて』

関連資料:

- 143 ページの『ROLLFORWARD DATABASE』
- 40 ページの『データベース・ロギングの構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logarchmeth1 - 1 次ログ・アーカイブ方式構成パラメーター』

ロールフォワードの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャー保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスが制御されます。ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

ロールフォワード・ユーティリティーを使用するには、 SYSADM、SYSCTRL、または SYSMAINT 権限が必要です。

ロールフォワードの使用

前提条件:

ロールフォワード・リカバリーを実行するデータベースに接続していてはなりません。ロールフォワード・ユーティリティーは指定されたデータベースに自動的に接続を確立し、この接続はロールフォワード操作が完了すると終了します。

進行中のロールフォワード操作をキャンセルせずに表スペースをリストアすることはしないでください。そうしてしまうと、ある表スペースはロールフォワード進行状態にあり、別の表スペースはロールフォワード・ペンディング状態にある、とい

う状態になることがあります。進行中のロールフォワード操作は、ロールフォワード進行状態にある表スペース上でのみ作動します。

データベースには、ローカル・エージェントとリモート・エージェントがあります。

制限:

ロールフォワード・ユーティリティーには、以下の制限が適用されます。

- ロールフォワード操作は一度に 1 つしか起動することができません。リカバリーする表スペースが多数ある場合は、それらすべてを同一の操作に指定することができます。
- 最後に実行したバックアップの後で表スペースの名前を変更した場合、その表スペースをロールフォワードする際には必ず新しい名前を使用してください。以前の表スペースの名前は認識されません。
- 実行中のロールフォワード操作をキャンセルすることはできません。完了したロールフォワード操作のみキャンセルすることができます。ただし、STOP オプションが指定されていない操作、または完了前に失敗したロールフォワード操作はキャンセルできません。
- 直前のタイム・スタンプより前のタイム・スタンプを指定して、表スペースのロールフォワード操作を特定のポイント・イン・タイムまで*継続する* ことはできません。特定のポイント・イン・タイムが指定されない場合は、直前のポイント・イン・タイムが使用されます。 STOP を指定するだけで、特定のポイント・イン・タイムまでのロールフォワード操作を開始することができます。ただし、これは関係する表スペースがすべて同一のオフライン・バックアップ・イメージからリストアされた場合にのみ可能です。この場合、ログ処理は必要ありません。進行中のロールフォワード操作が完了する前またはキャンセルされる前に、別のロールフォワード操作を別の表スペース・リストで開始すると、エラー・メッセージ (SQL4908) が戻されます。 LIST TABLESPACES コマンドをすべてのノード上で起動し、現在ロールフォワード中 (ロールフォワード進行状態) の表スペース、およびロールフォワード作動可能 (ロールフォワード・ペンディング状態) の表スペースを判別してください。次の 3 つのオプションがあります。
  - すべての表スペースに対する進行中のロールフォワード操作を終了する。
  - 表スペースのサブセットに対するロールフォワード操作を終了する。 (ロールフォワード操作が特定のポイント・イン・タイムまで継続するようになっている場合、すべてのノードがかかわるため、このオプションは使用できないことがあります。)
  - 進行中のロールフォワード操作をキャンセルする。
- パーティション・データベース環境では、ロールフォワード・ユーティリティーはデータベースのカタログ・ノードから起動する必要があります。

手順:

ロールフォワード・ユーティリティーは、コマンド行プロセッサー (CLP)、コントロール・センターにあるロールフォワード・データベース・ノートブック、または **db2Rollforward** アプリケーション・プログラミング・インターフェース (API) を通して起動できます。

CLP によって発行する ROLLFORWARD DATABASE コマンドの例を以下に示します。

```
db2 rollforward db sample to end of logs and stop
```

「データベースのロールフォワード (Rollforward Database)」ノートブックをオープンするには、以下のようにします。

1. コントロール・センターから、「データベース (Databases)」フォルダーが見つかるまでオブジェクト・ツリーを展開します。
2. 「データベース (Databases)」フォルダーをクリックします。ウィンドウの右側部分 (目次ペイン) に、既存のデータベースがすべて表示されます。
3. 目次ペイン内で対象となるデータベースをマウスの右ボタンでクリックし、ポップアップ・メニューから「ロールフォワード (Rollforward)」を選択します。「データベースのロールフォワード (Rollforward Database)」ノートブックがオープンします。

詳しい情報については、コントロール・センターのオンライン・ヘルプ機能をご覧ください。

関連概念:

- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『組み込み SQL または DB2 CLI プログラムにおける管理 API』
- 「*管理ガイド: インプリメンテーション*」の『コントロール・センター用のプラグイン・アーキテクチャーの紹介』

関連資料:

- 153 ページの『db2Rollforward - データベースのロールフォワード』

表スペースにおける変更のロールフォワード

データベースが順方向リカバリーであれば、データベース全体を使用する代わりに、表スペースをバックアップしたり、リストアしたり、ロールフォワードしたりできます。個別の表スペースについてリカバリー計画を決めておくこともでき、これにより時間が節約されます。つまり、データベース全体をリカバリーするよりは、データベースの一部をリカバリーした方が時間は短縮されるからです。たとえば、ディスクが不良で、そのディスクに表スペースが 1 つしか含まれていない場合、その表スペースのみをリストアおよびロールフォワードすることができます。その際データベース全体をリカバリーする必要はなく、データベースの残りの部分に対するユーザー・アクセスに影響を与えることもありません。ただし、損傷した表スペースにシステム・カタログ表が含まれている場合は別です。その状態ではデータベースに接続することができません。 (システム・カタログ表を含む表スペース・レベルのバックアップ・イメージが使用可能である場合、システム・カタログ表スペースはそれだけでリストアできます。) 表スペース・レベルのバックアップにより、データベースの重要な部分を他の部分より頻繁にバックアップすることも可能になり、これによりデータベース全体のバックアップより時間は短縮されます。

表スペースがリストアされた後は、常にロールフォワード・ペンディング状態になります。表スペースを使用可能にするためには、表スペースにロールフォワード・

リカバリーを実行する必要があります。ログの最後までロールフォワードすることもでき、特定の時点までロールフォワードすることもできます。ただし、システム・カタログ表を含む表スペースを特定のポイント・イン・タイムまでロールフォワードすることはできません。必ずログの最後までロールフォワードし、データベース内のすべての表スペースに整合性があるようにしてください。

表スペースがロールフォワードされる時、表スペースに影響するログ・レコードが含まれていなくても、 DB2® はすべてのログ・ファイルを処理します。表スペースに影響するログ・レコードが含まれないと分かっているログ・ファイルをスキップするには、 DB2_COLLECT_TS_REC_INFO レジストリー変数を *ON* に設定します。ログ・ファイルのスキップに必要な情報が確実に収集されるようにするには、ログ・ファイルを作成して使用する前に、レジストリー変数を設定する必要があります。

データベース・ディレクトリーにある、表スペース変更履歴ファイル (DB2TSCHG.HIS) は、それぞれの表スペースごとにどのログを処理するべきかを追跡します。 **db2logsForRfwd** ユーティリティーを使用してこのファイルの内容を表示することができ、そこから項目を削除するには、PRUNE HISTORY コマンドを使用します。データベース・リストア操作中、DB2TSCHG.HIS はバックアップ・イメージからリストアされ、その後データベース・ロールフォワード操作中に最新になります。ログ・ファイルの情報が使用できない場合には、ログ・ファイルは、すべての表スペースのリカバリーにそれが必要であるかのように扱われます。

それぞれのログ・ファイルの情報は、ログが非アクティブになった後にディスクにフラッシュされるので、破損の結果としてこの情報が消失する可能性があります。これを補うため、リカバリー操作がログ・ファイルの途中から開始した場合には、ログ全体は、それがシステムのすべての表スペースの変更を含んでいるかのように扱われます。その後、アクティブ・ログが処理され、それらのための情報は再作成されます。より古いログ・ファイルまたはアーカイブログ・ファイルの情報が破損状態で消失し、それらの情報がデータ・ファイル内にない場合には、それらのログ・ファイルは、表スペース・リカバリー操作時に、それらがすべての表スペースの変更を含んでいるものとして扱われます。

表スペースのロールフォワードを実行する前に、 LIST TABLESPACES SHOW DETAIL コマンドを起動してください。このコマンドは、表スペースがロールフォワードできる最早ポイントである *最小リカバリー時間* を戻します。最小リカバリー時間は、データ定義言語 (DDL) ステートメントが表スペースまたは表スペース内の表に対し実行されると、更新されます。表スペースは、システム・カタログ表に含まれる情報と同期するように、少なくとも最小リカバリー時間までロールフォワードする必要があります。複数の表スペースをリカバリーする場合、表スペースは少なくとも、リカバリーされるすべての表スペースの最小リカバリー時間のうちで最大の時間までロールフォワードする必要があります。パーティション・データベース環境では、LIST TABLESPACES SHOW DETAIL コマンドをすべてのパーティションで発行してください。表スペースは、少なくとも、すべてのパーティションにあるすべての表スペースの最小リカバリー時間のうちで最大の時間までロールフォワードする必要があります。

特定の時点にロールフォワードする場合で、表が複数の表スペースに含まれている場合には、すべての表スペースを同時にロールフォワードする必要があります。た

とえば、表データがある表スペースに含まれていて、その表の索引が別の表スペースに含まれている場合は、両方の表スペースを同じ特定の時点まで同時にロールフォワードする必要があります。

表のデータおよび長形式オブジェクトが別々の表スペースに存在する場合に、その長形式オブジェクト・データが再編成された場合は、データと長形式のオブジェクトのための表スペースは同時にリストアし、ロールフォワードする必要があります。表再編成後に、関係する表スペースのバックアップを作成する必要があります。

表スペースを特定の時点までロールフォワードしたい場合で、その表スペースの中の表が次のいずれかである場合には、

- 別の表スペースにあるマテリアライズ照会表またはステージング表の基本表
- 別の表スペースにあるマテリアライズ照会表またはステージング表

両方の表スペースを同じ時点までロールフォワードする必要があります。そのようにしない場合は、チェック・ペンディング状態で、マテリアライズ照会表またはステージング表がロールフォワード操作の最後に置かれます。マテリアライズ照会表は完全に更新される必要があり、ステージング表は不完全とマークされます。

特定の時点まで表スペースをロールフォワードしようとするときに、その表スペースに含まれる表が別の表スペースに含まれる別の表との間で参照保全の関係にある場合は、同じ特定の時点まで両方の表スペースを同時にロールフォワードする必要があります。そのようにしない場合は、チェック・ペンディング状態で、参照保全の関係にある子表がロールフォワード操作の最後に置かれます。子表が後で制約違反のチェックをされた時に、表全体のチェックが必要になります。以下のいずれかの表が存在する場合には、それらも子表とともにチェック・ペンディング状態に置かれます。

- 子表の任意の下層マテリアライズ照会表
- 子表の任意の下層ステージング表
- 子表の任意の下層外部キー表

それらの表は、チェック・ペンディング状態から戻るために完全処理が必要です。両方の表スペースを同時にロールフォワードすると、ポイント・イン・タイム指定ロールフォワード操作の終了時に制約がアクティブなままになります。

ポイント・イン・タイム指定の表スペース・ロールフォワード操作を実行する際、トランザクションが、ある表スペースではロールバックされ、別の表スペースではコミットされるという状態にならないよう注意してください。この状態が発生するのは次の場合です。

- ポイント・イン・タイム指定ロールフォワード操作が、トランザクションにより更新された表スペースのサブセットに実行され、その特定のポイント・イン・タイムがトランザクションのコミットされたポイント・イン・タイムより前になっている。
- 特定の時点までロールフォワードされる表スペースに含まれている表にトリガーが関連付けられているか、ロールフォワードされる表スペース以外の表スペースに影響を与えるトリガーによりその表が更新されている。

解決方法として、この状態が発生しないような適切なポイント・イン・タイムを見つけてください。

QUIESCE TABLESPACES FOR TABLE コマンドを発行して、トランザクションの整合性が保証された時点を確立し、表スペースのロールフォワードに使用することができます。静止要求 (共用、更新意図、または排他モードで) は、 (ロックを使用して) それらの表スペースに対する実行中のトランザクションがすべて完了するのを待ち、新規の要求をブロックします。静止要求が認可されると、表スペースは整合性のある状態になります。ロールフォワードを停止するのに適切な時刻を決定するには、リカバリー履歴ファイルを調べて静止点を見つけ、それらが最小リカバリー時間よりも後に起こったかを確認することができます。

表スペースのポイント・イン・タイム指定ロールフォワード操作が完了した後、表スペースはバックアップ・ペンディング状態になります。ロールフォワードを実行した時点と現在の時間との間の表スペースに対するすべての更新は除去されてしまうため、その表スペースのバックアップを作成する必要があります。表スペースは、直前のデータベース・レベルや表スペース・レベルのバックアップ・イメージからロールフォワードできなくなります。以下の例では、表スペース・レベルのバックアップ・イメージが必要な理由とその使用方法を示しています。 (表スペースを使用可能にするためには、データベース全体、バックアップ・ペンディング状態の表スペース、またはバックアップ・ペンディング状態の表スペースを含む表スペースのセットのいずれかをバックアップできます。)

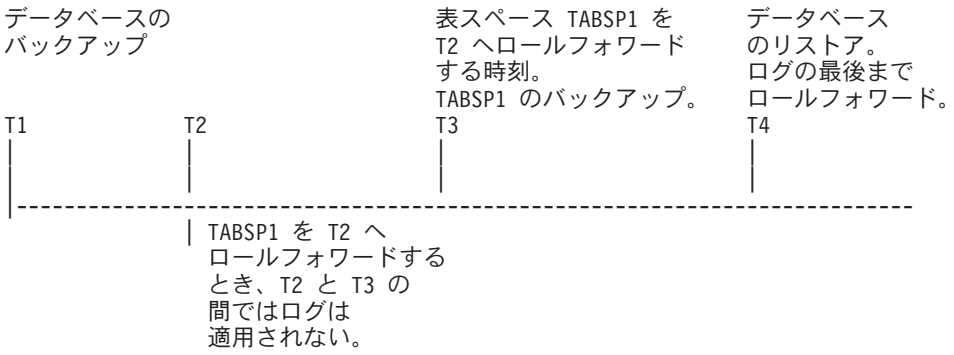


図 15. 表スペースのバックアップ要件

上記の例では、データベースは時刻 T1 にバックアップされます。次に、ポイント・イン・タイム T3 に、表スペース TABSP1 は特定のポイント・イン・タイム (T2) までロールフォワードされ、表スペースはポイント・イン・タイム T3 の後でバックアップされます。表スペースはバックアップ・ペンディング状態であるため、このバックアップ操作は必須です。表スペース・バックアップ・イメージのタイム・スタンプは時刻 T3 より後ですが、表スペースは時刻 T2 にあります。 T2 と T3 の間では、ログ・レコードは TABSP1 には適用されません。時刻 T4 では、データベースが T1 で作成されたバックアップ・イメージを使用してリストアされ、ログの最後までロールフォワードされます。表スペース TABSP1 は時刻 T3 でリストア・ペンディング状態になります。これは、TABSP1 に対し T3 と T4 の間では T2 と T3 の間のログ変更項目を表スペースに適用しないで操作が行われたものとデータベース・マネージャーが想定するからです。これらのログ変更が、実際にはデータベースに対するロールフォワードの一部として適用されていた場合、これは誤った想定となります。表スペース・レベルのバックアップは、表スペースが指定ポイント・イン・タイムまでロールフォワードされた後で作成しなければな

りませんが、このバックアップにより、直前のポイント・イン・タイム指定ロールフォワード操作 (例では T3) の後でもその表スペースをロールフォワードできます。

表スペース TABSP1 を T4 までリカバリーしたい場合は、 T3 の後に作成したバックアップ・イメージ (必須バックアップまたはそれ以後のバックアップ) から表スペースをリストアし、ログの最後まで TABSP1 をロールフォワードします。

上記の例では、時刻 T4 までデータベースをリストアする最も効果的な方法は、必須ステップを以下の順序で実行する方法です。

1. データベースをリストアする。
2. 表スペースをリストアする。
3. データベースをロールフォワードする。
4. 表スペースをロールフォワードする。

データベースをロールフォワードする前に表スペースをリストアするので、データベースをロールフォワードする際にログ・レコードを表スペースに適用するのにリソースを使用することはありません。

時刻 T3 より後の時間の TABSP1 バックアップ・イメージが見つからないか、TABSP1 を T3 またはそれより前にリストアしたい場合は、以下の操作を実行できます。

- 表スペースを T3 までロールフォワードする。表スペースはデータベース・バックアップ・イメージからリストアされているため、表スペースを再びリストアする必要はありません。
- 時刻 T1 で作成したデータベース・バックアップを使用して表スペースを再びリストアし、次に表スペースを時刻 T3 より前の時刻までロールフォワードする。
- 表スペースをドロップする。

パーティション・データベース環境では、以下のようになります。

- 表スペースのすべての部分を同時に同じポイント・イン・タイムまでロールフォワードする必要があります。これにより、表スペースはそれぞれのデータベース・パーティションで整合性が保証されます。
- 一部のデータベース・パーティションがロールフォワード・ペンディング状態で、さらに他のデータベース・パーティションで一部の表スペースがロールフォワード・ペンディング状態である (ただしデータベース・パーティション自体はその状態でない) 場合は、まずデータベース・パーティションをロールフォワードし、次に表スペースをロールフォワードします。
- 表スペースをログの最後までロールフォワードする場合は、データベース・パーティションごとにリストアする必要はありません。リカバリーが必要なデータベース・パーティションでのみリストアが必要です。ただし、特定のポイント・イン・タイムまで表スペースをロールフォワードしたい場合は、各データベース・パーティションごとにリストアする必要があります。

関連概念:

- 140 ページの『ロード・コピー・ロケーション・ファイルを使用したデータのリカバリー』

関連資料:

- 143 ページの『ROLLFORWARD DATABASE』

ドロップされた表のリカバリー

ときに、まだ必要なデータのある表をドロップしてしまうことがあります。そのような場合は、表ドロップ操作の後でもリカバリー可能なクリティカル表の作成を考慮するとよいでしょう。

表データを、データベース・リストア操作によってリカバリーし、その後、表がドロップされる前のポイント・イン・タイムまでのデータベース・ロールフォワード操作を実行できます。これはデータベースが大きいと時間がかかり、そのリカバリーのあいだはデータは使用できなくなります。

ドロップされた表をリカバリーする機能により、表スペース・レベルのリストアおよびロールフォワード操作を使用して、ドロップされた表をリカバリーすることができます。これはデータベース・レベルのリカバリーより速く、ユーザーもデータベースをそのまま使用できます。

前提条件:

ドロップされた表がリカバリー可能になるには、表が常駐する表スペースで DROPPED TABLE RECOVERY オプションがオンになっていなければなりません。これは、表スペース作成の間に行うことができます。または、 ALTER TABLESPACE ステートメントを起動することによっても行えます。 DROPPED TABLE RECOVERY オプションは表スペースに固有で、REGULAR 表スペースに限定されます。ある表スペースで、ドロップされた表のリカバリーが可能かどうかを判別するには、 SYSCAT.TABLESPACES カタログ表にある DROP_RECOVERY 列を照会することができます。ドロップした表のリカバリーは、新規に作成されたデータ表スペースにおいてデフォルトで使用可能です。

DROP TABLE ステートメントが、ドロップされた表のドロップが可能になっている表スペースを持つ表に対して実行されると、追加の項目 (ドロップされた表を識別する) がログ・ファイル内に作成されます。項目はリカバリー履歴ファイルでも作成され、これには表を再作成するのに使用できる情報が含まれます。

制限:

ドロップされた表からリカバリー可能なデータのタイプについて、いくつかの制限事項があります。以下のものは、リカバリーすることはできません。

- ラージ・オブジェクト (LOB) または長形式フィールドのデータ。 LARGE 表スペースには DROPPED TABLE RECOVERY オプションはサポートされていません。 LOB または LONG VARCHAR 列を含むドロップされた表をリカバリーしようとすると、生成されるエクスポート・ファイルでこれらの列は NULL に設定されます。 DROPPED TABLE RECOVERY オプションは REGULAR 表スペースにのみ使用できるもので、一時または LARGE 表スペースには使用できません。

- 行タイプと関連したメタデータ。 (データはリカバリーされますが、メタデータはリカバリーされません。)型付き表の階層表にあるデータはリカバリーされます。このデータには、ドロップされた型付き表に現れたよりも詳細にわたる情報が含まれます。

リカバリーされるデータが GRAPHIC または VARGRAPHIC データ・タイプである場合、複数のコード・ページが含まれている可能性があります。このデータをリカバリーするには、IMPORT または LOAD コマンドの USEGRAPHICCODEPAGE ファイル・タイプ修飾子を指定する必要があります。この場合、データをリカバリーするために LOAD コマンドを使用すると、リカバリー操作のパフォーマンスが改善されます。

手順:

一度でリカバリーできるドロップされた表は 1 つだけです。以下のようにして、ドロップされた表をリカバリーできます。

1. LIST HISTORY DROPPED TABLE コマンドを起動して、ドロップされた表を識別します。ドロップされた表の ID は、Backup ID 列にリストされます。
2. 表がドロップされる前に作成されたデータベース・レベルまたは表スペース・レベルのバックアップ・イメージをリストアします。
3. 表データが含まれているファイルを書き込むエクスポート・ディレクトリーを作成します。このディレクトリーは、すべてのデータベース・パーティションからアクセスできるものか、またはそれぞれのパーティションに存在するかのいずれかでなければなりません。このエクスポート・ディレクトリーの下のサブディレクトリーは、各データベース・パーティションごとに自動的に作成されます。これらのサブディレクトリーの名前は NODE*nnnn* です。ここで、*nnnn* はデータベース・パーティションまたはノード番号を表します。それぞれのデータベース・パーティションにあったときと同じ、ドロップされた表データを含むデータ・ファイルは、 data という下位のサブディレクトリーにエクスポートされます。たとえば、¥export_directory¥NODE0000¥data。
4. ROLLFORWARD DATABASE コマンドの RECOVER DROPPED TABLE オプションを使用して、表がドロップされた後の特定のポイント・イン・タイムまでロールフォワードします。または、ログの最後までロールフォワードします。こうすると、表スペースまたはデータベース内の他の表への更新は失われません。
5. リカバリー履歴ファイルから CREATE TABLE ステートメントを使用して表を再作成します。
6. ロールフォワード操作中にエクスポートされた表データを、表にインポートします。

DATALINK 列に関連したリンク・ファイルの名前はリカバリーできます。表データのインポートの後に、表を DB2 #ata Links Manager#に合わせて調整する必要があります。ファイルのバックアップは、ガーベッジ・コレクションですでに削除されたかどうかによって、 DB2 Data Links Manager によりリカバリーされる場合とリカバリーされない場合があります。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

- 143 ページの『ROLLFORWARD DATABASE』
- 298 ページの『LIST HISTORY』

ロード・コピー・ロケーション・ファイルを使用したデータのリカバリー

DB2LOADREC レジストリー変数を使って、ロード・コピー所在情報のあるファイルを識別します。このファイルは、ロールフォワード・リカバリー中にロード・コピーの位置情報として使われます。次の情報が含まれています。

- メディアの種類
- 使用するメディア装置の数
- 表のロード操作中に生成されるロード・コピーの位置
- ロード・コピーのファイル名 (もしあれば)

ロケーション・ファイルが存在しない場合、あるいはファイル内に一致する項目がない場合は、ログ・レコードからの情報が使われます。

ファイル内の情報は、ロールフォワード・リカバリーの実行前に上書きされることがあります。

注:

1. パーティション・データベース環境では、 **db2set** コマンドを使用して、すべてのデータベース・パーティション・サーバーに DB2LOADREC レジストリー変数を設定する必要があります。
2. パーティション・データベース環境では、ロード・コピー・ファイルは各データベース・パーティション・サーバーに存在していなければならず、ファイル名 (パスも含め) は同じでなければなりません。
3. DB2LOADREC レジストリー変数により識別されるファイルの項目が無効な場合には、古いロード・コピーのロケーション・ファイルが使用され、無効な項目に代わって情報を提供します。

次の情報がロケーション・ファイル内にあります。最初の 5 つのパラメーターには有効な値を指定する必要があり、これらのパラメーターはロード・コピーを識別するために使われます。ロード・コピーが記録されるたびに、この構造全体が繰り返されます。たとえば、次のとおりです。

```
TIMestamp      19950725182542         * Time stamp generated at load time
SCHema         PAYROLL                * Schema of table loaded
TABlename      EMPLOYEES              * Table name
DATabasename   DBT                    * Database name
DB2instance    toronto                * DB2INSTANCE
BUFfernumber   NULL                   * Number of buffers to be used for
                                        recovery
SESsionnumber  NULL                   * Number of sessions to be used for
                                        recovery
TYPeofmedia    L                      * Type of media - L for local device
                                                         A for TSM
                                                         O for other vendors
LOCationnumber 3                      * Number of locations
   ENTry       /u/toronto/dbt.payroll.employes.001
   ENT         /u/toronto/dbt.payroll.employes.002
   ENT         /dev/rmt0
TIM            19950725192054
SCH            PAYROLL
```

```
TAB             DEPT
DAT             DBT
DB2®             toronto
BUF             NULL
SES             NULL
TYP             A
TIM             19940325192054
SCH             PAYROLL
TAB             DEPT
DAT             DBT
DB2             toronto
BUF             NULL
SES             NULL
TYP             O
SHRlib          /@sys/lib/backup_vendor.a
```

注:

1. 各キーワードの最初の 3 文字が有効です。すべてのキーワードが必須で、指定のとおりの順序でなければなりません。ブランク行は使用できません。
2. タイム・スタンプの形式は *yyyymmddhhmmss* です。
3. フィールドはすべて必須ですが、BUF と SES だけは NULL にすることができます。 SES が NULL の場合は、*numloadrecses* 構成パラメーターで指定されている値が使用されます。 BUF が NULL の場合は、デフォルト値は SES+2 です。
4. ロケーション・ファイルに無効な項目が 1 つでもあれば、それらの値を提供するために、以前のロード・コピー・ロケーション・ファイルが使用されます。
5. メディアの種類は、ローカル装置 (テープ、ディスク、またはディスケットの場合 L)、 TSM (A)、あるいは他のベンダー (0) を指定できます。種類が L の場合、ロケーション数とその後に続くロケーション項目が必要です。種類が A の場合、それ以上入力する必要はありません。種類が 0 の場合は、共用ライブラリー名を指定する必要があります。
6. SHRlib パラメーターは、ロード・コピー・データを保管するための機能を備えたライブラリーを指しています。
7. COPY NO または NONRECOVERABLE オプションを指定してロード操作を起動し、操作の完了後にデータベースや影響を受ける表スペースのバックアップ・コピーを取っていないなら、ロード操作後の特定のポイント・イン・タイムまでデータベースや表スペースをリストアすることはできません。つまり、ロールフォワード・リカバリーを使ってもデータベースや表スペースをロード操作後の状態に再構築することはできません。データベースや表スペースをリストアできるのは、ロード操作より前の時点の状態だけです。

特定のロード・コピーを使用したい場合には、データベースのリカバリー履歴ファイルを使用してその特定のロード操作のタイム・スタンプを判別することができます。パーティション・データベース環境の場合、リカバリー履歴ファイルは各データベース・パーティションにローカルに存在します。

関連資料:

- 371 ページの『付録 F. Tivoli Storage Manager』

パーティション・データベース・システムにおけるクロックの同期化

データベース・パーティション・サーバー全体で相対的な同期がとれたシステム・クロックを維持し、データベース操作がスムーズに行われ、順方向リカバリー性に制限が加わらないようにします。データベース・パーティション・サーバー間の時間差にトランザクションの操作および通信遅延時間を加えたものは、 *max_time_diff* (ノード間最大時間差) データベース・マネージャー構成パラメーターの値より小さくしてください。

ログ・レコードのタイム・スタンプがトランザクションの順序を確実に示すようにするために、パーティション・データベース・システムの DB2® は、ログ・レコードに記録するタイム・スタンプの基準として、各マシンのシステム・クロックを使用します。しかし、システム・クロックを進めると、ログ・クロックはそれに合わせて自動的に進みます。システム・クロックを遅らせることはできますが、ログのクロックを遅らせることはできず、システム・クロックをこの時刻に合わせるまでは、ずっと 進んだ時刻のままです。このとき、両方のクロックは同期しています。このことは、データベース・ノードで発生するシステム・クロック・エラーが短期間であっても、データベース・ログのタイム・スタンプではその影響が長く続く場合があることを意味します。

仮説的な例として、データベース・パーティション・サーバー A のシステム・クロックを、 1997 年に間違って 1999 年 11 月 7 日に設定したものとします。また、その誤りは訂正されましたが、そのデータベース・パーティション・サーバーのパーティションで更新トランザクションがコミットされた後 であったとします。そのデータベースが連続的に使用されていてその間で定期的に更新されていると、 1997 年 11 月 7 日から 1999 年 11 月 7 日までの間の任意の時点は、ロールフォワード・リカバリーでは実際には処理不能になります。データベース・パーティション・サーバー A でコミット (COMMIT) が実行されると、データベース・ログのタイム・スタンプは 1999 に設定され、データベース・ログのクロックは 1999 年 11 月 7 日のままです。この状態は、システム・クロックがこの時間と一致するまで続きます。この時間フレーム内の時点へロールフォワードしようとすると、指定された停止点 (1997 年 11 月 7 日) を超えた最初のタイム・スタンプで操作は停止します。

DB2 ではシステム・クロックに対する更新を制御することはできませんが、*max_time_diff* データベース・マネージャー構成パラメーターを指定しておくと、次のような問題が発生するのを防ぐことができます。

- このパラメーターに指定できる値は、1 分から 24 時間までです。
- 非カタログ・ノードに最初の接続要求が出されると、データベース・パーティション・サーバーはその時間をデータベースのカタログ・ノードに送信します。カタログ・ノードはその時間を受け取ると、接続を要求するノードの時間と自分自身の時間が、 *max_time_diff* パラメーターで指定された範囲内であることをチェックします。この範囲を超えると、接続は拒否されます。
- 更新トランザクションがデータベース内の 3 つ以上のデータベース・パーティション・サーバーに関係している場合は、トランザクションは関係するデータベース・パーティション・サーバー間で同期がとれていることを確認した後、更新をコミットします。複数のデータベース・パーティション・サーバーの時間差が、

max_time_diff で指定した値を超えていると、トランザクションはロールバックされ、他のデータベース・パーティション・サーバーへ正しくない時間が伝送されるのを防ぎます。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『max_time_diff - 「ノード間最大時差」構成パラメーター』

クライアント/サーバーのタイム・スタンプの変換

ここでは、クライアント/サーバー環境でのタイム・スタンプの生成について説明します。

- ロールフォワード操作のために現地時間を指定する場合は、戻されるすべてのメッセージも現地時間でなければなりません。

  注: すべての時刻は、サーバー上および (パーティション・データベース環境では) カタログ・ノード上で変換されます。

- タイム・スタンプ・ストリングは、サーバー上で GMT に変換されますので、時刻はクライアントの時間帯ではなく、サーバーの時間帯を表します。クライアントがサーバーとは異なる時間帯にある場合には、サーバーの現地時間を使用する必要があります。
- タイム・スタンプ・ストリングが夏時間調整時のための時刻変更が近づいている場合には、その停止時刻が時刻変更の前か後かを知ることは、正確にその時刻を指定するために重要です。

関連概念:

- 129 ページの『ロールフォワードの概要』
- 142 ページの『パーティション・データベース・システムにおけるクロックの同期化』

ROLLFORWARD DATABASE

データベースのログ・ファイルに記録されたトランザクションを適用することによって、データベースをリカバリーします。データベースまたは表スペースのバックアップ・イメージがリストアされた後、あるいはすべての表スペースがメディア・エラーのためにデータベースによってオフラインにされた場合に呼び出されます。ロールフォワード・リカバリーを使用してデータベースをリカバリーするには、前もってデータベースがリカバリー可能になっていなければなりません (すなわち、データベース構成パラメーター *logarchmeth1* または *logarchmeth2* が OFF 以外の値に設定されていなければなりません)。

有効範囲:

パーティション・データベース環境では、このコマンドはカタログ・パーティションからしか呼び出せません。ポイント・イン・タイムへのデータベースまたは表スペースのロールフォワード操作は、 db2nodes.cfg ファイルにリストされているすべてのパーティションに影響を与えます。ログの終わりへのデータベースまたは表スペースのロールフォワード操作は、指定されたパーティションに影響を与えま

す。パーティションが指定されない場合、コマンドは、 db2nodes.cfg ファイルにリストされているすべてのパーティションに影響を与えます。特定のパーティションでロールフォワードが必要とされない場合には、そのパーティションは無視されます。

権限:

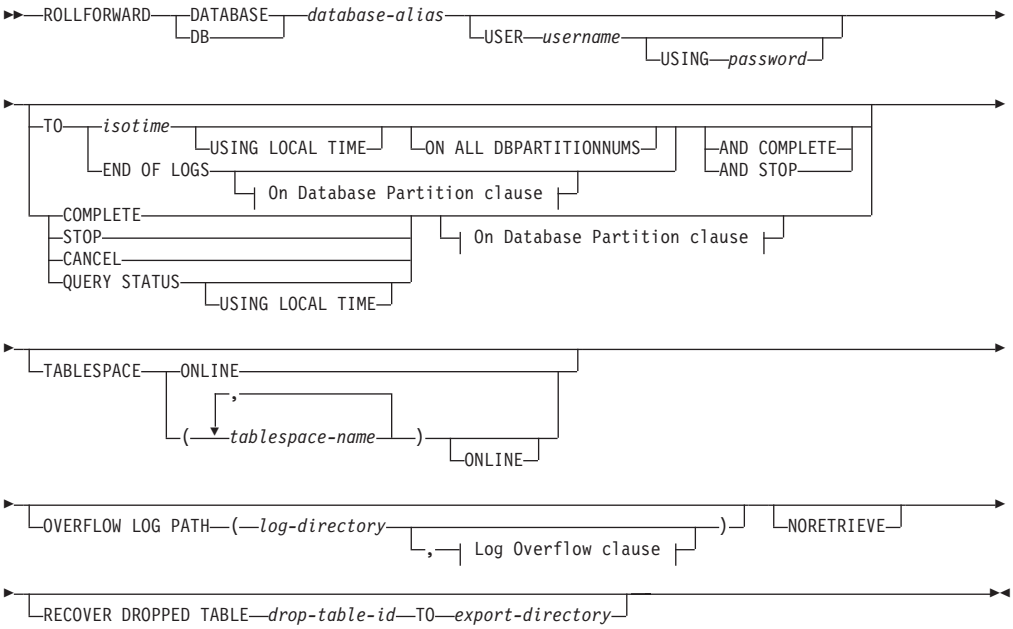
以下のいずれかが必要です。

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

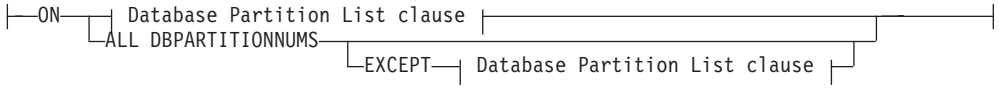
必要な接続:

なし。このコマンドは、データベース接続を確立します。

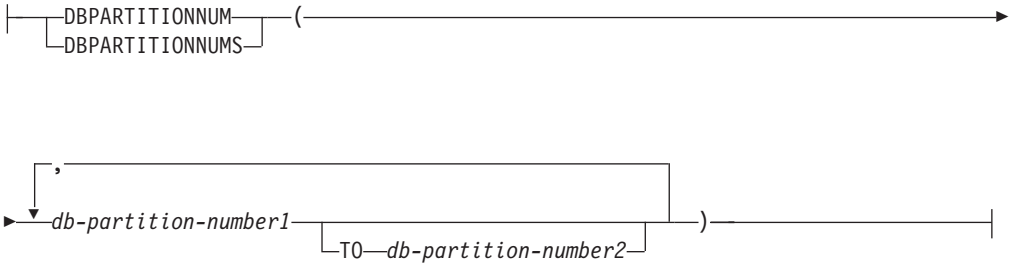
コマンド構文:



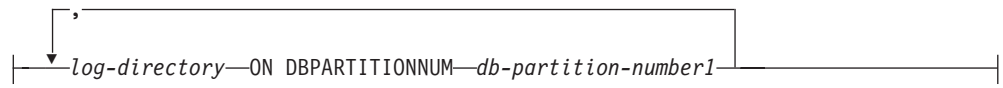
On Database Partition clause:



Database Partition List clause:



Log Overflow clause:



コマンド・パラメーター:

DATABASE database-alias
ロールフォワード・リカバリーするデータベースの別名。

USER username
データベースがロールフォワード・リカバリーされる際のユーザー名。

USING password
ユーザー名を認証するために使用するパスワード。パスワードを省略すると、ユーザーに入力を求めるプロンプトが出ます。

TO

isotime
コミットされたすべてのトランザクションがロールフォワードされる時点 (その時点の前にコミットされたすべてのトランザクションのほかに、ちょうどその時点にコミットされたトランザクションを含む)。

この値は、日付と時刻の組み合わせを識別する 7 つの部分の文字ストリングからなる、タイム・スタンプとして指定します。形式は、*yyyy-mm-dd-hh.mm.ss.nnnnnn* (年、月、日、時、分、秒、マイクロ秒) の協定世界時 (UTC) です。 UTC は、さまざまなログに関連したタイム・スタンプが (たとえば、夏時間に関連して時間が変更されることによって) 同じにならないようにするのに役立ちます。バックアップ・イメージのタイム・スタンプは、バックアップ操作が開始した地方時に基づいています。 CURRENT TIMEZONE 特殊レジスターは、UTC とアプリケーション・サーバーの地方時との時差を指定します。時差は、時刻期間 (最初の 2 桁が時間数、次の 2 桁が分数、最後の 2 桁が秒数を表す 10 進数) で表されます。地方時から CURRENT TIMEZONE を減算すると、地方時を UTC に変換できます。

USING LOCAL TIME
ユーザーが GMT 時間ではなくユーザーの現地時間を使用して特定の時点にロールフォワードすることを可能にします。これによって、ユーザーがローカル・マシンで特定の時点にロールフォワードすることが容易になり、現地時間を GMT ポイント・イン・タイムに変換することによって生じる潜在的なユーザー・エラーの発生を除去します。

注:

1. ユーザーがロールフォワードのために現地時間を指定した場合、ユーザーに戻されるすべてのメッセージも現地時間で表示されます。すべての時刻はサーバー上で変換され、パーティション・データベース環境の場合にはカタログ・データベース・パーティション上で変換されることに注意してください。

2. タイム・スタンプ・ストリングはサーバー上で GMT に変換されるので、時間はクライアントではなくサーバーの時間帯に基づく現地時間となります。クライアントの時間帯とサーバーの時間帯とが異なる場合、サーバーの現地時間を使用してください。これはコントロール・センターの現地時間オプションとは異なります。そのオプションは、クライアントの現地時間を使用します。
3. タイム・スタンプ・ストリングが夏時間調整のための時間変更に接近している場合、停止時刻が時間変更の前か後かを判別して、それを適切に指定することが大切です。

END OF LOGS
データベースの構成パラメーターである *logpath* にリストされる、すべてのオンライン・アーカイブ・ログ・ファイルからコミットされた全トランザクションが、適用されることを指定します。

ALL DBPARTITIONNUMS
`db2nodes.cfg` ファイルで指定されている、すべてのパーティションについてトランザクションがロールフォワードされることを指定します。データベース・パーティション文節が指定されていない場合、これがデフォルトです。

EXCEPT
パーティション・リストに指定されているパーティションを除き、`db2nodes.cfg` ファイルで指定されている、すべてのパーティションについてトランザクションがロールフォワードされることを指定します。

ON DBPARTITIONNUM / ON DBPARTITIONNUMS
一連のデータベース・パーティションについてデータベースをロールフォワードします。

db-partition-number1
データベース・パーティション・リスト内のデータベース・パーティション番号を指定します。

db-partition-number2
2 番目のデータベース・パーティション番号を指定して、*db-partition-number1* から *db-partition-number2* までのすべてのパーティションがデータベース・パーティション・リストに含まれるようにします。

COMPLETE / STOP
ログ・レコードのロールフォワードを停止し、未完了のトランザクションをロールバックし、データベースのロールフォワード・ペンディング状態をオフにすることによって、ロールフォワード・リカバリー処理を完了します。これにより、すでにロールフォワードされたデータベースまたは表スペースへのアクセスが認められます。これらのキーワードは同等です。どちらか一方を指定し、両方は指定しないでください。キーワード AND を使用すると、一度に複数の操作を指定することができます (たとえば、 `db2 rollforward db sample to end of logs and complete`)。

注: 表スペースをある時点までロールフォワードすると、表スペースはバックアップ・ペンディング状態に置かれます。

CANCEL

ロールフォワード・リカバリー操作を取り消します。これにより、順方向リカバリーが開始されている、すべてのパーティションのデータベースまたは 1 つ以上の表スペースがリストア・ペンディング状態になります。

- データベースのロールフォワード操作が進行中ではない (つまり、データベースがロールフォワード・ペンディング状態である) 場合、このオプションは、データベースをリストア・ペンディング状態にします。
- 表スペースのロールフォワード操作が進行中ではない (つまり、表スペースがロールフォワード・ペンディング状態である) 場合は、表スペース・リストを指定しなければなりません。リスト内のすべての表スペースが、リストア・ペンディング状態になります。
- 表スペースのロールフォワード操作が*進行中*である (つまり、少なくとも 1 つの表スペースがロールフォワード進行状態にある) 場合は、ロールフォワード進行中状態にあるすべての表スペースがリストア・ペンディング状態になります。表スペース・リストを指定する場合、そのリストには、ロールフォワード進行中状態にある表スペースがすべて含まれていなければなりません。リストのすべての表スペースが、リストア・ペンディング状態になります。
- ある時点までロールフォワードする場合、渡される表スペース名はすべて無視され、ロールフォワード進行中状態にある表スペースがすべてリストア・ペンディング状態になります。
- 表スペース・リストを指定してログの終わりまでロールフォワードする場合は、リストされている表スペースのみがリストア・ペンディング状態になります。

このオプションは、*実際に実行されている* ロールフォワード操作を取り消すために使用することはできません。このオプションは、その時点で実際に実行されているロールフォワードではなく、進行中のロールフォワード操作を取り消すためにだけに使用できます。以下の場合に、ロールフォワード操作は実行中ではなく進行中の可能性があります。

- 異常終了した。
- STOP オプションが指定されなかった。
- エラーのために失敗した。表スペースをリストア・ペンディング状態にするエラーもあります。たとえば、リカバリー不能ロード操作によるロールフォワードなどです。

注: このオプションを使用する場合は注意し、いくつかの表スペースがロールフォワード・ペンディング状態またはリストア・ペンディング状態になっているために、進行中のロールフォワード操作が完了しない場合だけ使用してください。はっきりと分からない場合は、表スペースがロールフォワード進行中状態になっているのか、ロールフォワード・ペンディング状態になっているのかを識別するために、LIST TABLESPACES コマンドを使用してください。

QUERY STATUS

データベース・マネージャーがロールフォワードしたログ・ファイル、次に必要とされるアーカイブ・ファイル、およびロールフォワード処理が開始されてから、最後にコミットされたトランザクションのタイム・スタンプ

(CUT 形式) をリストします。パーティション・データベース環境では、この状況情報は各パーティションに関して戻されます。戻される情報には、次のフィールドが含まれています。

データベース・パーティション番号

ロールフォワード状況
状態は次のいずれかです。データベースまたは表スペースのロールフォワード・ペンディング、データベースまたは表スペースのロールフォワード進行中、データベースまたは表スペースのロールフォワード処理の停止、またはペンディングなし。

読み込む予定の次のログ・ファイル
次に必要なログ・ファイルの名前から成るストリング。パーティション・データベース環境で、ロールフォワード・ユーティリティーに障害が起こり、ログ・ファイルの欠落を示す戻りコード、またはログ情報の不一致が生じたことを示す戻りコードが戻されたとき、この情報を使用します。

処理済みログ・ファイル
これ以上リカバリーに必要なく、そのディレクトリーから除去できる、処理済みのログ・ファイルの名前から成るストリング。たとえば、最も古い非コミット・トランザクションがログ・ファイル *x* で開始する場合は、古くなったログ・ファイルの範囲には *x* が含まれなくなり、範囲は *x* - 1 で終了します。

コミットされた最終トランザクション
ISO 書式のタイム・スタンプから成る文字列 (*yyyy-mm-dd-hh.mm.ss*)。このタイム・スタンプは、ロールフォワード・リカバリー処理の完了後に、コミットされた最終トランザクションを示しています。タイム・スタンプはデータベースに適用されます。表スペースのロールフォワード・リカバリーでは、データベースにコミットされた最終トランザクションのタイム・スタンプを表します。

注: TO、STOP、COMPLETE、または CANCEL 文節を省略すると、QUERY STATUS がデフォルト値になります。 TO、STOP、または COMPLETE を指定した場合は、コマンドが正常に実行されれば、状況情報が表示されます。個々の表スペースを指定する場合は、それらの表スペースは無視されます。状況要求は、指定された表スペースだけに適用されるものではないからです。

TABLESPACE
このキーワードは、表スペース・レベルのロールフォワード・リカバリーを行う場合に使用します。

tablespace-name
ある時点までの表スペース・レベルのロールフォワード・リカバリーを行う場合は必須です。表スペースのサブセットに対してログの終わりまでのロールフォワード・リカバリーを指定できます。パーティション・データベース環境では、リストの各表スペースがロールフォワードされている各パーティションに存在している必要はありません。もし表スペースが存在している場合、その表スペースは適正な状態にあることを示します。

ONLINE
このキーワードは、表スペース・レベルのロールフォワード・リカバリーをオンラインで行えるようにするために指定します。これは、ロールフォワード・リカバリーの進行中に、他のエージェントが接続できることを意味します。

OVERFLOW LOG PATH log-directory
リカバリー処理中にアーカイブされたログを探索する代替のログ・パスを指定します。ログ・ファイルが *logpath* データベース構成パラメーターで指定されるロケーション以外のロケーションに移動された場合に、このパラメーターを使用します。パーティション・データベース環境では、これはすべてのパーティションの (完全修飾) デフォルト・オーバーフロー・ログ・パスです。単一パーティション・データベースの場合は、相対オーバーフロー・ログ・パスを指定できます。

注: OVERFLOW LOG PATH コマンド・パラメーターは、データベース構成パラメーター OVERFLOWLOGPATH の値 (存在する場合) を上書きします。

log-directory ON DBPARTITIONNUM
パーティション・データベース環境では、特定のパーティションのデフォルト・オーバーフロー・ログ・パスを別のログ・パスでオーバーライドできます。

NORETRIEVE
ユーザーがアーカイブ・ログの検索を使用不可にできるようにして、ユーザーがスタンバイ・マシン上のどのログ・ファイルをロールフォワードするかを制御することを可能にします。これには以下の利点があります。

- ロールフォワードするログ・ファイルを制御することにより、スタンバイ・マシンが稼働マシンよりも必ず X 時間遅れているようにして、ユーザーが両方のシステムに影響を与えることを防止できます。
- スタンバイ・システムがアーカイブにアクセスできない場合 (たとえば、TSM がアーカイブの場合にはオリジナル・マシンだけがこのファイルを検索できます)
- 実動システムがファイルをアーカイブしている際に、スタンバイ・システムが同じファイルを検索していることも可能であり、その場合には不完全なログ・ファイルが取得されます。この問題は Noretrieve を使用して解決できます。

RECOVER DROPPED TABLE drop-table-id
ドロップされた表をロールフォワード操作中にリカバリーします。表 ID を取得するには、LIST HISTORY コマンドを使用します。

TO export-directory
表データが含まれているファイルを書き込むディレクトリーを指定します。指定するディレクトリーは、すべてのデータベース・パーティションにアクセスできるものでなければなりません。

例:

例 1

ROLLFORWARD DATABASE コマンドでは、それぞれをキーワード AND で区切ることによって、一度に複数の操作を指定することができます。たとえば、ログの終わりまでロールフォワードし、完了する場合、コマンドを別々に指定すると、次のようになります。

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete
```

これらは次のように結合することができます。

```
db2 rollforward db sample to end of logs and complete
```

上記の 2 つは同じですが、このような操作は 2 つのステップで実行することをお勧めします。ロールフォワード操作が期待どおりに進行したことを確認してから、ロールフォワード操作を停止することが重要です。そうしない場合、ログが失われる可能性があります。これは、ロールフォワード・リカバリー中に不良ログが検出され、不良ログが「ログの終わり」を意味すると解釈される場合は特に重要です。このような場合は、それ以降のログのロールフォワード操作を続けるために、そのログの損傷していないバックアップ・コピーを使用することができます。

例 2

ログの終わりまでロールフォワードします (2 つの表スペースがリストアされています)。

```
db2 rollforward db sample to end of logs
db2 rollforward db sample to end of logs and stop
```

これら 2 つのステートメントは同等です。ログの終わりまで表スペースのロールフォワード操作を実行する場合、 AND STOP または AND COMPLETE を使用する必要はありません。表スペース名は必須ではありません。指定しない場合には、ロールフォワード・リカバリーを必要としているすべての表スペースが組み込まれます。これらの表スペースの一部のみをロールフォワードする場合は、それらの名前を指定しなければなりません。

例 3

3 つの表スペースがリストアされた後、1 つをログの終わりまでロールフォワードし、他の 2 つをある時点までロールフォワードします (両方ともオンラインで行われます)。

```
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
```

2 つのロールフォワード操作が並行して実行されるわけではないことに注意してください。 2 番目のコマンドは、最初のロールフォワード操作が正常に完了した場合にのみ起動することができます。

例 4

データベースをリストアした後、 OVERFLOW LOG PATH でユーザー出口がアーカイブ・ログを保管するディレクトリーを指定して、ある時点までロールフォワードします。

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

例 5 (パーティション・データベース環境)

0、1、および 2 の 3 つのデータベース・パーティションがあります。表スペース TBS1 はすべてのパーティションで定義されており、表スペース TBS2 はパーティション 0 および 2 で定義されています。データベース・パーティション 1 でデータベースをリストアし、データベース・パーティション 0 および 2 で TBS1 をリストアした後、データベース・パーティション 1 でデータベースをロールフォワードします。

```
db2 rollforward db sample to end of logs and stop
```

これにより、警告 SQL1271 (「データベースは回復されましたが、データベース・パーティション 0 および 2 で 1 つ以上の表スペースがオフラインになっています。」) が戻されます。

```
db2 rollforward db sample to end of logs
```

これにより、データベース・パーティション 0 および 2 で TBS1 がロールフォワードされます。この場合、文節 TABLESPACE(TBS1) はオプションです。

例 6 (パーティション・データベース環境)

データベース・パーティション 0 および 2 でのみ表スペース TBS1 をリストアした後、データベース・パーティション 0 および 2 で TBS1 をロールフォワードします。

```
db2 rollforward db sample to end of logs
```

データベース・パーティション 1 は無視されます。

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

データベース・パーティション 1 で TBS1 がロールフォワード・リカバリー可能な状態になっていないため、これは失敗します。 SQL4906N が報告されます。

```
db2 rollforward db sample to end of logs on dbpartitionnums (0, 2)
   tablespace(TBS1)
```

これは正常に完了します。

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

データベース・パーティション 1 で TBS1 がロールフォワード・リカバリー可能な状態になっていないため、これは失敗します。すべての部分は一緒にロールフォワードされなければなりません。

注: 表スペースをある時点までロールフォワードする場合、データベース・パーティション文節は受け入れられません。ロールフォワード操作は、表スペースが存在するすべてのデータベース・パーティションで行う必要があります。

データベース・パーティション 1 で TBS1 をリストアした後

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

これは正常に完了します。

例 7 (パーティション・データベース環境)

すべてのデータベース・パーティションで表スペースをリストアした後、 PIT2 までロールフォワードしますが、AND STOP は指定しません。ロールフォワード操作はまだ進行中です。それを取り消し、PIT1 までロールフォワードします。

```
db2 rollforward db sample to pit2 tablespace(TBS1)
db2 rollforward db sample cancel tablespace(TBS1)

** restore TBS1 on all database partitions **

db2 rollforward db sample to pit1 tablespace(TBS1)
db2 rollforward db sample stop tablespace(TBS1)
```

例 8 (パーティション・データベース環境)

db2nodes.cfg ファイルにリスト表示されている 8 個のデータベース・パーティション (3 から 10 まで) に存在する表スペースをロールフォワード・リカバリーします。

```
db2 rollforward database dwtest to end of logs tablespace (tssprodt)
```

ログの終わり (ある時点ではなく) までのこの操作は正常に完了します。表スペースが存在するデータベース・パーティションは指定する必要がありません。ユーティリティーは、デフォルトとして db2nodes.cfg ファイルを使用します。

例 9 (パーティション・データベース環境)

(データベース・パーティション 6 上の) 単一パーティションのデータベース・パーティション・グループに存在する 6 個の小さな表スペースをロールフォワード・リカバリーします。

```
db2 rollforward database dwtest to end of logs on dbpartitionnum (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

ログの終わり (ある時点ではなく) までのこの操作は正常に完了します。

使用上の注意:

オンライン・バックアップ操作中に作成されたイメージからリストアする場合は、ロールフォワード操作のポイント・イン・タイムは、オンライン・バックアップの完了ポイント・イン・タイムより後でなければなりません。指定時刻の前にロールフォワード操作が停止する場合、データベースはロールフォワード・ペンディング状態になります。表スペースがロールフォワード中の場合は、ロールフォワード進行中状態になります。

1 つ以上の表スペースをある時点までロールフォワードしている場合は、ロールフォワード操作は、最低でも最小リカバリー時間 (この表スペース用のシステム・カタログまたは表への最新の更新) まで継続する必要があります。表スペースの最小リカバリー時間 (協定世界時 (UTC)) は、 LIST TABLESPACES SHOW DETAIL コマンドを使用して検索できます。

データベースのロールフォワードには、テープ装置を使用したロード・リカバリーが必要とされる場合があります。別のテープを求める要求が出された場合は、次のいずれか 1 つで応答できます。

c 続行。警告メッセージを生成した装置の使用を続けます (たとえば、新しいテープをマウントしたときなど)。

d 装置の終了。警告メッセージを生成した装置の使用を停止します (たとえば、それ以上テープがない場合)。

t 終了。すべての装置を終了します。

ロールフォワード・ユーティリティーが、必要とする次のログを検出できない場合は、そのログ名が SQLCA に戻され、ロールフォワード・リカバリーが停止します。使用可能なログがなくなった場合は、ロールフォワード・リカバリーを終了するために STOP オプションを使用します。未完了のトランザクションはロールバックされ、データベースまたは表スペースが確実に整合した状態になるようにします。

互換性:

バージョン 8 より前のバージョンとの互換性:

- キーワード DBPARTITIONNUM の代わりに NODE を使用できます。
- キーワード DBPARTITIONNUMS の代わりに NODES を使用できます。

関連資料:

- 77 ページの『BACKUP DATABASE』
- 102 ページの『RESTORE DATABASE』

db2Rollforward - データベースのロールフォワード

データベース・ログ・ファイルに記録されたトランザクションを適用することによって、データベースをリカバリーします。この API は、データベースまたは表スペースのバックアップがリストアされた後、あるいはメディア・エラーが原因で表スペースがデータベースによってオフラインにされた場合に呼び出されます。ロールフォワード・リカバリーを用いてデータベースをリカバリーするには、前もってデータベースがリカバリー可能でなければなりません (すなわち、データベース構成パラメーター *logarchmeth1* が**オン**に設定されていなければなりません)。

有効範囲:

パーティション・データベース環境では、この API はカタログ・パーティションからのみ呼び出すことができます。データベースまたは表スペースのポイント・イン・タイムを指定したロールフォワード呼び出しは、 `db2nodes.cfg` ファイルにリストされているすべてのデータベース・パーティション・サーバーに影響を与えます。ログの終わりを指定したデータベースまたは表スペース・ロールフォワード呼び出しは、指定されたデータベース・パーティション・サーバーに影響を与えます。データベース・パーティション・サーバーが 1 つも指定されていない場合には、 `db2nodes.cfg` ファイルにリストされているすべてのデータベース・パーティ

ション・サーバーに影響を与えます。特定のデータベース・パーティション・サーバーでロールフォワードが必要ない場合、そのデータベース・パーティション・サーバーは無視されます。

権限:

以下のいずれかが必要です。

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

必要な接続:

なし。この API によってデータベース接続が確立されます。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2Rollforward */
/* ... */
SQL_API_RC SQL_API_FN
db2Rollforward (
  db2Uint32 versionNumber,
  void *pDB2RollforwardStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2RollforwardStruct
{
  struct db2RfwdInputStruct   *piRfwdInput;
  struct db2RfwdOutputStruct  *poRfwdOutput;
} db2RollforwardStruct;

typedef SQL_STRUCTURE db2RfwdInputStruct
{
  sqluint32                   iVersion;
  char                        *piDbAlias;
  db2Uint32                   iCallerAction;
  char                        *piStopTime;
  char                        *piUserName;
  char                        *piPassword;
  char                        *piOverflowLogPath;
  db2Uint32                   iNumChngLgOvrflw;
  struct sqlurf_newlogpath    *piChngLogOvrflw;
  db2Uint32                   iConnectMode;
  struct sqlu_tablespace_bkrst_list *piTablespaceList;
  db2int32                    iAllNodeFlag;
  db2int32                    iNumNodes;
  SQL_PDB_NODE_TYPE           *piNodeList;
  db2int32                    iNumNodeInfo;
  char                        *piDroppedTblID;
  char                        *piExportDir;
  db2Uint32                   iRollforwardFlags;
} db2RfwdInputStruct;

typedef SQL_STRUCTURE db2RfwdOutputStruct
{
  char                        *poApplicationId;
  sqlint32                    *poNumReplies;
  struct sqlurf_info          *poNodeInfo;
} db2RfwdOutputStruct;
```

```
typedef SQL_STRUCTURE sqlurf_newlogpath
{
   SQL_PDB_NODE_TYPE nodenum;
   unsigned short             pathlen;
   char                       logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
} sqlurf_newlogpath;

typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
  long                          num_entry;
  struct sqlu_tablespace_entry *tablespace;
} sqlu_tablespace_bkrst_list;

typedef SQL_STRUCTURE sqlu_tablespace_entry
{
  sqluint32                  reserve_len;
  char                       tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
  char                       filler[1];
} sqlu_tablespace_entry;

typedef SQL_STRUCTURE sqlurf_info
{
   SQL_PDB_NODE_TYPE nodenum;
   sqlint32        state;
   unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
} sqlurf_info;
/* ... */
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2Rollforward */
/* ... */
SQL_API_RC SQL_API_FN
db2gRollforward (
  db2Uint32 versionNumber,
  void *pDB2gRollforwardStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2gRollforwardStruct
{
  struct db2gRfwdInputStruct  *piRfwdInput;
  struct db2RfwdOutputStruct  *poRfwdOutput;
} db2gRollforwardStruct;

SQL_STRUCTURE db2gRfwdInputStruct
{
  db2Uint32                  iDbAliasLen;
  db2Uint32                  iStopTimeLen;
  db2Uint32                  iUserNameLen;
  db2Uint32                  iPasswordLen;
  db2Uint32                  iOvrflwLogPathLen;
  db2Uint32                  iDroppedTblIDLen;
  db2Uint32                  iExportDirLen;
  sqluint32                  iVersion;
  char                       *piDbAlias;
  db2Uint32                  iCallerAction;
  char                       *piStopTime;
  char                       *piUserName;
  char                       *piPassword;
  char                       *piOverflowLogPath;
  db2Uint32                  iNumChngLgOvrflw;
  struct sqlurf_newlogpath   *piChngLogOvrflw;
```

```
  db2Uint32                   iConnectMode;
  struct sqlu_tablespace_bkrst_list *piTablespaceList;
  db2int32                    iAllNodeFlag;
  db2int32                    iNumNodes;
  SQL_PDB_NODE_TYPE           *piNodeList;
  db2int32                    iNumNodeInfo;
  char                        *piDroppedTblID;
  char                        *piExportDir;
  db2Uint32                   iRollforwardFlags;
} db2gRfwdInputStruct;

typedef SQL_STRUCTURE db2RfwdOutputStruct
{
  char                        *poApplicationId;
  sqlint32                    *poNumReplies;
  struct sqlurf_info          *poNodeInfo;
} db2RfwdOutputStruct;

typedef SQL_STRUCTURE sqlurf_newlogpath
{
   SQL_PDB_NODE_TYPE nodenum;
   unsigned short             pathlen;
   char                       logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
} sqlurf_newlogpath;

typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
  long                         num_entry;
  struct sqlu_tablespace_entry *tablespace;
} sqlu_tablespace_bkrst_list;

typedef SQL_STRUCTURE sqlu_tablespace_entry
{
  sqluint32                   reserve_len;
  char                        tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
  char                        filler[1];
} sqlu_tablespace_entry;

typedef SQL_STRUCTURE sqlurf_info
{
   SQL_PDB_NODE_TYPE nodenum;
   sqlint32        state;
   unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
} sqlurf_info;
/* ... */
```

API パラメーター:

versionNumber
入力。 2 番目のパラメーターとして渡される構造のバージョンとリリースのレベルを指定します。

pDB2RollforwardStruct
入力。 *db2RollforwardStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

piRfwdInput
入力。 *db2RfwdInputStruct* 構造を指すポインター。

poRfwdOutput
出力。 *db2RfwdOutputStruct* 構造を指すポインター。

iDbAliasLen
入力。データベース別名の長さ (バイト単位) を指定します。

iStopTimeLen
入力。停止時刻パラメーターの長さ (バイト単位) を指定します。停止時刻が提供されていない場合は、ゼロに設定してください。

iUserNameLen
入力。ユーザー名の長さ (バイト単位) を指定します。ユーザー名が提供されていない場合は、ゼロに設定してください。

iPasswordLen
入力。パスワードの長さ (バイト単位) を指定します。パスワードが提供されていない場合は、ゼロに設定してください。

iOverflowLogPathLen
入力。オーバーフロー・ログ・パスの長さ (バイト単位) を指定します。オーバーフロー・ログ・パスが提供されていない場合は、ゼロに設定してください。

iVersion
入力。ロールフォワード・パラメーターのバージョン ID。
SQLUM_RFWD_VERSION として定義されています。

piDbAlias
入力。データベース別名を含むストリング。これは、システム・データベース・ディレクトリーにカタログされている別名です。

iCallerAction
入力。実行するアクションを指定します。有効な値 (db2ApiDf.h で定義) は、以下のとおりです。

DB2ROLLFORWARD_ROLLFWD
piStopTime で指定された時点までロールフォワードします。データベースのロールフォワードの場合、データベースは*ロールフォワード・ペンディング* 状態のままになります。表スペースのロールフォワードの場合、表スペースは*ロールフォワード処理* 状態のままになります。

DB2ROLLFORWARD_STOP
ロールフォワード・リカバリーを終了します。新しいログ・レコードは処理されず、コミットされていないトランザクションはバックアウトされます。データベースまたは表スペースの*ロールフォワード・ペンディング* 状態はオフになります。同義語は DB2ROLLFORWARD_RFWD_COMPLETE です。

DB2ROLLFORWARD_RFWD_STOP
piStopTime で指定された時点までロールフォワードし、ロールフォワード・リカバリーを終了します。データベースまたは表スペースの*ロールフォワード・ペンディング* 状態はオフになります。同義語は DB2ROLLFORWARD_RFWD_COMPLETE です。

DB2ROLLFORWARD_QUERY
nextarclog、*firstarcdel*、*lastarcdel*、および *lastcommit* の照会値。データベース状況とノード番号を戻します。

DB2ROLLFORWARD_PARM_CHECK
ロールフォワードを実行することなく、パラメーターの妥当性を検査します。

DB2ROLLFORWARD_CANCEL
現在実行中のロールフォワード操作を取り消します。データベースまたは表スペースは、リカバリー・ペンディング状態に置かれます。

注: このオプションは、ロールフォワードが実際に実行中であるときには使用できません。ロールフォワードが休止されている (つまり、STOP を待っている) 場合、あるいはロールフォワード中にシステム障害が発生した場合に使用できます。このオプションの使用に際しては、注意が必要です。

データベースのロールフォワードには、テープ装置を使用したロード・リカバリーが必要とされる場合があります。装置に関してユーザーの介入が必要な場合、ロールフォワード API は警告メッセージを戻します。以下の 3 つのアクションのいずれかを指定して、再び API を呼び出すことができます。

DB2ROLLFORWARD_LOADREC_CONT
警告メッセージを生成した装置の使用を続けます (たとえば、新しいテープをマウントしたときなど)。

DB2ROLLFORWARD_DEVICE_TERM
警告メッセージを生成した装置の使用を停止します (たとえば、それ以上テープがない場合)。

DB2ROLLFORWARD_LOAD_REC_TERM
ロード・リカバリーに使用されているすべての装置を終了させます。

piStopTime
入力。 ISO 形式のタイム・スタンプを含む文字ストリング。このタイム・スタンプで設定された時刻を過ぎると、データベース・リカバリーは停止します。可能な限りロールフォワードしたい場合には、SQLUM_INFINITY_TIMESTAMP を指定してください。DB2ROLLFORWARD_QUERY、DB2ROLLFORWARD_PARM_CHECK、およびいずれかのロード・リカバリー (DB2ROLLFORWARD_LOADREC_*xxx*) 呼び出し側アクションの場合は、NULL にすることができます。

piUserName
入力。アプリケーションのユーザー名を含むストリング。 NULL にすることもできます。

piPassword
入力。提供されたユーザー名 (ある場合) のパスワードを含むストリング。NULL にすることもできます。

piOverflowLogPath
入力。使用される代替ログ・パスを指定します。このユーティリティーを使用する前に、アクティブ・ログ・ファイルの他に、アーカイブ・ログ・ファイルをユーザーが *logpath* に移動させることが必要です。このことは、*logpath* に十分なスペースがない場合に問題になる可能性があります。その問題を解決するために、オーバーフロー・ログ・パスが備えられています。ロールフォワード・リカバリー中に、必要なログ・ファイルは、まず *logpath* で探索され、次にオーバーフロー・ログ・パスで探索されます。表スペースのロールフォワード・リカバリーに必要なログ・ファイルは、*logpath* またはオーバーフロー・ログ・パスのいずれかに置かれる可能性があります。呼び出し側がオーバーフロー・ログ・パスを指定しない場合、デフォルト値は *logpath* です。パーティション・データベース環境では、オーバーフロー・ログ・パスは有効な完全修飾パスでなければなりません。デフォルトのパスは、各ノードのデフォルトのオーバーフロー・ログ・パスです。単一パーティション・データベース環境では、サーバーがローカルであれば、オーバーフロー・ログ・パスは相対パスにすることもできます。

iNumChngLgOvrflw
入力。パーティション・データベース環境のみ。変更されるオーバーフロー・ログ・パスの数。新しいログ・パスは、指定されたデータベース・パーティション・サーバーのデフォルトのオーバーフロー・ログ・パスだけをオーバーライドします。

piChngLogOvrflw
入力。パーティション・データベース環境のみ。変更されるオーバーフロー・ログ・パスの完全修飾名が入っている構造を指すポインター。新しいログ・パスは、指定されたデータベース・パーティション・サーバーのデフォルトのオーバーフロー・ログ・パスだけをオーバーライドします。

iConnectMode
入力。有効な値 (`db2ApiDf.h` で定義) は、以下のとおりです。

DB2ROLLFORWARD_OFFLINE
オフライン・ロールフォワード。データベースのロールフォワード・リカバリーの場合には、必ずこの値を指定してください。

DB2ROLLFORWARD_ONLINE
オンライン・ロールフォワード。

piTablespaceList
入力。ログの終わりまで、または指定された時点までロールフォワードされる表スペースの名前が入っている構造を指すポインター。指定されない場合には、ロールフォワードを必要とする表スペースが選択されます。

iAllNodeFlag
入力。パーティション・データベース環境のみ。ロールフォワード操作が、`db2nodes.cfg` で定義されているすべてのデータベース・パーティション・サーバーに適用されるかどうかを示します。有効な値は以下のとおりです。

DB2_NODE_LIST
piNodeList で渡されたリスト内のデータベース・パーティション・サーバーに適用されます。

DB2_ALL_NODES
すべてのデータベース・パーティション・サーバーに適用されます。 *piNodeList* は NULL でなければなりません。これはデフォルト値です。

DB2_ALL_EXCEPT
piNodeList で渡されたリスト内のデータベース・パーティション・サーバーを除いた、すべてのデータベース・パーティション・サーバーに適用されます。

DB2_CAT_NODE_ONLY
カタログ・パーティションにのみ適用されます。 *piNodeList* は NULL でなければなりません。

iNumNodes
入力。 *piNodeList* 配列内のデータベース・パーティション・サーバーの数を指定します。

piNodeList
入力。ロールフォワード操作を実行する対象のデータベース・パーティション・サーバー番号の配列を指すポインター。

iNumNodeInfo
入力。出力パラメーター *poNodeInfo* のサイズを定義します。これは、ロールフォワードされるそれぞれのデータベース・パーティションからの状況情報を保持するのに十分な大きさでなければなりません。単一パーティション・データベース環境では、このパラメーターは 1 に設定しなければなりません。このパラメーターの値は、この API が呼び出されるデータベース・パーティション・サーバーの数と同じにする必要があります。

piDroppedTblID
入力。リカバリーが実行されているドロップ済み表の ID を含むストリング。

piExportDir
入力。ドロップした表データのエクスポート先のディレクトリー。

RollforwardFlags
入力。ロールフォワード・フラグを指定します。有効な値 (`db2ApiDf.h` で定義) は、以下のとおりです。

DB2ROLLFORWARD_EMPTY_FLAG
フラグが指定されていません。

DB2ROLLFORWARD_LOCAL_TIME
ユーザーが GMT 時間ではなくユーザーの現地時間を使用して特定の時点にロールフォワードすることを可能にします。これによって、ユーザーがローカル・マシンで特定の時点にロールフォワードすることが容易になり、現地時間を GMT ポイント・イン・タイムに変換することによって生じる潜在的なユーザー・エラーの発生を除去します。

DB2ROLLFORWARD_NO_RETRIEVE
ユーザーがアーカイブ・ログの検索を使用不可にすることを許可することによって、スタンバイ・マシン上でどのログ・ファイルがロ

ールフォワードされるべきかを制御します。ログ・ファイルのロールフォワードを制御することによって、スタンバイ・マシンが稼働マシンより X 時間遅れていることを確認でき、ユーザーが両システムに影響を与えることを防ぐことができます。スタンバイ・システムがアーカイブにアクセスせず、たとえば TSM がアーカイブの場合に、元のマシンがファイルを検索することだけを許可する場合に、このオプションは役立ちます。また、実動システムがファイルをアーカイブし、スタンバイ・システムが同じファイルを検索している間、スタンバイ・システムが不完全なログ・ファイルを検索するという可能性も除去します。

poApplicationId
出力。アプリケーション ID。

poNumReplies
出力。受信した応答の数。

poNodeInfo
出力。データベース・パーティション応答情報。

nodenum
ノード番号。

pathlen
新規 logpath の長さ。

logpath
新規オーバーフロー・ログ・パス。

num_entry
tablespace フィールドによって示されたリスト内の項目数。

tablespace
sqlu_tablepsace_entry 構造を指すポインター。

reserve_len
tablespace_entry フィールドに指定された文字ストリングの長さ。 C 言語以外の場合です。

tablespace_entry
表スペース名。

state 状態情報。

nextarclog
次に必要とされるアーカイブ・ログ・ファイルの戻された名前を保管するバッファー。 DB2ROLLFORWARD_QUERY 以外の呼び出し側アクションが指定された場合、このフィールドに戻される値は、ファイルのアクセス時にエラーが発生したことを示すものです。考えられる原因は、次のとおりです。

- ファイルが、データベース・ログ・ディレクトリー内に見付からなかった、またはオーバーフロー・ログ・パス・パラメーターで指定したパスになかった。
- ログ・アーカイブ方式が、アーカイブ・ファイルを戻すことに失敗した。

firstarcdel
リカバリーに必要ではなくなった最初のアーカイブ・ログ・ファイルの戻された名前を保管するバッファー。このファイルおよび lastarcdel (を含む) までのすべてのファイルは、ディスクを空けるために移動することができます。

たとえば、firstarcdel および lastarcdel に戻される値が S0000001.LOG および S0000005.LOG である場合、以下のログ・ファイルを移動できます。

- S0000001.LOG
- S0000002.LOG
- S0000003.LOG
- S0000004.LOG
- S0000005.LOG

lastarcdel
データベース・ログ・ディレクトリーから除去できる最新のアーカイブ・ログ・ファイルの戻された名前を保管するバッファー。

lastcommit
ISO 形式のタイム・スタンプを含むストリング。この値は、ロールフォワード操作の終了後に、最後にコミットされたトランザクションのタイム・スタンプを表します。

使用上の注意:

データベース・マネージャーは、アーカイブおよびログ・ファイルに格納された情報を使用して、最後のバックアップのときにデータベースで実行されたトランザクションを再構築します。

この API の呼び出し時に実行されるアクションは、呼び出し前のデータベースの *rollforward_pending* フラグによって異なります。これは db2CfgGet (構成パラメーターの入手) を使用して照会できます。データベースがロールフォワード・ペンディング状態にある場合、 *rollforward_pending* フラグは DATABASE に設定されています。 1 つまたは複数の表スペースが SQLB_ROLLFORWARD_PENDING または SQLB_ROLLFORWARD_IN_PROGRESS 状態にある場合、フラグは TABLESPACE に設定されています。データベースも表スペースもロールフォワードする必要がない場合は、*rollforward_pending* フラグが NO に設定されています。

この API の呼び出し時にデータベースがロールフォワード・ペンディング状態にある場合は、データベースがロールフォワードされます。表スペースは、異常状態によって 1 つまたは複数の表スペースがオフラインにならない限り、データベースのロールフォワードが正常に終了すると正常の状態に戻ります。 *rollforward_pending* フラグが TABLESPACE に設定されている場合には、ロールフォワード・ペンディング状態にある表スペース、あるいは名前によって要求された表スペースだけがロールフォワードされます。

注: 表スペースのロールフォワードが異常終了してしまった場合、ロールフォワード中だった表スペースは、 SQLB_ROLLFORWARD_IN_PROGRESS 状態に置かれます。次に ROLLFORWARD DATABASE を呼び出したときには、SQLB_ROLLFORWARD_IN_PROGRESS 状態にあるそれらの表スペースだけが処理され

ます。選択された表スペース名のセットに SQLB_ROLLFORWARD_IN_PROGRESS 状態のすべての表スペースが含まれているのではない場合には、要求されていない表スペースが SQLB_RESTORE_PENDING 状態に置かれます。

データベースがロールフォワード・ペンディング状態になく、特定の時点が指定されない場合には、ロールフォワード進行中状態にある表スペースがログの終わりまでロールフォワードされます。ロールフォワード進行中状態の表スペースがない場合には、ロールフォワード・ペンディング状態にある表スペースがログの終わりまでロールフォワードされます。

この API は、ログ・ファイルの読み取りを、バックアップ・イメージに一致するログ・ファイルから始めます。ログ・ファイルをロールフォワードする前に、DB2ROLLFORWARD_QUERY 呼び出し側アクションを指定してこの API を呼び出すと、このログ・ファイルの名前を判別することができます。

ログ・ファイル内のトランザクションは、データベースに再適用されます。ログは、情報が使用可能である限り、あるいは停止時刻パラメーターで指定された時刻まで、順方向に処理されます。

以下のイベントが生じると、リカバリーが停止します。

- ログ・ファイルがこれ以上見つからない。
- ログ・ファイル内のタイム・スタンプが、停止時刻パラメーターで指定された完了タイム・スタンプを超えた。
- ログ・ファイルの読み取り中に、エラーが発生した。

一部のトランザクションは、リカバリーされない可能性があります。 *lascommit* で戻された値は、最後にコミットされ、データベースに適用されたトランザクションのタイム・スタンプを示します。

アプリケーションまたは人為エラーが原因でデータベースのリカバリーが必要となった場合、エラーが発生する前の時点でリカバリーを停止することを指示するために、 *piStopTime* にタイム・スタンプ値を指定することができます。これは、データベースの全ロールフォワード・リカバリーと、表スペースの特定の時点までのロールフォワードに適用されます。また、このことにより、前回失敗したリカバリーの試みで判別された、ログ読み取りエラーが発生する前の時点でリカバリーを停止させることも可能になります。

rollforward_recovery フラグが DATABASE に設定されている場合、ロールフォワード・リカバリーが終了するまで、データベースは使用できません。DB2ROLLFORWARD_STOP または DB2ROLLFORWARD_RFWRD_STOP の呼び出し側アクションを指定してこの API を呼び出すことにより、データベースのロールフォワード・ペンディング状態をオフにすれば、リカバリーを終了させることができます。*rollforward_recovery* フラグが TABLESPACE になれば、データベースを使用できるようになります。ただし、SQLB_ROLLFORWARD_PENDING および SQLB_ROLLFORWARD_IN_PROGRESS 状態の表スペースは、表スペースのロールフォワード・リカバリーを実行するための API が呼び出されるまで使用不可になります。表スペースをある時点までロールフォワードすると、表スペースは、正常なロールフォワード後にバックアップ・ペンディング状態に置かれます。

RollforwardFlags オプションが DB2ROLLFORWARD_LOCAL_TIME に設定されている場合、ユーザーに戻されるすべてのメッセージは現地時間で示されます。パーティション・データベース環境の場合、すべての時間はサーバー、またはカタログ・パーティション上で変換されます。タイム・スタンプ・ストリングはサーバー上で GMT に変換されるので、時間はクライアントではなくサーバーの時間帯に基づく現地時間となります。クライアントの時間帯とサーバーの時間帯とが異なる場合、サーバーの現地時間を使用してください。これはコントロール・センターの現地時間オプションとは異なります。そのオプションは、クライアントの現地時間を使用します。タイム・スタンプ・ストリングが夏時間調整のための時間変更に接近している場合、停止時刻が時間変更の前か後かを判別して、それを適切に指定することが大切です。

関連資料:

- 「*管理 API リファレンス*」の『SQLCA』
- 113 ページの『db2Restore - データベースのリストア』

関連サンプル:

- 『dbrecov.sqc -- How to recover a database (C)』
- 『dbrecov.sqC -- How to recover a database (C++)』

ロールフォワード・セッション - CLP の例

例 1

ROLLFORWARD DATABASE コマンドでは、それぞれをキーワード AND で区切ることによって、一度に複数の操作を指定することができます。たとえば、ログの終わりまでロールフォワードし、完了する場合、コマンドを別々に指定すると、次のようになります。

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete
```

これらは次のように結合することができます。

```
db2 rollforward db sample to end of logs and complete
```

上記の 2 つは同じですが、このような操作は 2 つのステップで実行することをお勧めします。ロールフォワード操作が期待どおりに進行したことを確認してから、ロールフォワード操作を停止することが重要です。そうしない場合、ログが失われる可能性があります。これは、ロールフォワード・リカバリー中に不良ログが検出され、不良ログが「ログの終わり」を意味すると解釈される場合は特に重要です。このような場合は、それ以降のログのロールフォワード操作を続けるために、そのログの損傷していないバックアップ・コピーを使用することができます。

例 2

ログの終わりまでロールフォワードします (2 つの表スペースがリストアされています)。

```
db2 rollforward db sample to end of logs
db2 rollforward db sample to end of logs and stop
```

これら 2 つのステートメントは同等です。ログの終わりまで表スペースのロールフォワード操作を実行する場合、 AND STOP または AND COMPLETE を使用する必要はありません。表スペース名は必須ではありません。指定しない場合には、ロールフォワード・リカバリーを必要としているすべての表スペースが組み込まれます。これらの表スペースの一部のみをロールフォワードする場合は、それらの名前を指定しなければなりません。

例 3

3 つの表スペースがリストアされた後、1 つをログの終わりまでロールフォワードし、他の 2 つをある時点までロールフォワードします (両方ともオンラインで行われます)。

```
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
```

2 つのロールフォワード操作が並行して実行されるわけではないことに注意してください。 2 番目のコマンドは、最初のロールフォワード操作が正常に完了した場合にのみ起動することができます。

例 4

データベースをリストアした後、 OVERFLOW LOG PATH でユーザー出口がアーカイブ・ログを保管するディレクトリーを指定して、ある時点までロールフォワードします。

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

例 5 (パーティション・データベース環境)

0、1、および 2 の 3 つのノードがあります。表スペース TBS1 はすべてのノードで定義されており、表スペース TBS2 はノード 0 および 2 で定義されています。ノード 1 でデータベースをリストアし、ノード 0 および 2 で TBS1 をリストアした後、ノード 1 でデータベースをロールフォワードします。

```
db2 rollforward db sample to end of logs and stop
```

これにより、警告 SQL1271 (「データベースはリカバリーされましたが、ノード 0 および 2 で 1 つまたは複数の表スペースがオフラインになっています。」) が戻されます。

```
db2 rollforward db sample to end of logs
```

これにより、ノード 0 および 2 で TBS1 がロールフォワードされます。この場合、文節 TABLESPACE(TBS1) はオプションです。

例 6 (パーティション・データベース環境)

ノード 0 および 2 でのみ表スペース TBS1 をリストアした後、ノード 0 および 2 で TBS1 をロールフォワードします。

```
db2 rollforward db sample to end of logs
```

ノード 1 は無視されます。

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

ノード 1 で TBS1 がロールフォワード操作可能になっていないため、これは失敗します。 SQL4906N が報告されます。

```
db2 rollforward db sample to end of logs on nodes (0, 2) tablespace(TBS1)
```

これは正常に完了します。

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

ノード 1 で TBS1 がロールフォワード操作可能になっていないため、これは失敗します。すべての部分は一緒にロールフォワードされなければなりません。

注: 表スペースを特定のポイント・イン・タイムまでロールフォワードする場合、ノード文節は受け入れられません。ロールフォワード操作は表スペースが存在するすべてのノードで行わなければなりません。

ノード 1 で TBS1 をリストアした後、

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

これは正常に完了します。

例 7 (パーティション・データベース環境)

すべてのノードで表スペースをリカバリーした後、 PIT2 までロールフォワードしますが、AND STOP は指定しません。ロールフォワード操作はまだ進行中です。それを取り消し、PIT1 までロールフォワードします。

```
db2 rollforward db sample to pit2 tablespace(TBS1)
db2 rollforward db sample cancel tablespace(TBS1)

** restore TBS1 on all nodes **

db2 rollforward db sample to pit1 tablespace(TBS1)
db2 rollforward db sample stop tablespace(TBS1)
```

例 8 (パーティション・データベース環境)

db2nodes.cfg ファイルにリスト表示されている 8 個のノード (3 ～ 10) にある表スペースをロールフォワード・リカバリーします。

```
db2 rollforward database dwtest to end of logs tablespace (tssprodt)
```

ログの終わり (ある時点ではなく) までのこの操作は正常に完了します。表スペースが存在するノードは指定する必要がありません。ユーティリティーは、デフォルトとして db2nodes.cfg ファイルを使用します。

例 9 (パーティション・データベース環境)

(ノード 6 上の) 1 つのノード・データベース・パーティション・グループに存在する 6 個の小さな表スペースを、ロールフォワード・リカバリーします。

```
db2 rollforward database dwtest to end of logs on node (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

ログの終わり (ある時点ではなく) までのこの操作は正常に完了します。

第 5 章 データベースのリカバリー

このセクションでは、DB2 UDB リカバリー・ユーティリティーについて説明します。これは、リカバリー履歴ファイルの情報に基づき、必要なリストアおよびロールフォワード操作を実行してデータベースを指定した時点までリカバリーするものです。

以下のトピックについて説明します。

リカバリーの概要

リカバリー・ユーティリティーは、リカバリー履歴ファイルの情報に基づき、必要なリストアおよびロールフォワード操作を実行してデータベースを指定した時点までリカバリーします。このユーティリティーを使用するときには、データベースを特定時点までまたはログ・ファイルの最後までリカバリーすることを指定します。そうすると、このユーティリティーは、最適なバックアップ・イメージを選択して、リカバリー操作を実行します。

リカバリー・ユーティリティーは、以下の RESTORE DATABASE コマンド・オプションをサポートしません。

- TABLESPACE tablespace-name。表スペース・リストア操作は、サポートされていません。
- INCREMENTAL。増分リストア操作は、サポートされていません。
- OPEN num-sessions SESSIONS。 TSM または他のベンダー製品とともに使用する入出力セッションの数は指定できません。
- BUFFER buffer-size。リストア操作に使用するバッファーのサイズは設定できません。
- DLREPORT filename。リンク解除されるレポート・ファイルのファイル名は指定できません。
- WITHOUT ROLLING FORWARD。正常なリストア操作後にデータベースをロールフォワード・ペンディング状態にしないことを指定できません。
- WITHOUT DATALINK。リンクされたファイルの調整は実行されないことを指定できません。
- PARALLELISM n。リストア操作の並列処理の度合いは指定できません。
- WITHOUT PROMPTING。リストア操作を無人で実行することは指定できません。

関連概念:

- 143 ページの『クライアント/サーバーのタイム・スタンプの変換』

関連タスク:

- 168 ページの『リカバリーの使用』

関連資料:

- 102 ページの『RESTORE DATABASE』
- 40 ページの『データベース・ロギングの構成パラメーター』

リカバリーの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャー保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスが制御されます。ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

リカバリー・ユーティリティーを使用するには、 SYSADM、SYSCTRL、または SYSMAINT 権限が必要です。

リカバリーの使用

前提条件:

リカバリーを実行するデータベースに接続していてはなりません。データベース・リカバリー・ユーティリティーは、指定されたデータベースに自動的に接続を確立し、この接続はリカバリー操作が完了すると終了します。

データベースは、ローカルとリモートであることができます。

手順:

リカバリー・ユーティリティーは、コマンド行プロセッサー (CLP)、または **db2Recover** アプリケーション・プログラミング・インターフェース (API) を通して起動できます。

次の例は、CLP での RECOVER DATABASE コマンドの使用方法を示すものです。

```
db2 recover db sample
```

注: パーティション・データベース環境では、リカバリー・ユーティリティーはデータベースのカタログ・ノードから起動する必要があります。

関連概念:

- 167 ページの『リカバリーの概要』

関連資料:

- 175 ページの『db2Recover - データベースのリカバリー』
- 169 ページの『RECOVER DATABASE』

クライアント/サーバーのタイム・スタンプの変換

ここでは、クライアント/サーバー環境でのタイム・スタンプの生成について説明します。

- ロールフォワード操作のために現地時間を指定する場合は、戻されるすべてのメッセージも現地時間でなければなりません。

  注: すべての時刻は、サーバー上および (パーティション・データベース環境では) カタログ・ノード上で変換されます。
- タイム・スタンプ・ストリングは、サーバー上で GMT に変換されますので、時刻はクライアントの時間帯ではなく、サーバーの時間帯を表します。クライアントがサーバーとは異なる時間帯にある場合には、サーバーの現地時間を使用する必要があります。
- タイム・スタンプ・ストリングが夏時間調整時のための時刻変更が近づいている場合には、その停止時刻が時刻変更の前か後かを知ることは、正確にその時刻を指定するために重要です。

関連概念:

- 129 ページの『ロールフォワードの概要』
- 142 ページの『パーティション・データベース・システムにおけるクロックの同期化』

RECOVER DATABASE

データベースを、特定のポイント・イン・タイムまで、またはログの終わりまでリストアおよびロールフォワードします。

有効範囲:

パーティション・データベース環境では、このコマンドはカタログ・パーティションからしか呼び出せません。ポイント・イン・タイムへのデータベース・リカバリー操作は、 `db2nodes.cfg` ファイルにリストされているすべてのパーティションに影響を与えます。ログの終わりまでのデータベース・リカバリー操作は、指定されたパーティションに影響を与えます。パーティションが指定されない場合、コマンドは、 `db2nodes.cfg` ファイルにリストされているすべてのパーティションに影響を与えます。

権限:

既存のデータベースにリカバリーするには、次のいずれかが必要です。

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

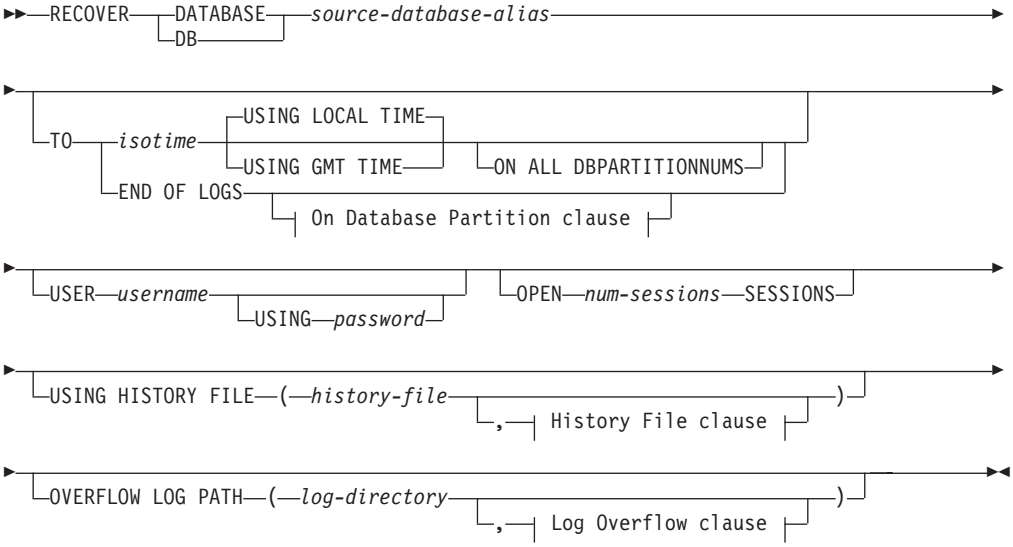
新規のデータベースにリカバリーするには、次のいずれかが必要です。

- *sysadm*
- *sysctrl*

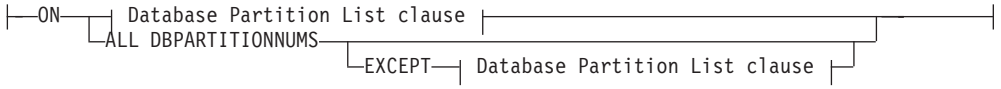
必要な接続:

既存のデータベースをリカバリーするには、データベース接続が必要です。このコマンドを呼び出せば、指定したデータベースへの接続が自動的に確立され、リカバリー操作が終了すると接続が解放されます。新しいデータベースにリカバリーするには、インスタンス接続とデータベース接続が必要です。データベースを作成するには、インスタンス・アタッチが必要です。

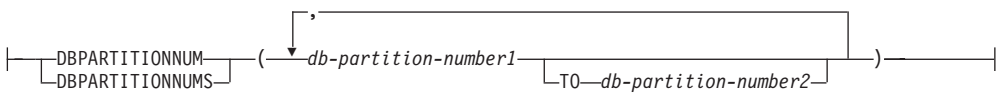
コマンド構文:



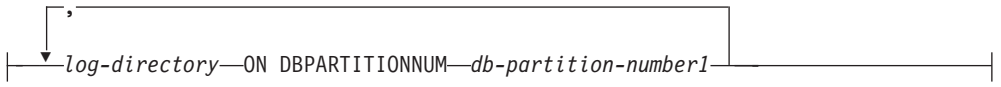
On Database Partition clause:



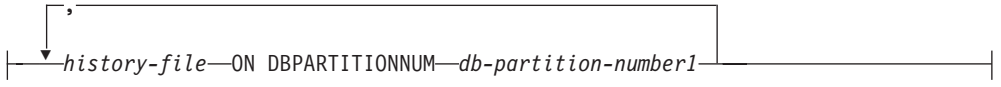
Database Partition List clause:



Log Overflow clause:



History File clause:



コマンド・パラメーター:

DATABASE *database-alias*
リカバリーするデータベースの別名。

USER *username*
データベースがリカバリーされる際のユーザー名。

USING *password*
ユーザー名を認証するために使用するパスワード。パスワードを省略すると、ユーザーに入力を求めるプロンプトが出ます。

TO

isotime コミットされたすべてのトランザクションがリカバリーされる時点 (その時点の前にコミットされたすべてのトランザクションのほかに、ちょうどその時点にコミットされたトランザクションを含む)。

この値は、日付と時刻の組み合わせを識別する 7 つの部分の文字ストリングからなる、タイム・スタンプとして指定します。形式は、*yyyy-mm-dd-hh.mm.ss.nnnnnn* (年、月、日、時、分、秒、マイクロ秒) の協定世界時 (UTC) です。 UTC は、さまざまなログに関連したタイム・スタンプが (たとえば、夏時間に関連して時間が変更されることによって) 同じにならないようにするのに役立ちます。バックアップ・イメージのタイム・スタンプは、バックアップ操作が開始した地方時に基づいています。 CURRENT TIMEZONE 特殊レジスターは、UTC とアプリケーション・サーバーの地方時との時差を指定します。時差は、時刻期間 (最初の 2 桁が時間数、次の 2 桁が分数、最後の 2 桁が秒数を表す 10 進数) で表されます。地方時から CURRENT TIMEZONE を減算すると、地方時を UTC に変換できます。

USING LOCAL TIME
リカバリーに指定するポイント・イン・タイム。このオプションを指定することにより、ユーザーは GMT 時間ではなくユーザーの現地時間を使用して特定の時点にリカバリーできます。これによって、ユーザーがローカル・マシンで特定の時点にリカバリーすることが容易になり、現地時間を GMT ポイント・イン・タイムに変換することによって生じる潜在的なユーザー・エラーの発生を除去します。

注:

1. ユーザーがリカバリーのために現地時間を指定した場合、ユーザーに戻されるすべてのメッセージも現地時間で表示されます。すべての時刻はサーバー上で変換され、パーティション・データベース環境の場合にはカタログ・データベース・パーティション上で変換されることに注意してください。
2. タイム・スタンプ・ストリングはサーバー上で GMT に変換されるので、時間はクライアントではなくサーバーの時間帯に基づく現地時間となります。クライアントの時間帯とサーバーの時間帯とが異なる場合、サーバーの現地時間を使用してください。これはコントロール・センターの現地時間オプションとは異なります。そのオプションは、クライアントの現地時間を使用します。
3. タイム・スタンプ・ストリングが夏時間調整のための時間変更に接近している場合、停止時刻が時間変更の前か後かを判別して、それを適切に指定することが大切です。

USING GMT TIME
リカバリーに指定するポイント・イン・タイム。

END OF LOGS
データベースの構成パラメーターである *logpath* にリストされる、すべてのオンライン・アーカイブ・ログ・ファイルからコミットされた全トランザクションが、適用されることを指定します。

ON ALL DBPARTITIONNUMS
db2nodes.cfg ファイルで指定されている、すべてのパーティションについてトランザクションがロールフォワードされることを指定します。データベース・パーティション文節が指定されていない場合、これがデフォルトです。

EXCEPT
パーティション・リストに指定されているパーティションを除き、db2nodes.cfg ファイルで指定されている、すべてのパーティションについてトランザクションがロールフォワードされることを指定します。

ON DBPARTITIONNUM / ON DBPARTITIONNUMS
一連のデータベース・パーティションについてデータベースをロールフォワードします。

db-partition-number1
データベース・パーティション・リスト内のデータベース・パーティション番号を指定します。

db-partition-number2
2 番目のデータベース・パーティション番号を指定して、*db-partition-number1* から *db-partition-number2* までのすべてのパーティションがデータベース・パーティション・リストに含まれるようにします。

OPEN *num-sessions* **SESSIONS**
Tivoli Storage Manager (TSM) またはベンダー製品とともに使用する入出力セッションの数を指定します。

USING HISTORY FILE *history-file*

history-file **ON DBPARTITIONNUM**
パーティション・データベース環境で、異なる履歴ファイルを可能にします。

OVERFLOW LOG PATH *log-directory*
リカバリー処理中にアーカイブされたログを探索する代替のログ・パスを指定します。ログ・ファイルが *logpath* データベース構成パラメーターで指定されるロケーション以外のロケーションに移動された場合に、このパラメーターを使用します。パーティション・データベース環境では、これはすべてのパーティションの (完全修飾) デフォルト・オーバーフロー・ログ・パスです。単一パーティション・データベースの場合は、相対オーバーフロー・ログ・パスを指定できます。

注: OVERFLOW LOG PATH コマンド・パラメーターは、データベース構成パラメーター *overflowlogpath* の値 (存在する場合) を上書きします。

log-directory **ON DBPARTITIONNUM**
パーティション・データベース環境では、特定のパーティションのデフォルト・オーバーフロー・ログ・パスを別のログ・パスでオーバーライドできます。

例:

単一パーティション・データベース環境で、リカバリーするデータベースが現在存在しており、履歴ファイルの最新バージョンが *dftdbpath* で利用可能である場合、

1. 最新のバックアップ・イメージを使用し、すべてデフォルト値を使用してログの終わりまでロールフォワードするには、次のようにします。

   ```
   RECOVER DB SAMPLE
   ```

2. データベースをある PIT (時刻ポイント) までリカバリーするには、次のコマンドを発行します。指定された PIT に達するまで、使用可能な最新のイメージがリストアされ、ログが適用されます。

   ```
   RECOVER DB SAMPLE TO  2001-12-31-04:00:00
   ```

3. 履歴ファイルの保管バージョンを使用してデータベースをリカバリーするには、次のコマンドを発行します。たとえば、現在の履歴ファイルに含まれていないような非常に古い PIT までリカバリーすることが必要な場合、ユーザーはその時刻からの履歴ファイルのバージョンを提供できます。ユーザーがその時刻からの履歴ファイルを保管している場合、そのバージョンを使用してリカバリーを実行できます。

   ```
   RECOVER DB SAMPLE TO  1999-12-31-04:00:00
      USING HISTORY FILE (/home/user/old1999files/db2rhist.asc)
   ```

単一パーティション・データベース環境で、リカバリーするデータベースが存在していない場合は、 USING HISTORY FILE 文節を使用して履歴ファイルを指定する必要があります。

1. その履歴ファイルのバックアップを作成していないため、バックアップ・イメージ内のコピーしか利用できない場合には、 ROLLFORWARD の後に RESTORE を発行することをお勧めします。しかし、RECOVER を使用するためには、履歴ファイルをイメージからどこかの場所 (たとえば /home/user/oldfiles/db2rhist.asc) に抽出してから、このコマンドを発行する必要があります。 (履歴ファイルのそのバージョンには、ロールフォワードに必要なログ・ファイルに関する情報が含まれていないため、その履歴ファイルは RECOVER には適していません。)

   ```
   RECOVER DB SAMPLE TO END OF LOGS
      USING HISTORY FILE (/home/user/fromimage/db2rhist.asc)
   ```

2. 履歴のバックアップ・コピーを周期的に、あるいは頻繁に作成している場合は、USING HISTORY 文節を使用して、履歴ファイルのこのバージョンを指定するようにしてください。ファイルが /home/user/myfiles/db2rhist.asc なら、発行するコマンドは次のようになります。

   ```
   RECOVER DB SAMPLE TO PIT
      USING HISTORY FILE (/home/user/myfiles/db2rhist.asc)
   ```

   (この場合、要求されたポイント・イン・タイム (PIT) より前に取られたバックアップが含まれている限り、履歴ファイルの任意のコピーを使用できます。最新のものである必要はありません。)

パーティション・データベース環境で、データベースがすべてのデータベース・パーティション上に存在し、すべてのデータベース・パーティション上の *dftdbpath* に最新の利用可能な履歴ファイルが存在する場合、次のとおりです。

1. すべてのノードでデータベースを特定の PIT までリカバリーする場合。 DB2 は、リストア操作を開始する前に、その PIT がどのノードでも到達可能であるかどうかを検証します。

   ```
   RECOVER DB SAMPLE TO  2001:12:31:04:00:00
   ```

2. すべてのノードでデータベースをこの PIT までリカバリーする場合。 DB2 は、リストア操作を開始する前に、その PIT がどのノードでも到達可能であるかどうかを検証します。各ノードでの RECOVER 操作は、単一パーティション RECOVER の場合と同一です。

   ```
   RECOVER DB SAMPLE TO END OF LOGS
   ```

3. *dftdbpath* に履歴ファイルの最新バージョンがありますが、複数の特定の履歴ファイルを使用したい場合があるかもしれません。特に指定しない場合、各パーティションでは /home/user/oldfiles/db2rhist.asc にローカルに存在する履歴ファイルが使用されます。例外はノード 2 と 4 です。ノード 2 では /home/user/node2files/db2rhist.asc が使用され、ノード 4 では /home/user/node4files/db2rhist.asc が使用されます。

   ```
   RECOVER DB SAMPLE TO  1999:12:31:04:00:00
      USING HISTORY FILE (/home/user/oldfiles/db2rhist.asc,
         /home/user/node2files/db2rhist.asc ON DBPARTITIONNUM 2,
         /home/user/node4files/db2rhist.asc ON DBPARTITIONNUM 4)
   ```

4. すべてのノードではなく一部のノードをリカバリーすることも可能です。しかし、その場合は PIT RECOVER は実行できず、リカバリーは EOL まで実行する必要があります。

   ```
   RECOVER DB SAMPLE TO END OF LOGS ON DBPARTITIONNUMS(2 TO 4, 7, 9)
   ```

パーティション・データベース環境で、データベースが存在しない場合、次のとおりです。

1. その履歴ファイルのバックアップを作成していないため、バックアップ・イメージ内のコピーしか利用できない場合には、 ROLLFORWARD の後に RESTORE を発行することをお勧めします。しかし、RECOVER を使用するためには、履歴ファイルをイメージからどこかの場所 (たとえば /home/user/oldfiles/db2rhist.asc) に抽出してから、このコマンドを発行する必要があります。 (履歴ファイルのそのバージョンには、ロールフォワードに必要なログ・ファイルに関する情報が含まれていないため、その履歴ファイルはリカバリーには適していません。)

   ```
   RECOVER DB SAMPLE TO PIT
      USING HISTORY FILE (/home/user/fromimage/db2rhist.asc)
   ```

2. 履歴のバックアップ・コピーを周期的に、あるいは頻繁に作成している場合は、USING HISTORY 文節を使用して、履歴ファイルのこのバージョンを指定するようにしてください。ファイルが /home/user/myfiles/db2rhist.asc なら、次のコマンドを発行できます。

   ```
   RECOVER DB SAMPLE TO END OF LOGS
      USING HISTORY FILE (/home/user/myfiles/db2rhist.asc)
   ```

使用上の注意:

- データベースをリカバリーするためには、磁気テープ装置を使用したロード・リカバリーが必要になる場合があります。別のテープを求める要求が出された場合は、次のいずれか 1 つで応答できます。

  c 続行。警告メッセージを生成した装置の使用を続けます (たとえば、新しいテープをマウントしたときなど)。

  d 装置の終了。警告メッセージを生成した装置の使用を停止します (たとえば、それ以上テープがない場合)。

  t 終了。すべての装置を終了します。
- リカバリー操作のリストア部分において障害が発生した場合は、 RECOVER DATABASE コマンドを再発行することができます。リストア操作が成功したが、ロールフォワード操作中にエラーが発生した場合は、リカバリー操作全体をやり直す必要はなく (それには非常に時間がかかる)、 ROLLFORWARD DATABASE コマンドを発行することができます。
- パーティション・データベース環境で、リカバリー操作のリストア部分でエラーが発生した場合、単一データベース・パーティションでのエラーでしかないという可能性があります。 RECOVER DATABASE コマンドを再発行するとすべてのデータベース・パーティションでデータベースがリストアされますが、その代わりに、障害が発生したデータベース・パーティションに関する RESTORE DATABASE を発行してから、 ROLLFORWARD DATABASE コマンドを発行するほうが効率的です。

db2Recover - データベースのリカバリー

データベースを、特定のポイント・イン・タイムまで、またはログの終わりまでリストアおよびロールフォワードします。

有効範囲:

パーティション・データベース環境では、この API はカタログ・パーティションからのみ呼び出すことができます。データベース・パーティション・サーバーが 1 つも指定されていない場合には、 `db2nodes.cfg` ファイルにリストされているすべてのデータベース・パーティション・サーバーに影響を与えます。特定のポイント・イン・タイムが指定される場合、この API はすべてのデータベース・パーティションに影響します。

権限:

既存のデータベースにリカバリーするには、次のいずれかが必要です。

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

新規のデータベースにリカバリーするには、次のいずれかが必要です。

- *sysadm*
- *sysctrl*

必要な接続:

既存のデータベースをリカバリーするには、データベース接続が必要です。この API を呼び出せば、指定したデータベースへの接続が自動的に確立され、リカバリー操作が終了すると接続が解放されます。新規のデータベースにリカバリーする場合、インスタンスおよびデータベース。データベースを作成するには、インスタンス・アタッチが必要です。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2Recover */
/* ... */
SQL_API_RC SQL_API_FN
  db2Recover (
       db2Uint32 versionNumber,
       void * pDB2RecovStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RecoverStruct
{
  char                        *piSourceDBAlias;
  char                        *piUsername;
  char                        *piPassword;
  db2Uint32                   iRecoverCallerAction;
   db2Uint32           iOptions;
  sqlint32                    *poNumReplies;
  struct sqlurf_info          *poNodeInfo;
  char                        *piStopTime;
  char                        *piOverflowLogPath;
  db2Uint32                   iNumChngLgOvrflw;
  struct sqlurf_newlogpath    *piChngLogOvrflw;
  db2int32                    iAllNodeFlag;
  db2int32                    iNumNodes;
  SQL_PDB_NODE_TYPE           *piNodeList;
  db2int32                    iNumNodeInfo;
  db2Uint32                   iRollforwardFlags;
  char                        *piHistoryFile;
  db2Uint32                   iNumChngHistoryFile;
  struct sqlu_histFile        *piChngHistoryFile;
} db2RecoverStruct;
/* ... */
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2gRecover */
/* ... */
SQL_API_RC SQL_API_FN
  db2gRecover (
       db2Uint32 versionNumber,
       void * pDB2gRecoverStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRecoverStruct
{
  char                        *piSourceDBAlias;
  db2Uint32                   iSourceDBAliasLen;
  char                        *piUserName;
  db2Uint32                   iUserNameLen;
  char                        *piPassword;
  db2Uint32                   iPasswordLen;
  db2Uint32                   iRecoverCallerAction;
   db2Uint32           iOptions;
```

```
  sqlint32                    *poNumReplies;
  struct sqlurf_info          *poNodeInfo;
  char                        *piStopTime;
  db2Uint32                   iStopTimeLen;
  char                        *piOverflowLogPath;
  db2Uint32                   iOverflowLogPathLen;
  db2Uint32                   iNumChngLgOvrflw;
  struct sqlurf_newlogpath    *piChngLogOvrflw;
  db2int32                    iAllNodeFlag;
  db2int32                    iNumNodes;
  SQL_PDB_NODE_TYPE           *piNodeList;
  db2int32                    iNumNodeInfo;
  db2Uint32                   iRollforwardFlags;
  char                        *piHistoryFile;
  db2Uint32                   iHistoryFileLen;
  db2Uint32                   iNumChngHistoryFile;
  struct sqlu_histFile        *piChngHistoryFile;
} db2gRecoverStruct;
/* ... */
```

API パラメーター:

versionNumber
入力。 2 番目のパラメーター *pDB2RecoverStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2RecoverStruct
入力。 *db2RecoverStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

piSourceDBAlias
入力。リカバリーするデータベースのデータベース別名を含むストリング。

iSourceDBAliasLen
piSourceDBAlias の長さ (バイト単位)。

piUserName
入力。接続の試行時に使用されるユーザー名を含むストリングを指定します。 NULL にすることもできます。

iUserNameLen
piUsername の長さ (バイト単位)。

piPassword
入力。ユーザー名とともに使用されるパスワードを含むストリングを指定します。 NULL にすることもできます。

iPasswordLen
piPassword の長さ (バイト単位)。

iRecoverCallerAction
入力。有効な値は以下のとおりです。

DB2RESTORE_NOINTERRUPT
リストア操作を開始します。リストアを自動実行するよう指定します。通常ユーザーの介入を要求するシナリオは、呼び出し側への最初の戻りなしに試行されるか、エラーを生成します。この呼び出し側アクションは、たとえば、リストアに必要なメディアがすべてマ

ウントされていることが明らかで、ユーティリティーによるプロンプトが必要とされない場合に使用してください。

DB2RESTORE_CONTINUE
ユーザーがユーティリティーによって要求された何らかのアクション (たとえば、新しいテープのマウント) を実行した後で、リストア操作を継続します。

DB2RESTORE_TERMINATE
ユーザーがユーティリティーによって要求された何らかのアクションの実行に失敗した場合、リストア操作を終了します。

DB2RESTORE_DEVICE_TERMINATE
リストア操作によって使用される装置のリストから特定の装置を除外します。特定の装置が入力でいっぱいになると、リストア・ユーティリティーは呼び出し側に警告を戻します。その場合、この呼び出し側アクションを指定してリストア操作を再び呼び出すことによって、警告が生成される原因となった装置を使用装置のリストから除外してください。

DB2RESTORE_PARM_CHK
リストア操作を実行することなく、パラメーターの妥当性を検査します。このオプションは、呼び出しが戻った後でデータベース接続を終了しません。この呼び出しが正常に戻った後、ユーザーが DB2RESTORE_CONTINUE で呼び出しを発行し、処置を進めることが期待されます。

DB2RESTORE_PARM_CHK_ONLY
リストア操作を実行することなく、パラメーターの妥当性を検査します。この呼び出しが戻る前に、この呼び出しによって確立したデータベース接続は終了し、後続する呼び出しは必要なくなります。

DB2RESTORE_TERMINATE_INCRE
完了する前に増分リストア操作を終了します。

DB2ROLLFORWARD_LOADREC_CONT
警告メッセージを生成した装置の使用を続けます (たとえば、新しいテープをマウントしたときなど)。

DB2ROLLFORWARD_DEVICE_TERM
警告メッセージを生成した装置の使用を停止します (たとえば、それ以上テープがない場合)。

DB2ROLLFORWARD_LOAD_REC_TERM
ロード・リカバリーに使用されているすべての装置を終了させます。

iOptions
入力。有効な値は以下のとおりです。

DB2RECOVER_EMPTY_FLAG
フラグが指定されていません。

DB2RECOVER_LOCAL_TIME
piStopTime によって停止時刻として指定された値が、 GMT ではなくローカル時間であることを示します。これはデフォルト設定です。

DB2RECOVER_GMT_TIME
このフラグは、*piStopTime* によって停止時刻として指定された値が、 GMT (グリニッジ標準時) であることを示します。

poNumReplies
出力。受信した応答の数。

poNodeInfo
出力。データベース・パーティション応答情報。

piStopTime
入力。 ISO 形式のタイム・スタンプを含む文字ストリング。このタイム・スタンプで設定された時刻を過ぎると、データベース・リカバリーは停止します。可能な限りロールフォワードしたい場合には、SQLUM_INFINITY_TIMESTAMP を指定してください。DB2ROLLFORWARD_QUERY、DB2ROLLFORWARD_PARM_CHECK、およびいずれかのロード・リカバリー (DB2ROLLFORWARD_LOADREC_) 呼び出し側アクションの場合は、NULL にすることができます。

iStopTimeLen
piStopTime の長さ (バイト単位)。

piOverflowLogPath
入力。使用される代替ログ・パスを指定します。このユーティリティーを使用する前に、アクティブ・ログ・ファイルの他に、アーカイブ・ログ・ファイルを *logpath* 構成パラメーターで指定した場所に移動させる必要があります。このことは、ログ・パスに十分なスペースがない場合に問題になる可能性があります。その問題を解決するために、オーバーフロー・ログ・パスが備えられています。ロールフォワード・リカバリー中に、必要なログ・ファイルは、まずログ・パスで探索され、次にオーバーフロー・ログ・パスで探索されます。表スペースのロールフォワード・リカバリーに必要なログ・ファイルは、ログ・パスまたはオーバーフロー・ログ・パスのいずれかに置かれる可能性があります。呼び出し側がオーバーフロー・ログ・パスを指定しない場合、デフォルト値はログ・パスです。パーティション・データベース環境では、オーバーフロー・ログ・パスは有効な完全修飾パスでなければなりません。デフォルトのパスは、各データベース・パーティションのデフォルトのオーバーフロー・ログ・パスです。単一パーティション・データベース環境では、サーバーがローカルであれば、オーバーフロー・ログ・パスは相対パスにすることもできます。

iOverflowLogPathLen
piOverflowLogPath の長さ (バイト単位)。

iNumChngLgOvrflw
入力。パーティション・データベース環境のみ。変更されるオーバーフロー・ログ・パスの数。新しいログ・パスは、指定されたデータベース・パーティション・サーバーのデフォルトのオーバーフロー・ログ・パスだけをオーバーライドします。

piChngLogOvrflw
入力。パーティション・データベース環境のみ。変更されるオーバーフロー・ログ・パスの完全修飾名が入っている構造を指すポインター。新しいログ・パスは、指定されたデータベース・パーティション・サーバーのデフォルトのオーバーフロー・ログ・パスだけをオーバーライドします。

iAllNodeFlag
入力。パーティション・データベース環境のみ。ロールフォワード操作が、db2nodes.cfg で定義されているすべてのデータベース・パーティション・サーバーに適用されるかどうかを示します。有効な値は以下のとおりです。

DB2_NODE_LIST
piNodeList で渡されたリスト内のデータベース・パーティション・サーバーに適用されます。

DB2_ALL_NODES
すべてのデータベース・パーティション・サーバーに適用されます。 *piNodeList* は NULL でなければなりません。これはデフォルト値です。

DB2_ALL_EXCEPT
piNodeList で渡されたリスト内のデータベース・パーティション・サーバーを除いた、すべてのデータベース・パーティション・サーバーに適用されます。

DB2_CAT_NODE_ONLY
カタログ・パーティションにのみ適用されます。 *piNodeList* は NULL でなければなりません。

iNumNodes
入力。 *piNodeList* 配列内のデータベース・パーティション・サーバーの数を指定します。

piNodeList
入力。ロールフォワード操作を実行する対象のデータベース・パーティション・サーバー番号の配列を指すポインター。

iNumNodeInfo
入力。出力パラメーター *poNodeInfo* のサイズを定義します。これは、ロールフォワードされるそれぞれのデータベース・パーティションからの状況情報を保持するのに十分な大きさでなければなりません。単一パーティション・データベース環境では、このパラメーターは 1 に設定しなければなりません。このパラメーターの値は、この API が呼び出されるデータベース・パーティション・サーバーの数と同じにする必要があります。

RollforwardFlags
入力。ロールフォワード・フラグを指定します。有効な値 (db2ApiDf.h で定義) は、以下のとおりです。

DB2ROLLFORWARD_EMPTY_FLAG
フラグが指定されていません。

DB2ROLLFORWARD_LOCAL_TIME
ユーザーが GMT 時間ではなくユーザーの現地時間を使用して特定の時点にロールフォワードすることを可能にします。これによっ

て、ユーザーがローカル・マシンで特定の時点にロールフォワードすることが容易になり、現地時間を GMT ポイント・イン・タイムに変換することによって生じる潜在的なユーザー・エラーの発生を除去します。

piHistoryFile
履歴ファイル。

iHistoryFileLen
piHistoryFile の長さ (バイト単位)。

iNumChngHistoryFile
リスト内の履歴ファイルの数。

piChngHistoryFile
履歴ファイルのリスト。

使用上の注意:

関連資料:

- 169 ページの『RECOVER DATABASE』

第 2 部 高可用性

第 6 章 高可用性およびフェイルオーバー・サポートの紹介

e-business が成功するかどうかは、トランザクション処理システムを途切れることなく使用できるかどうかにかかっています。さらにそれは 1 日 24 時間、週 7 日 (「24 × 7」) 使用可能な、 DB2 などのデータベース管理システム主導型のものでなければなりません。このセクションでは、以下のことを説明します。

- 『高可用性』
- 189 ページの『オンライン・スプリット・ミラーおよびサスペンド入出力サポートによる高可用性』
- 195 ページの『UNIX ベースのシステム用の障害モニター機能』
- 197 ページの『db2fm - DB2 障害モニター』

高可用性

高可用性 (HA) とは、常時稼働していて、お客様にも常時利用可能なシステムのことを説明するのに使用される用語です。このためには以下の条件が必要です。

- トランザクションは、ピーク時の運用期間中に目に見える性能低下 (さらには利用不可になる) を起こさずに効率的に処理される必要があります。パーティション・データベース環境では、DB2® はパーティション内およびパーティション間並列処理の、両方の利点を利用してトランザクションを効率的に処理することができます。*パーティション内並列処理* は、SMP 環境で使用することができ、これにより複雑な SQL ステートメントを構成する一部分の SQL ステートメントを同時に処理することができます。一方、パーティション・データベース環境での*パーティション間並列処理* は、関与しているすべてのノード上の照会を同時に処理することを指します。このとき各ノードが表内の行のサブセットを処理します。
- システムは、ハードウェアまたはソフトウェアの障害が発生した場合、また災害時に即時にリカバリーできなければなりません。 DB2 には、極めて迅速なクラッシュ・リカバリーを可能にする、連続的なチェックポイント実行を行う拡張機能および並列リカバリー機能があります。

  即時にリカバリーできるかどうかは、十分にテストしたバックアップと回復の手順が確立されていることにも依存しています。
- エンタープライズ・データベースを動かすソフトウェアは、連続的に稼働し、トランザクション処理のために常時使用可能でなければなりません。データベース・マネージャーが稼働している状態を保つため、障害が起こった場合に別のデータベース・マネージャーが引き受けられるようにしておく必要があります。これはフェイルオーバーと呼ばれます。フェイルオーバー 機能を使用すると、ハードウェアの故障があった場合に、あるシステムから別のシステムにトランザクション処理を自動的に移すことが可能になります。

高可用性災害時リカバリー (HADR) データベース複製機能を使用すると、フェイルオーバー機能をインプリメントすることができます。 HADR は、変更内容を、1 次データベースと呼ばれるソース・データベースから、スタンバイ・データベース

と呼ばれるターゲット・データベースへ複製することにより、データ損失を防ぐことができる高可用性ソリューションです。

フェイルオーバー機能は、別マシンにデータベースのコピーを保持することでも実現できます。そのコピーのデータベースは、1 次データベースのログ・ファイルを絶えずロールフォワードすることにより維持されます。そのために、*ログ・シッピング* と呼ばれる処理により、 1 次データベースのログ・ファイルをスタンバイ・マシンにコピーします。これは、1 次データベースにおいて、アーカイブ・ログを保持しているストレージ装置から直接コピー、または、アーカイブ・ログをコピーするユーザー出口プログラムを利用して行います。この方法では、まずはじめに、スタンバイ・マシンに 1 次データベースのコピーを作成する必要がありますが、それは、1 次データベースのバックアップをリストア・ユーティリティーでリストアする、または、スプリット・ミラー機能 (これ自体はストレージ製品の機能) を利用してリストアすることで行います。また、スプリット・ミラー機能とサスペンド入出力サポートを使用して、コピーを即時に初期化して新規データベースとして利用することも可能です。こうして、スタンバイ・マシンにリストアされたスタンバイ・データベースは、 1 次マシンからコピーされるログを使用して、ロールフォワード処理を連続して実行します。 1 次データベースに障害が起こった場合、残っているログ・ファイルはすべてスタンバイ・マシンにコピーして移されます。 to end of logs and stop のオプションを指定した ROLLFORWARD 実行後、すべてのクライアントはスタンバイ・マシン上のスタンバイ・データベースに再接続されます。

フェイルオーバー・サポートは、使用しているシステムに追加できる、プラットフォーム固有のソフトウェアによって提供されることもあります。たとえば、

- Tivoli® System Automation for Linux。

  Tivoli System Automation についての詳細は、「Highly Available DB2 Universal Database™ using Tivoli System Automation for Linux」というタイトルの白書を参照してください。これは、「DB2 UDB and DB2 Connect™ Online Support」Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できます。

- High Availability Cluster Multi-Processing, Enhanced Scalability, for AIX®。

  HACMP/ES についての詳細は、「IBM® DB2 Universal Database Enterprise Edition for AIX and HACMP/ES」というタイトルの白書を参照してください。これは、「DB2 UDB and DB2 Connect Online Support」Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できます。

- Microsoft® Cluster Server (Windows® オペレーティング・システム版)。

  Microsoft Cluster Server については、「DB2 UDB and DB2 Connect Online Support」Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できる以下の白書を参照してください。「Implementing IBM DB2 Universal Database V8.1 Enterprise Server Edition with Microsoft Cluster Server」。

- Sun™ Cluster、または VERITAS Cluster Server (Solaris™ オペレーティング環境版)。

  Sun Cluster についての情報は、「DB2 Universal Database and High Availability on Sun Cluster 3.X」というタイトルの白書を参照してください。これは 「DB2 UDB and DB2 Connect Online Support」web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できます。 VERITAS

Cluster Server についての詳細は、「DB2 UDB and High Availability with VERITAS Cluster Server」というタイトルの白書を参照してください。これは、「IBM Support and downloads」Web サイト (http://www.ibm.com/support/docview.wss?uid=swg21045033) から入手できます。

- Multi-Computer/ServiceGuard (Hewlett-Packard 用)。

  HP MC/ServiceGuard についての詳細は、HP MC/ServiceGuard 高可用性ソフトウェアを用いた IBM DB2 ESE V8.1 を説明している白書を参照してください。これは、「IBM DB2 Information Management Products for HP」Web サイト (http://www-306.ibm.com/software/data/hp/) から入手できます。

フェイルオーバー・ストラテジーは通常、システム・クラスター構成を基本とします。*クラスター* とは、単一システムとして協調して稼働する、接続された複数のシステムのグループのことです。 1 つのクラスター内の各物理マシンには、1 つ以上の論理ノードが含まれます。クラスタリングにより、障害が発生したときには、障害の発生したサーバーの処理を引き継ぐことによってサーバーが互いにバックアップし合うことができます。

IP アドレス・テークオーバー (または IP テークオーバー) とは、あるサーバーが使用不可になった場合にサーバー IP アドレスを 1 つのマシンから別のマシンに移すことができるということです。つまり、IP テークオーバー前と後では、クライアント・アプリケーションは同じサーバーにアクセスしているつもりでも、実際には物理的に異なるマシンにアクセスすることになります。

フェイルオーバー・ソフトウェアは、*ハートビート監視* またはキープアライブ・パケット をシステム間で使用して、クラスター内のシステムが稼働していることを確認する場合もあります。ハートビート監視は、クラスター内のすべてのノード間で定常的に通信し続けるシステム・サービスを含んでいます。ハートビートが検出されない場合、バックアップ・システムへのフェイルオーバーが開始します。エンド・ユーザーは通常、システムに障害が起こったことに気付きません。

現在使用されている最も一般的な 2 つのフェイルオーバー・ストラテジーは*アイドル・スタンバイ* および*相互テークオーバー* として知られています。ただし、これらの用語で述べられる構成は、ベンダーによって異なる用語で呼ばれていることがあります。

アイドル・スタンバイ
この構成では、一方のシステムが使用されて DB2 インスタンスを実行します。 2 番目のシステムは「アイドル」、あるいは、スタンバイ・モードになっており、最初のシステムにかかわるオペレーティング・システムまたはハードウェアの故障があったときに、インスタンスを引き受ける準備をしています。スタンバイ・システムは必要になるまで使用されないため、全体のシステム・パフォーマンスは影響を受けません。

相互テークオーバー
この構成では、各システムがそれぞれ別のシステムの指定されたバックアップになっています。全体のシステム・パフォーマンスに影響があることもあります。これは、バックアップ・システムがフェイルオーバー後に追加の処理を行わなければならない、つまりそのシステム自身の処理と障害の発生したシステムの処理を行わなければならないためです。

フェイルオーバー・ストラテジーはインスタンス、パーティション、または複数の論理ノードをフェイルオーバーするのに使用することができます。

関連概念:

- 「*管理ガイド: プランニング*」の『並列処理』
- 3 ページの『バックアップおよびリカバリー計画の作成』
- 189 ページの『オンライン・スプリット・ミラーおよびサスペンド入出力サポートによる高可用性』
- 255 ページの『Solaris オペレーティング環境でのクラスター・サポート』
- 258 ページの『Sun Cluster 3.0 のサポート』
- 261 ページの『VERITAS Cluster Server のサポート』
- 249 ページの『Microsoft Cluster Server のサポート』
- 243 ページの『High Availability Cluster Multi-Processing のサポート』
- 201 ページの『高可用性災害時リカバリーの概要』

ログ・シッピングによる高可用性

ログ・シッピングとは、すべてのログ・ファイルをスタンバイ・マシンにコピーする処理のことです。これは、1 次データベースにおいて、アーカイブ・ログを保持しているストレージ装置から直接コピー、または、アーカイブ・ログをコピーするユーザー出口プログラムを利用して行います。スタンバイ・データベースは、稼働マシンにより作成されたログ・ファイルを連続的にロールフォワードしています。稼働マシンに障害が起こった時、フェイルオーバーが発生し、以下のことがなされます。

- 残ったログがスタンバイ・マシンに転送される。
- スタンバイ・データベースにおいて to end of logs and stop のオプションを指定した ROLLFORWARD が実行される。
- クライアントがスタンバイ・データベースに再接続し、操作を再開する。

スタンバイ・マシンには、それ自体のリソース (すなわち、ディスク) がありますが、稼働データベースと同じ物理的および論理的定義を持っている必要があります。この方法を用いる時に、1 次データベースは、リストア・ユーティリティーまたはスプリット・ミラー機能を使用してスタンバイ・マシンにリストアされます。

災害時リカバリーの状況でデータベースを確実にリカバリーできるようにするために、以下の点を考慮してください。

- アーカイブの場所は、基本サイトとは地理的に分離しているべきである。
- 1 次データベースのログをスタンバイ・データベース・サイトへ遠隔ミラーする。
- データ損失ゼロとするためには、同期ミラーを使用する。このことは、最近のディスク・サブシステム (ESS や EMC など) や、別のリモート・ミラーリング・テクノロジーを使用して行えます。 NVRAM キャッシュ (ローカルとリモートの両方) も、災害時リカバリーのパフォーマンスへの影響を最小化するためにお勧めします。

注:

1. 索引の再作成が 1 次データベースで行われたことを示すログ・レコードをスタンバイ・データベースが処理する時、スタンバイ・サーバー上の索引は自動的に再作成されません。データベースへの最初の接続時、またはスタンバイ・サーバーがロールフォワード・ペンディング状態から解放された後に初めて索引にアクセスしようとした時に、索引はスタンバイ・サーバーに再作成されます。 1 次サーバーのいずれかの索引が再作成された場合には、スタンバイ・サーバーが 1 次サーバーと再同期を取ることをお勧めします。
2. ロード・ユーティリティーが、COPY YES オプションを指定した状態で 1 次データベース上で実行される場合、スタンバイ・データベースは、ロールフォワード時に、そのコピー・イメージへアクセスできる必要があります。
3. ロード・ユーティリティーが、COPY NO オプションを指定した状態で 1 次データベース上で実行される場合、スタンバイ・データベースは再同期を取る必要があります。再同期しない場合、表スペースはリストア・ペンディング状態に置かれます。
4. スタンバイ・マシンを初期化するには、2 つの方法があります。
   a. バックアップ・イメージからそこにリストアすることにより。
   b. 実動システムのスプリット・ミラーを作成し、 STANDBY オプションで **db2inidb** コマンドを実行することにより。

   スタンバイ・マシンを初期化した後にのみ ROLLFORWARD コマンドをスタンバイ・システム上で実行することができます。
5. ログに記録されていない操作は、スタンバイ・データベースでは再実行できません。結果として、そうした操作をした後には、スタンバイ・データベースを再同期することをお勧めします。このことは、オンライン・スプリット・ミラーおよびサスペンド入出力サポートを通して行えます。

関連概念:

- 189 ページの『オンライン・スプリット・ミラーおよびサスペンド入出力サポートによる高可用性』
- 201 ページの『高可用性災害時リカバリーの概要』

関連タスク:

- 192 ページの『スプリット・ミラーをスタンバイ・データベースとして使用する』

関連資料:

- 375 ページの『付録 G. データベース・リカバリー用のユーザー出口』

オンライン・スプリット・ミラーおよびサスペンド入出力サポートによる高可用性

サスペンド入出力 は、オンライン・スプリット・ミラー処理 (つまり、データベースをシャットダウンすることなくミラーを分割する) の完全インプリメンテーションを提供することにより、システムの連続可能性を実現しています。*スプリット・ミラー* はデータを含むディスクをミラーリングしていて、コピーが必要なときにミラーを分割して作成できるデータベースの「瞬間」コピーです。*ディスク・ミラーリング* は、すべてのデータを別個の 2 つのハード・ディスクに書き込む処理で

す。一方がもう一方のミラーになっています。ミラーを*分割する* とは、データベースの 1 次コピーと 2 次コピーを分離する処理のことです。

大きなデータベースは DB2® バックアップ・ユーティリティーを使用してバックアップしないという場合、サスペンド入出力およびスプリット・ミラー機能を使用してミラー・イメージからコピーを作成することができます。この方法では以下の利点もあります。

- 稼働マシンからバックアップ操作のオーバーヘッドを除去します。
- 高速にシステムを複製します。
- アイドル・スタンバイ・フェイルオーバーを高速にインプリメントできます。初期のリストア操作は不要です。また、ロールフォワードが遅すぎたりエラーが発生する場合に、再初期化を高速に実行してこれらに対応することが可能です。

db2inidb コマンドは、スプリット・ミラーを初期化して以下のように使用できるようにします。

- クローン・データベースとして
- スタンバイ・データベースとして
- バックアップ・イメージとして

このコマンドは、スプリット・ミラーに対してのみ発行することができます。そのコマンドは、スプリット・ミラーを使用する前に実行しなければなりません。

パーティション・データベース環境では、入出力を中断してすべてのパーティションに同時に書き込む必要はありません。 1 つ以上のパーティションのサブセットを中断して、オフライン・バックアップを実行するためにスプリット・ミラーを作成できます。カタログ・パーティションがサブセットに含まれる場合は、それを最後に中断する必要があります。

パーティション・データベース環境では、**db2inidb** コマンドは各パーティション上でそれぞれ実行する必要があります。それから、それらのパーティションのスプリット・イメージを使用することができます。 db2inidb コマンドは、**db2_all** コマンドを使用してすべてのパーティションで同時に実行することができます。しかし、RELOCATE USING オプションを使用する場合は、 **db2_all** コマンドを使用して全パーティションに対して同時に **db2inidb** を実行することはできません。パーティションごとにそれぞれ別個の構成ファイル (変更するパーティションの NODENUM 値が含まれる) を用意する必要があります。たとえば、データベースの名前を変更する場合は、すべてのパーティションが影響を受けることになり、各パーティションごとに別個の構成ファイルを用意して **db2relocatedb** コマンドを実行する必要があります。単一データベース・パーティションに属するコンテナーを移動する場合は、そのパーティション上で一度だけ **db2relocatedb** コマンドを実行することが必要です。

注: スプリット・ミラーが、データベースを構成するすべてのコンテナーおよびディレクトリー (ボリューム・ディレクトリーを含む) を含んでいることを確認してください。

関連資料:

- 289 ページの『db2inidb - ミラー・データベースの初期化』

オンライン・スプリット・ミラー処理

スプリット・ミラーを使用したデータベースのクローン作成

クローン・データベースを作成するには、以下の手順に従ってください。クローン・データベースに書き込むことはできますが、一般的に、レポート作成など、読み取り専用の処理で使用されます。

制限:

クローン・データベースをバックアップして、そのバックアップ・イメージを元のシステムにリストアしたり、また元のシステムで作成したログ・ファイルを使用してロールフォワードしたりすることはできません。

手順:

データベースのクローンを作成するには、以下のステップに従ってください。

1. 1 次データベース上で入出力をサスペンドします。

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

2. 該当するコマンドを使用して、 1 次データベースからミラーを分割します。

   注: ボリューム・ディレクトリーを含め、データベース・ディレクトリー全体をコピーするようにしてください。さらに、データベース・ディレクトリー外にある、ログ・ディレクトリーおよびコンテナー・ディレクトリーもコピーする必要があります。

3. 1 次データベース上で入出力を再開します。

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

4. 2 次システムのミラー・データベースをカタログします。

   注: デフォルトでは、ミラー・データベースは、 1 次データベースと同じシステムに存在できません。これは、同じディレクトリー構造を持ち、1 次データベースと同じインスタンス名を使用する、 2 次システム上に置く必要があります。ミラー・データベースを、1 次データベースと同じシステムに置かなければならない場合、**db2relocatedb** ユーティリティーか、 **db2inidb** コマンドの RELOCATE USING オプションを使用して、このことを実現できます。

5. 2 次システムでデータベース・インスタンスを開始します。

   ```
   db2start
   ```

6. 2 次システムでミラー・データベースを初期化します。

   ```
   db2inidb database_alias as snapshot
   ```

   必要であれば、db2inidb コマンドの RELOCATE USING オプションを指定して、クローン・データベースを再配置します。

   ```
   db2inidb database_alias as snapshot relocate using relocatedbcfg.txt
   ```

   ここで、relocatedbcfg.txt ファイルには、データベースを再配置するのに必要な情報が示されています。

注:

a. このコマンドにより、分割時に未了であったトランザクションがロールバックされ、新規ログ・チェーン・シーケンスが開始されます。そのため、1 次データベースからのいずれのログもクローン・データベース上で適用すること (ロールフォワード・リカバリーで用いること) はできません。

b. RELOCATE USING オプションの使用前に、データベース・ディレクトリー (ボリューム・ディレクトリーを含む)、ログ・ディレクトリー、およびコンテナー・ディレクトリーを再配置したいところに移動する必要があります。

関連概念:

- 189 ページの『オンライン・スプリット・ミラーおよびサスペンド入出力サポートによる高可用性』

関連タスク:

- 192 ページの『スプリット・ミラーをスタンバイ・データベースとして使用する』
- 194 ページの『スプリット・ミラーをバックアップ・イメージとして使用する』

関連資料:

- 「コマンド・リファレンス」の『db2relocatedb - データベースの再配置コマンド』
- 「コマンド・リファレンス」の『SET WRITE コマンド』

スプリット・ミラーをスタンバイ・データベースとして使用する

スタンバイ・データベースとして使用するために、データベースのスプリット・ミラーを作成するには、以下の手順に従ってください。 1 次データベースで障害が発生して、クラッシュ・リカバリーが必要な場合、スタンバイ・データベースを使用して、1 次データベースを引き継ぐことができます。

手順:

スプリット・ミラーをスタンバイ・データベースとして使用するには、以下のステップに従ってください。

1. 1 次データベース上で入出力をサスペンドします。

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

2. 該当するコマンドを使用して、 1 次データベースからミラーを分割します。

   注: ボリューム・ディレクトリーを含め、データベース・ディレクトリー全体をコピーするようにしてください。さらに、データベース・ディレクトリー外にある、ログ・ディレクトリーおよびコンテナー・ディレクトリーもコピーする必要があります。

3. 1 次データベース上で入出力を再開します。

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

4. 2 次システムのミラー・データベースをカタログします。

   注: デフォルトでは、ミラー・データベースは、 1 次データベースと同じシステムに存在できません。これは、同じディレクトリー構造を持ち、1 次データベースと同じインスタンス名を使用する、 2 次システム上に置く必要

があります。ミラー・データベースを、1 次データベースと同じシステムに置かなければならない場合、 **db2relocatedb** ユーティリティーか、**db2inidb** コマンドの RELOCATE USING オプションを使用して、このことを実現できます。

5. 2 次システムでデータベース・インスタンスを開始します。

   ```
   db2start
   ```

6. 2 次システムでミラー・データベースを初期化して、ロールフォワード・ペンディング状態にします。

   ```
   db2inidb database_alias as standby
   ```

   必要であれば、db2inidb コマンドの RELOCATE USING オプションを指定して、スタンバイ・データベースを再配置します。

   ```
   db2inidb database_alias as standby relocate using relocatedbcfg.txt
   ```

   ここで、relocatedbcfg.txt ファイルには、データベースを再配置するのに必要な情報が示されています。

   注:

   a. DMS 表スペース (データベース管理スペース) だけがある場合、2 次システム側で、DB2 のバックアップ・コマンドにより、フル・データベース・バックアップを取得することが可能です。これにより、稼働データベース側には、バックアップ取得負荷をかけないようにすることが可能です。

   b. RELOCATE USING オプションの使用前に、データベース・ディレクトリー (ボリューム・ディレクトリーを含む)、ログ・ディレクトリー、およびコンテナー・ディレクトリーを再配置したいところに移動する必要があります。

7. 1 次システムからログ・ファイルを取得できるように、ユーザー出口プログラムをセットアップします。
8. データベースを、to end of logs オプションで、あるいは、to point-in-time にてポイント・イン・タイム指定で、ロールフォワードします。
9. ログの最後、あるいは、指定されたポイント・イン・タイムに達するまで、ログの取得およびロールフォワードを実行し続けます。
10. スタンバイ・データベースをオンラインにするには、ROLLFORWARD コマンドを STOP オプションを指定して実行します。

    注: 1 次データベースのログは、ミラー・データベースのロールフォワード・ペンディング状態解除後には、そのミラー・データベースに適用できません。

関連概念:

- 189 ページの『オンライン・スプリット・ミラーおよびサスペンド入出力サポートによる高可用性』

関連タスク:

- 191 ページの『スプリット・ミラーを使用したデータベースのクローン作成』
- 194 ページの『スプリット・ミラーをバックアップ・イメージとして使用する』

関連資料:

- 289 ページの『db2inidb - ミラー・データベースの初期化』

- 「コマンド・リファレンス」の『db2relocatedb - データベースの再配置コマンド』
- 「コマンド・リファレンス」の『SET WRITE コマンド』

スプリット・ミラーをバックアップ・イメージとして使用する

バックアップ・イメージとして使用するために、1 次データベースのスプリット・ミラーを作成するには、以下の手順に従ってください。この手順は、1 次データベースでデータベース・バックアップ操作を実行する代わりに使用できます。

手順:

スプリット・ミラーを 『バックアップ・イメージ』 として使用するには、以下のステップに従ってください。

1. 1 次データベース上で入出力をサスペンドします。
   ```
   db2 set write suspend for database
   ```
2. 該当するコマンドを使用して、 1 次データベースからミラーを分割します。
3. 1 次データベース上で入出力を再開します。
   ```
   db2 set write resume for database
   ```
4. 1 次システムで障害が発生し、バックアップからのリストアの必要が生じます。
5. 1 次データベース・インスタンスを停止します。
   ```
   db2stop
   ```
6. 該当するコマンドを使用して、スプリットされたデータを 1 次システム上にコピーします。**その際、2 次システム側にあるスプリットされたログ・ファイルをコピーしないでください。** 1 次システム側にあるログが、ロールフォワード・リカバリーに必要となります。
7. 1 次データベース・インスタンスを開始します。
   ```
   db2start
   ```
8. 1 次データベースを初期化します。
   ```
   db2inidb database_alias as mirror
   ```
9. 1 次データベースを、to end of logs オプションで、あるいは、to point-in-time にてポイント・イン・タイム指定で、ロールフォワードし、stop オプションで、ロールフォワードを完了させます。

関連概念:

- 189 ページの『オンライン・スプリット・ミラーおよびサスペンド入出力サポートによる高可用性』

関連タスク:

- 191 ページの『スプリット・ミラーを使用したデータベースのクローン作成』
- 192 ページの『スプリット・ミラーをスタンバイ・データベースとして使用する』

関連資料:

- 289 ページの『db2inidb - ミラー・データベースの初期化』

UNIX ベースのシステム用の障害モニター機能

UNIX® ベースのシステムにおいて、障害モニター機能は、 DB2 を確実に稼働させておくようにする一連のプロセスを通して、非クラスター DB2® 環境の可用性を向上させます。すなわち、*init* デーモンが障害モニター・コーディネーター (FMC) をモニターし、 FMC が障害モニターをモニターし、障害モニターが DB2 をモニターします。

障害モニター・コーディネーター (FMC) は、 UNIX ブート・シーケンスで開始される障害モニター機能のプロセスです。 *init* デーモンは FMC を開始し、それが異常終了した場合には再開します。 FMC は、DB2 インスタンスごとに 1 つの障害モニターを開始します。それぞれの障害モニターは、デーモン・プロセスとして実行され、 DB2 インスタンスと同等のユーザー特権を持っています。障害モニターが開始すると、それが不完全な状態で終了することがないようにモニターされます。障害モニターに障害が起きた場合には、それは FMC により再開されます。さらに障害モニターは、それぞれ 1 つの DB2 インスタンスのモニターを受け持ちます。 DB2 インスタンスか不完全な状態で終了する場合には、障害モニターがそれを再開します。

注:

1. 高可用性クラスタリング製品 (すなわち、HACMP または MSCS) を使用している場合には、インスタンスの始動とシャットダウンはそのクラスタリング製品により制御されるため、障害モニター機能をオフにする必要があります。
2. 障害モニターは、**db2stop** コマンドが実行された場合のみ非アクティブになります。 DB2 インスタンスがその他の仕方でシャットダウンした場合には、障害モニターはそれを再開します。

障害モニター・レジストリー・ファイル

障害モニター・レジストリー・ファイルは、障害モニター・デーモンが開始される時、各物理マシン上のすべてのインスタンスごとに作成されます。このファイルの値により、障害モニターの動作を指定します。このファイルは、`/sqllib/` ディレクトリーにあり、 `fm.<machine_name>.reg` という名前が付いています。このファイルは、**db2fm** コマンドを使用して変更できます。その項目は以下のとおりです。

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

パラメーターの意味は以下のとおりです。

FM_ON
障害モニターを始動するかしないかを指定します。値が `NO` に設定されている場合、障害モニター・デーモンは始動しないか、すでに始動していた場合には停止されます。デフォルト値は `NO` です。

FM_ACTIVE
障害モニターがアクティブであるかそうでないかの指定をします。障害モニ

ターは、FM_ON および FM_ACTIVE が両方とも YES に設定されている時にのみ作動します。 FM_ON が YES に設定されており、FM_ACTIVE が NO に設定されている場合には、障害モニター・デーモンは始動しますが、アクティブにはなりません。これは、DB2 が停止した場合でも、DB2 をオンラインに戻そうとしないことを意味します。デフォルト値は YES です。

START_TIMEOUT
障害モニターがモニターしているサービスを障害モニターが始動するまでの時間の最大の長さを指定します。デフォルト値は 600 秒です。

STOP_TIMEOUT
障害モニターがモニターしているサービスを障害モニターが停止するまでの時間の最大の長さを指定します。デフォルト値は 600 秒です。

STATUS_TIMEOUT
障害モニターがモニターしているサービスの状況を障害モニターが入手するまでの時間の最大の長さを指定します。デフォルト値は 20 秒です。

STATUS_INTERVAL
モニタしているサービスの状況を取得するために行う、連続した 2 回の呼び出しの間の最小時間を指定します。デフォルト値は 20 秒です。

RESTART_RETRIES
モニターしているサービスの状況を取得する際、状況が取得できなかった場合に障害モニターが試行を繰り返す回数を指定します。この数に達すると、障害モニターは、そのサービスがオフラインになっていると判断して、そのサービスをオンラインに戻そうとします。デフォルト値は 3 です。

ACTION_RETRIES
サービスをオンラインにするために障害モニターが試行する回数を指定します。デフォルト値は 3 です。

NOTIFY_ADDRESS
障害モニターが通知メッセージを送信する相手の E メール・アドレスを指定します。デフォルトは、<instance_name>@<machine_name> です。

このファイルは、**db2fm** コマンドを使用して変更できます。たとえば:

インスタンス DB2INST1 の START_TIMEOUT の値を 100 秒に更新するには、以下のように発行します。

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

インスタンス DB2INST1 の STOP_TIMEOUT の値を 200 秒に更新するには、以下のように発行します。

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

インスタンス DB2INST1 の START_TIMEOUT の値を 100 秒に更新し、STOP_TIMEOUT の値を 200 秒に更新するには、以下のように発行します。

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

インスタンス DB2INST1 の障害モニターを始動するには、以下のように発行します。

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

インスタンス DB2INST1 の障害モニターを停止するには、以下のように発行します。

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

注: 障害モニター・レジストリー・ファイルが存在しない場合には、デフォルト値が使用されます。

関連資料:

- 197 ページの『db2fm - DB2 障害モニター』

db2fm - DB2 障害モニター

DB2 障害モニター・デーモンを制御します。 **db2fm** を使用すると、障害モニターを構成できます。

このコマンドは、UNIX 系プラットフォームでのみ利用できます。

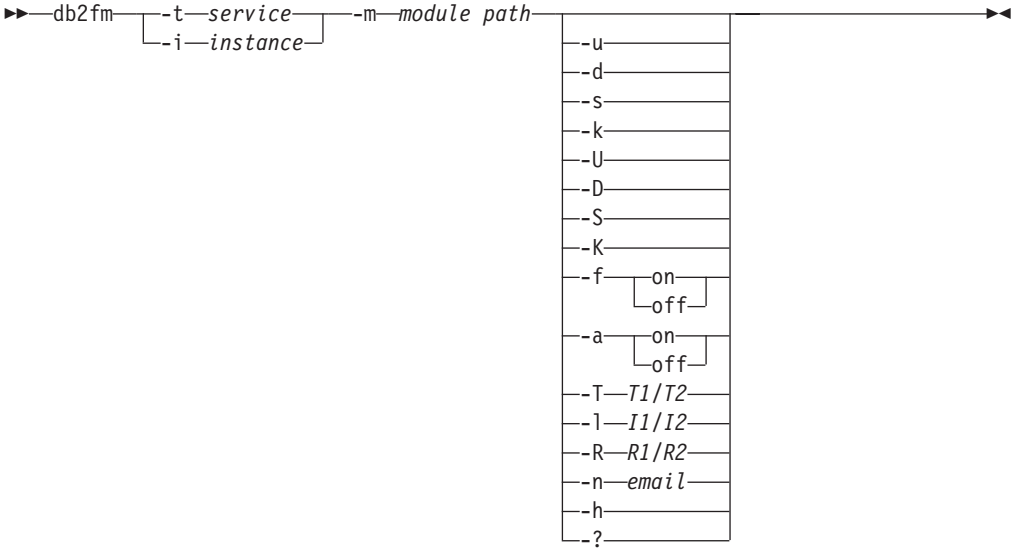
権限:

コマンド実行対象のインスタンスに対する許可。

必要な接続:

なし。

コマンド構文:



コマンド・パラメーター:

-m *module-path*
モニター対象製品の障害モニター共有ライブラリーの絶対パスを定義します。デフォルトは $INSTANCEHOME/sqllib/lib/libdb2gcf です。

-t *service*
ユニークなテキスト記述子をサービスに対して指定します。

-i *instance*
サービスのインスタンスを定義します。

-u サービスを開始します。

-U 障害モニター・デーモンを開始します。

-d サービスを停止します。

-D 障害モニター・デーモンを停止します。

-k サービスを強制終了します。

-K 障害モニター・デーモンを強制終了します。

-s サービスの状況を戻します。

-S 障害モニター・デーモンの状況を戻します。

注: サービスまたは障害モニターの状況は、次のいずれかになります。

- 適切にインストールされていません (Not properly installed)
- 適切にインストールされていますが、活動状態にありません (INSTALLED PROPERLY but NOT ALIVE)
- 活動状態ですが、使用できません (保守) (ALIVE but NOT AVAILABLE (maintenance))
- 使用可能です (AVAILABLE)
- 不明 (UNKNOWN)

-f *on\off*
障害モニターをオン/オフにします。

注: このオプションがオフに設定される場合、障害モニター・デーモンは開始されないか、デーモンが実行中の場合は終了されます。

-a *on\off*
障害モニターを活動化または非活動化します。

注: このオプションがオフに設定されると、障害モニターはアクティブでモニターしません。これは、サービスが停止する場合、再び始動されないことを意味します。

-T *T1/T2*
開始および停止タイムアウトを上書きします。

たとえば、

- -T 15/10 は、2 つのタイムアウトをそれぞれ更新します。
- -T 15 は、開始タイムアウトを 15 秒に更新します。
- -T /10 は、停止タイムアウトを 10 秒に更新します。

-I *I1/I2*
状況インターバル、タイムアウトをそれぞれ設定します。

-R *R1/R2*
中止する前に再試行される状況メソッドおよびアクションの回数を設定します。

-n *email*
イベント通知用の E メール・アドレスを設定します。

-h 使用法を画面に表示します。

-? 使用法を画面に表示します。

第 7 章 高可用性災害時リカバリー (HADR)

高可用性災害時リカバリーの概要

DB2® Universal Database (DB2 UDB) の高可用性災害時リカバリー (HADR) は、部分サイト障害と完全サイト障害の両方に高可用性の解決策を提供する、データベース複製機能です。 HADR は、データの変更内容を、1 次データベースと呼ばれるソース・データベースから、スタンバイ・データベースと呼ばれるターゲット・データベースへ複製することにより、データを損失から保護します。 HADR を使用しないデータベースのことを、標準データベースといいます。

データベースのほとんどまたはすべてで保護が必要な場合、または、スタンバイ・データベースで自動的に複製する必要のある DDL 操作を実行する場合、 HADR は最善の選択オプションになるかもしれません。

アプリケーションは、現行の 1 次データベースにのみアクセスできます。スタンバイ・データベースへの更新は、1 次データベースで生成されてスタンバイ・データベースへ送られるログ・データをロールフォワードすることによって行われます。

部分的なサイト障害は、ハードウェア、ネットワーク、またはソフトウェア (DB2 またはオペレーティング・システム) 障害によって生じることがあります。 HADR を使用しない場合、部分サイト障害では、データベース管理システム (DBMS) サーバーか、データベースが存在するマシンをリブートする必要があります。データベースおよびそれが存在するマシンを再始動するのにかかる時間の長さは、予測できません。データベースが整合した状態に戻って使用可能になるまでに、数分かかることがあります。 HADR を使用すると、スタンバイ・データベースが数秒で処理を引き継ぐことができます。さらに、自動クライアント転送機能、あるいはアプリケーションで再試行ロジックを組むことにより、元の 1 次データベースを使用していたクライアントを、スタンバイ・データベース (新しい 1 次データベース) に接続し直すことができます。

全サイト障害は、火災などの災害によってサイト全体が破壊される場合に生じ得ます。 HADR では、1 次データベースとスタンバイ・データベースとの通信に TCP/IP を使用するため、それぞれが別の場所に置かれていても構いません。たとえば、1 次データベースをある都市の本社に置き、スタンバイ・データベースを別の都市の営業所に置くことができます。 1 次サイトで災害が生じる場合、リモートのスタンバイ・データベースが、完全な DB2 機能を備えた 1 次データベースとしてサイトを引き継ぎ、データの可用性が維持されます。テークオーバー操作が行われた後で、元の 1 次データベースのバックアップを使用して、 1 次データベースの状態に戻すことができます。このことを、フェイルバックといいます。

HADR を使用する場合、同期、準同期、または非同期のいずれかの同期モードを指定して、データ消失の危険からの守るためのレベルを選択できます。

障害の発生した元の 1 次サーバーの修復後に、2 つのデータベースが整合可能であれば、元の 1 次サーバーを HADR ペアにスタンバイ・データベースとして再度加

えることができます。元の 1 次データベースが、スタンバイ・データベースとして HADR ペアのシステムに再統合された後で、データベースの役割を切り替えて、元の 1 次データベースをもう一度 1 次データベースにすることができます。

HADR は、DB2 製品ファミリーで提供されている、いくつかのレプリケーション・ソリューションの 1 つに過ぎません。 DB2 Information Integrator および DB2 UDB のバージョン 8.2 には、特定の構成で、高可用性を実現するためにも使用できる SQL レプリケーションおよび Q レプリケーション・ソリューションが含まれています。これらの機能は、論理的に整合したデータベース表のコピーを、複数の場所で維持するものです。さらに、列および行のフィルター操作、データ形式変更、表のコピーの更新のサポートなど、柔軟かつ複雑な機能を備えていますし、パーティション・データベース環境でも使用できます。

関連概念:

- 188 ページの『ログ・シッピングによる高可用性』
- 214 ページの『自動クライアント転送および高可用性災害時リカバリー』
- 211 ページの『高可用性災害時リカバリーの同期モード』
- 202 ページの『高可用性災害時リカバリーのシステム要件』
- 204 ページの『高可用性災害時リカバリーの制約事項』
- 204 ページの『高可用性災害時リカバリー用のデータベース構成』
- 「*IBM DB2 Information Integrator レプリケーションとイベント・パブリッシング入門*」の『Q レプリケーションと高可用性災害時リカバリー (HADR) の比較』

高可用性災害時リカバリーのシステム要件

高可用性災害時リカバリー (HADR) での最良のパフォーマンスを実現するには、システムが、ハードウェア、オペレーティング・システム、および DB2® Universal Database (DB2 UDB) についての以下の要件を満たすようにします。

推奨: パフォーマンスを良くするため、1 次データベースが存在するシステムと、スタンバイ・データベースが存在するシステムとで、同じハードウェアおよびソフトウェアを使用してください。スタンバイ・データベースが存在するシステムのリソースが、1 次データベースが存在するシステムのリソースよりも少ない場合、スタンバイ・データベースは、1 次データベースによって生成されたトランザクションを処理できない可能性があります。これにより、スタンバイ・データベースが遅れるか、1 次データベースのパフォーマンスが低下する場合があります。フェイルオーバー状態では、新規 1 次データベースには、クライアント・アプリケーションに適切に対応できるだけのリソースが必要です。

ハードウェアおよびオペレーティング・システム要件:

推奨: HADR 1 次データベースとスタンバイ・データベースとで、同じホスト・コンピューターを使用してください。つまり、同じベンダーの製品で、同じアーキテクチャーでなければならないということです。

1 次データベースとスタンバイ・データベースのオペレーティング・システムは、パッチも含めて、同じバージョンでなければなりません。アップグレード時には、短期間だけこのルールに違反できますが、細心の注意が必要です。

HADR ホスト・マシン間では、TCP/IP インターフェースが使用できる必要があり、高速大容量のネットワークが推奨されます。

DB2 UDB 要件:

1 次データベースとスタンバイ・データベースとで使用されるデータベース・バージョンは、同じでなければなりません。アップグレード時には、スタンバイ・データベースのデータベース・バージョンは、短期間だけ、1 次データベースよりも新しくても構いません。 1 次データベースの DB2 UDB バージョンは、スタンバイ・データベース・バージョンより新しいものであってはなりません。

1 次データベースとスタンバイ・データベースの両方の DB2 UDB ソフトウェアは、同じビット・サイズ (32 または 64 ビット) にする必要があります。表スペースとそのコンテナーは、1 次データベースとスタンバイ・データベースとで同じでなければなりません。同一でなければならないプロパティーには、表スペース・タイプ (DMS または SMS)、表スペース・サイズ、コンテナー・パス、コンテナー・サイズ、およびコンテナー・ファイル・タイプ (ロー・デバイスまたはファイル・システム) があります。ログ・ファイル用に割り振られたスペース量も、1 次データベースとスタンバイ・データベースの両方で同じにする必要があります。

1 次データベースで、CREATE TABLESPACE、ALTER TABLESPACE、または DROP TABLESPACE などの表スペース・ステートメントを発行すると、スタンバイ・データベースで再生されます。 1 次データベースで表スペース・ステートメントを発行する前に、関係する装置が両方のデータベースでセットアップされていることを確認する必要があります。

表スペースのセットアップが、1 次データベースとスタンバイ・データベースとで同一でない場合、スタンバイ・データベースでのログの再生時に、 OUT OF SPACE や TABLE SPACE CONTAINER NOT FOUND などのエラーが生じる可能性があります。これが生じた場合、影響を受ける表スペースは、ロールフォワード・ペンディング状態になり、その後のログ適用では無視されます。テークオーバー操作が行われる場合、アプリケーションでは表スペースを使用できません。必要なバックアップ・イメージおよびログ・ファイルのアーカイブが使用可能である場合、まず IGNORE ROLLFORWARD CONTAINER OPERATIONS オプションを指定した SET TABLESPACE CONTAINERS ステートメントを発行してから、ROLLFORWARD コマンドを発行することにより、表スペースをリカバリーすることができます。

1 次データベースとスタンバイ・データベースとで、同じデータベース・パスである必要はありません。相対コンテナー・パスが使用される場合、同じ相対パスであっても、1 次データベースとスタンバイ・データベースとで、異なる絶対コンテナー・パスにマッピングすることが可能です。

バッファー・プール要件:

バッファー・プール操作もスタンバイ・データベースで再生されるため、 1 次データベースとスタンバイ・データベースのメモリー量を同じにしておくことは重要です。

関連概念:

- 204 ページの『高可用性災害時リカバリーの制約事項』

関連タスク:

- 235 ページの『高可用性災害時リカバリー環境での回転アップグレードの実行』

高可用性災害時リカバリーの制約事項

次のリストは、高可用性災害時リカバリー (HADR) の制約事項をサマリーしています。

- HADR は、DB2® UDB Enterprise Server Edition (ESE) でのみサポートされます。しかし、ESE 上に複数のデータベース・パーティションがある場合にはサポートされません。
- 1 次データベースとスタンバイ・データベースでは、同じオペレーティング・システム・バージョンであることと、同じバージョンの DB2 UDB を使用することが求められます。ただし、アップグレード時の短期間は除きます。
- 1 次データベースとスタンバイ・データベースの DB2 UDB リリースは、同じビット・サイズ (32 または 64 ビット) にする必要があります。
- スタンバイ・データベースでの読み取りは、サポートされていません。クライアントは、スタンバイ・データベースへ接続できません。
- ログ・アーカイブは、現在の 1 次データベースによってのみ実行可能です。
- スタンバイ・データベースでのバックアップ操作はサポートされていません。
- データベース構成パラメーターやリカバリー履歴ファイルへの変更など、ログに記録されない操作は、スタンバイ・データベースに複製されません。
- COPY NO オプションを指定したロード操作はサポートされません。
- データ・リンクの使用はサポートされていません。

関連概念:

- 201 ページの『高可用性災害時リカバリーの概要』
- 202 ページの『高可用性災害時リカバリーのシステム要件』
- 216 ページの『高可用性災害時リカバリー用に複製された操作』
- 217 ページの『高可用性災害時リカバリー用の複製されない操作』

高可用性災害時リカバリー用のデータベース構成

高可用性災害時リカバリー (HADR) での最良のパフォーマンスを実現するには、データベース構成が以下の要件を満たすようにします。

推奨: データベース構成パラメーターとデータベース・マネージャー構成パラメーターは、可能な限り 1 次データベースおよびスタンバイ・データベースが存在するシステムで同一にする必要があります。構成パラメーターがスタンバイ・データベースで正しく設定されていないと、以下の問題が生じる可能性があります。

- 1 次データベースから送られたログ・ファイルの再生中に、スタンバイ・データベースにエラー・メッセージが戻されることがあります。

- テークオーバー操作後に、新しい 1 次データベースはワークロードを処理できないため、パフォーマンス上の問題が生じるか、アプリケーションが元の 1 次データベースに接続されているときには受け取ることのないエラー・メッセージを受け取ります。

1 次データベースでの構成パラメーターへの変更は、自動的にスタンバイ・データベースへ伝搬されるわけではなく、スタンバイ・データベース上で手動で行う必要があります。動的構成パラメーターの場合、データベース管理システム (DBMS) またはデータベースをシャットダウンして再始動しなくても、変更は有効になります。動的構成パラメーターではない場合、変更はスタンバイ・データベースの再始動後に有効になります。

注: 例外は、LOGFILSIZ データベース構成パラメーターです。このパラメーターはスタンバイ・データベースに複製されませんが、両方のデータベースでログ・ファイルを同一にするために、スタンバイ・データベースは、ローカル LOGFILSIZ 構成を無視し、1 次データベースのログ・ファイルのサイズと一致するローカル・ログ・ファイルを作成します。

スタンバイ・データベースでのログ受信バッファー・サイズ:

デフォルトでは、スタンバイ・データベースでのログ受信バッファー・サイズは、1 次データベースの LOGBUFSZ 構成パラメーターに指定した値の 2 倍です。このサイズでは不十分な場合もあります。たとえば、HADR 同期モードが非同期であり、1 次データベースおよびスタンバイ・データベースがピア状態 (詳細は第 7 章の『高可用性災害時リカバリーでのスタンバイ・データベース状態』内を参照してください) のときに、 1 次データベースが高いトランザクション・ロードを経験する場合、スタンバイ・データベースのログ受信バッファーが満杯になり、 1 次データベースからのログ送信操作は停止する可能性があります。これらの一時ピークを管理するため、DB2_HADR_BUF_SIZE レジストリー変数を変更して、スタンバイ・データベースのログ受信バッファーのサイズを増やすことができます。

ロード操作および HADR:

1 次データベースで COPY YES オプションを指定したロード操作が実行される場合、コマンドは 1 次データベースで実行され、 LOAD コマンドで指定したパスまたは装置経由でコピーにアクセスできる限り、データはスタンバイ・データベースに複製されます。スタンバイ・データベースがデータにアクセスできない場合、表が保管される表スペースが、スタンバイ・データベース上で使用不能としてマークされます。スタンバイ・データベースは、この表スペースに関係するこれ以降のログ・レコードを飛ばします。

1 次データベースで NONRECOVERABLE オプションを指定したロード操作が実行される場合、コマンドは1 次データベースで実行され、スタンバイ・データベースの表は使用不能としてマークされます。スタンバイ・データベースは、この表に関係するこれ以降のログ・レコードを飛ばします。 COPY YES および REPLACE オプションを指定した LOAD コマンドを発行して表を戻すか、表をドロップしてスペースをリカバリーするかを選択できます。

COPY NO オプションを指定したロード操作の実行は、HADR ではサポートされないため、コマンドは、NONRECOVERABLE オプションを指定したロード操作へ自動的に変換されます。 COPY NO オプションを指定したロード操作を、COPY YES

オプションを指定したロード操作へ変換できるようにするには、 1 次データベースで DB2_LOAD_COPY_NO_OVERRIDE レジストリー変数を設定します。このレジストリー変数は、スタンバイ・データベースには無視されます。 1 次データベースに指定された装置またはディレクトリーが、スタンバイ・データベース側で、同じパス、装置、またはロード・ライブラリーを使用してアクセス可能であることを確認してください。

Tivoli® Storage Manager (TSM) を使用して、COPY YES オプションを指定したロード操作を実行する場合、 1 次データベースおよびスタンバイ・データベースで VENDOROPT 構成パラメーターを設定しなければならない場合があります。 TSM の構成方法に応じて、1 次データベースの値とスタンバイ・データベースの値が同じではない場合があります。さらに、TSM を使用して、COPY YES オプションを指定したロード操作を実行するときには、 GRANT オプションを指定した **db2adutl** コマンドを発行し、スタンバイ・データベースに、ロードされるファイルの読み取りアクセスを GRANT する必要があります。

表データが、COPY YES オプションを指定したロード操作によって複製される場合、索引は、次のようにして複製されます。

- 索引モードが REBUILD に設定されていて、表属性が LOG INDEX BUILD に設定されている場合、または表属性が DEFAULT に設定されていて、LOGINDEXBUILD データベース構成パラメーターが ON に設定されている場合、 1 次データベースは、再作成索引オブジェクトをコピー・ファイルに組み込み、スタンバイ・データベースが索引オブジェクトを複製できるようにします。ロード操作の前に、スタンバイ・データベースの索引オブジェクトが使用不能としてマークされる場合、索引の再作成の結果として、ロード操作後に再び使用可能になります。
- 索引モードが INCREMENTAL に設定されていて、表属性が LOG INDEX BUILD に設定されている場合、または表属性が NULL に設定されていて、 1 次データベースの LOGINDEXBUILD データベース構成パラメーターが ON に設定されている場合、スタンバイ・データベースの索引オブジェクトは、ロード操作前に使用不能としてマークされていない場合にのみ更新されます。それ以外の場合には、索引はスタンバイ・データベース上で使用不能としてマークされます。

HADR 構成パラメーター:

いくつかの新しいデータベース構成パラメーターが、HADR をサポートするようになりました。これらのパラメーターを設定しても、データベースの役割は変更されません。データベースの役割を変更するには、START HADR または STOP HADR コマンドを発行する必要があります。

HADR 構成パラメーターに対する変更は、データベースがシャットダウンされて再始動されるまで有効にはなりません。パーティション・データベース環境では、HADR 構成パラメーターは表示されて変更可能ですが、パラメーターの設定内容は無視されます。

1 次データベースのローカル・ホスト名は、スタンバイ・データベースのリモート・ホスト名と同じでなければならず、スタンバイ・データベースのローカル・ホスト名は、1 次データベースのリモート・ホスト名と同じでなければなりません。各データベースのローカル・ホストとリモート・ホストを設定するには、

HADR_LOCAL_HOST および HADR_REMOTE_HOST 構成パラメーターを使用します。ローカル・ホスト名とリモート・ホスト名の構成の整合性は、接続の確立時に検査され、指定されたリモート・ホストは対象として考慮していたノードであることが確認されます。

同期モード (HADR_SYNCMODE) とタイムアウト期間 (HADR_TIMEOUT) は、 1 次データベースとスタンバイ・データベースの両方で同じでなければなりません。これらの構成パラメーターの整合性は、HADR ペアが接続を確立するときに検査されます。

TCP 接続は、1 次データベースとスタンバイ・データベース間で通信する際に使用されます。開始中であるか、接続が失われているためにスタンバイ・データベースに接続されていない 1 次データベースは、ローカル・ポートで新しい接続を listen します。 1 次データベースに接続されていないスタンバイ・データベースは、引き続き、リモート・ホストへの接続要求を発行します。

ローカル・ホストおよびローカル・サービス・パラメーター (HADR_LOCAL_HOST、HADR_LOCAL_SVC) は、 1 次データベースでのみ使用されますが、これらのパラメーターをスタンバイ・データベースでも設定して、スタンバイ・データベースを 1 次データベースとして機能させる場合に備える必要があります。

次のサンプル構成は、1 次データベースおよびスタンバイ・データベース用です。

1 次データベースの場合:

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

スタンバイ・データベースの場合:

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

関連概念:

- 201 ページの『高可用性災害時リカバリーの概要』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『hadr_db_role - HADR データベース役割構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『hadr_local_host - HADR ローカル・ホスト名構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『hadr_local_svc - HADR ローカル・サービス名構成パラメーター』

- 「*管理ガイド: パフォーマンス*」の『hadr_remote_host - HADR リモート・ホスト名構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『hadr_remote_inst - リモート・サーバーの HADR インスタンス名構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『hadr_remote_svc - HADR リモート・サービス名構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『hadr_syncmode - 対等状態にあるログ書き込みのための HADR 同期モード構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『hadr_timeout - HADR タイムアウト値構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『vendoropt - ベンダー・オプション構成パラメーター』

高可用性災害時リカバリーでのスタンバイ・データベースの状態

高可用性災害時リカバリー (HADR) 機能を使用する場合、スタンバイ・データベースが開始されると、ローカル・キャッチアップ状態になり、ローカル・ログ・パスのログ・ファイルを読み取ろうとします。ローカル・ログ・パスにログ・ファイルがなく、ログ・アーカイブ方式が指定されている場合、ログ・ファイルは指定の方式で取り出されます。ログ・ファイルが読み取られたら、スタンバイ・データベース上で再生されます。この時点では、1 次データベースへの接続は必要ありません。ただし、接続が存在しない場合、スタンバイ・データベースは 1 次データベースへ接続しようとします。ローカル・ログ・ファイルの最後になったら、スタンバイ・データベースは、リモート・キャッチアップ・ペンディング状態になります。

1 次データベースへの接続が確立されるまで、スタンバイ・データベースはリモート・キャッチアップ・ペンディング状態のままです。接続が確立された時点で、スタンバイ・データベースはリモート・キャッチアップ状態になります。このときに、1 次データベースは、ログ・パスまたはログ・アーカイブ方式を使用してログ・データを読み取り、ログ・ファイルをスタンバイ・データベースへ送信します。スタンバイ・データベースは、ログ・データを受信して適用します。ディスク上のすべてのログ・ファイルがスタンバイ・データベースで適用されると、 1 次システムとスタンバイ・システムはピア状態になります。

ピア状態では、1 次データベースがログ・ページをディスクにフラッシュするたびに、そのログ・ページがスタンバイ・データベースに送られます。そのログ・ページは、スタンバイ・データベースのローカル・ログ・ファイルに書き込まれ、 1 次データベースとスタンバイ・データベースのログ・ファイル・シーケンスが同一になります。続いて、そのログ・ページをスタンバイ・データベースで適用できます。

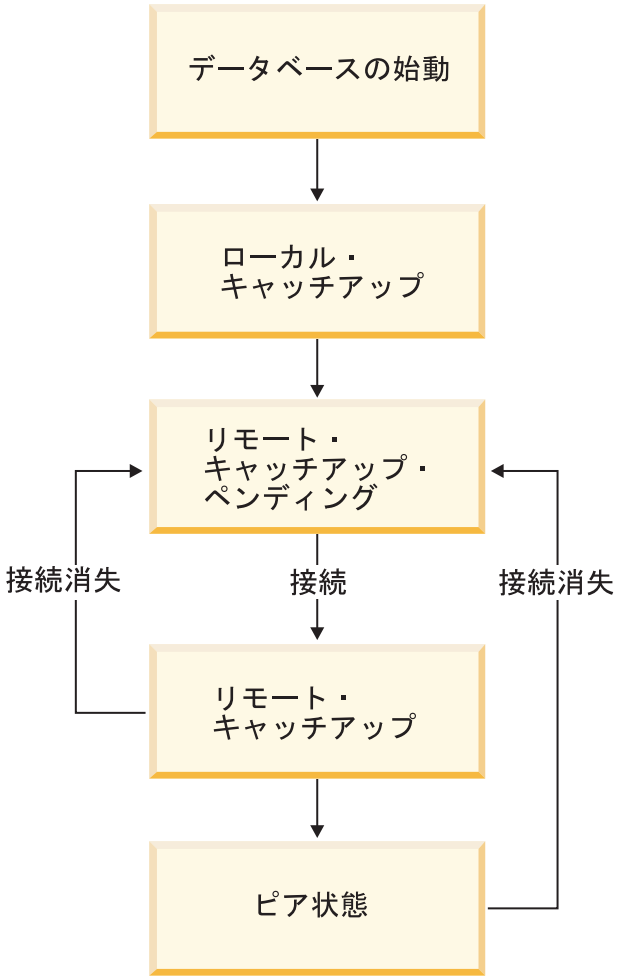


図 16. スタンバイ・データベースの状態

スタンバイ・データベースの状態は、DATABASE ON オプションを指定した GET SNAPSHOT コマンドを発行して確認できます。たとえば、スタンバイ・データベース MUSIC を持つ場合、次のコマンドを発行して状態を確認できます。

```
get snapshot for database on music
```

このコマンドがスタンバイ・データベースで発行される場合、スタンバイ・データベースのいずれかの状態が、 State フィールドに戻されます。照会がスタンバイ・データベースに接続された 1 次データベースで発行される場合、スタンバイ・データベースの状態が戻されます。 1 次データベースがスタンバイ・データベースに接続されていない場合、 disconnected が戻されます。

次の出力は、GET SNAPSHOT コマンドで戻された HADR status セクションを示しています。

```
HADR status

  Role                   = Primary
  State                  = Peer
  Synchronization mode   = Sync
  Connection status      = Connected, 11-03-2002 12:23:09.35092
  Heartbeat missed       = 0
  Local host             = host1.ibm.com
  Local service          = hadr_service
```

```
Remote host            = host2.ibm.com
Remote service         = hadr_service
Remote instance        = dbinst2
timeout(seconds)       = 120
Primary log position(file, page, LSN) = S0001234.LOG, 12, 0000000000BB800C
Standby log position(file, page, LSN) = S0001234.LOG, 12, 0000000000BB800C
Log gap running average(bytes) = 8723
```

注:

1. 1 次データベースとスタンバイ・データベースとの間の接続が、リモート・キャッチアップまたはピア状態で消失する場合、スタンバイ・データベースは、リモート・キャッチアップ・ペンディング状態になります。
2. スタンバイ・データベースは受信するログ・ファイルをローカル・ログ・パスに書き込むため、 1 次データベースとスタンバイ・データベース両方のログ・パスとして、共用ネットワークやローカル・ファイル・システムを使用してはなりません。 DB2® が共用ログ・パスを検出すると、エラー・メッセージが表示されます。
3. キャッチアップ処理の速度を速めるには、共用ログ・アーカイブ装置を使用できます。しかし、その共用装置が、磁気テープ・ドライブのようなシリアル装置である場合、読み取り操作と書き込み操作が混在するため、 1 次データベースとスタンバイ・データベースの両方で性能低下が生じる可能性があります。
4. 1 次データベースのログ・ファイルは、ローカル・キャッチアップで使用するスタンバイ・データベースのログ・パスへ、手動でコピーすることができます。これは、スタンバイ・データベースを開始する前に実行する必要があります。これは、ローカル・ログ・ファイルの最後に到達すると、スタンバイ・データベースは、リモート・キャッチアップ・ペンディング状態になり、もう一度ログ・ファイルにアクセスしようとしなくなるためです。さらに、スタンバイ・データベースがリモート・キャッチアップ状態になる場合、ログ・ファイルをログ・パスにコピーしようとすると、スタンバイ・データベースはローカル・ログ・ファイルの書き込みを許しません。スタンバイ・データベースがリモート・キャッチアップ・ペンディング状態になった後、さらに多くのローカル・ログ・ファイルが使用可能になる場合、スタンバイ・データベースをシャットダウンしてから再始動して、もう一度ローカル・キャッチアップ状態にすることができます。

関連概念:

- 49 ページの『ログ・ファイルの管理』
- 53 ページの『ログ・アーカイブを使用したログ・ファイルの管理』

関連資料:

- 375 ページの『付録 G. データベース・リカバリー用のユーザー出口』
- 「コマンド・リファレンス」の『GET SNAPSHOT コマンド』
- 40 ページの『データベース・ロギングの構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logarchmeth1 - 1 次ログ・アーカイブ方式構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logarchmeth2 - 2 次ログ・アーカイブ方式構成パラメーター』

高可用性災害時リカバリーの同期モード

高可用性災害時リカバリー (HADR) を使用すると、3 つの同期モードのいずれかを指定して、データ消失の危険から保護するときのレベルを選択できます。同期モードは、1 次データベースとスタンバイ・データベースとの間で、ログ書き込みが管理される方法を示すものです。これらのモードは、1 次データベースとスタンバイ・データベースがピア状態である場合にのみ適用されます。

同期モードを設定するには、HADR_SYNCMODE 構成パラメーターを使用します。有効な値は以下のとおりです。

SYNC (同期)

このモードは、トランザクション消失に対する最高度の保護を実現しますが、この使用のために、3 つのモードの中でもトランザクション応答時間は最長になります。

このモードでは、ログ書き込みは、ログが 1 次データベースのログ・ファイルに書き込まれ、ログがスタンバイ・データベース上のログ・ファイルにも書き込まれたことの確認通知をスタンバイ・データベースから 1 次データベースが受信した場合にのみ、成功したものと考えられます。ログ・データは、確実に両方のサイトに保管されます。

スタンバイ・データベースがログ・レコードを再生する前に破損する場合は、次回に開始したときに、ローカル・ログ・ファイルから取り出して再生することができます。 1 次データベースに障害が発生する場合、スタンバイ・データベースに対するフェイルオーバーにより、 1 次データベースでコミットされたトランザクションは、スタンバイ・データベースでもコミットされます。フェイルオーバー操作の後で、クライアントが新しい 1 次データベースに再接続する場合、新しい 1 次データベースでコミット済みと報告されているトランザクションが、元の 1 次データベースではコミット済みと報告されていない場合があります。このことが生じるのは、1 次データベースがスタンバイ・データベースからの確認通知メッセージを処理する前に障害が発生する場合です。クライアント・アプリケーションでは、データベースを照会することで、そのようなトランザクションが存在するかどうかを判別する必要があります。

1 次データベースがスタンバイ・データベースへの接続を失う場合、データベースはピア状態であるとは見なされなくなり、トランザクションはスタンバイ・データベースからの確認通知を待機するしかなくなります。データベースの切断時にフェイルオーバー操作が実行される場合、 1 次データベースでコミットされたすべてのトランザクションがスタンバイ・データベースに示される保証はありません。

1 次データベースがピア状態のときにそのデータベースに障害が発生する場合、フェイルオーバー操作の後で、HADR ペアにスタンバイ・データベースとして再度加わることができます。ログがスタンバイ・データベース上のログ・ファイルにも書き込まれたことの確認通知をスタンバイ・データベースから 1 次データベースが受信まで、トランザクションはコミット済みとは見なされないため、1 次データベースのログ・シーケンスは、スタンバイ・データベースのログ・シーケンスと同じになります。元の 1 次データベース (現在のスタンバイ・データベース) は、フェイルオーバー操作以降

に、新しい 1 次データベースで生成される新しいログ・レコードを再生することで、最新の内容に保つ必要があります。

1 次データベースの障害時にピア状態ではない場合、そのログ・シーケンスは、スタンバイ・データベースのログ・シーケンスとは違う可能性があります。フェイルオーバー操作を実行する必要がある場合、1 次データベースとスタンバイ・データベースのログ・シーケンスは異なる場合があります。これは、スタンバイ・データベースはフェイルオーバー後に独自のログ・シーケンスを開始するためです。一部の操作 (たとえば、表のドロップ) は取り消すことができないため、1 次データベースを、新しいログ・シーケンスが作成された時点まで復帰させることは不可能です。ログ・シーケンスが異なる場合、元の 1 次データベースに対して AS STANDBY オプションを指定した START HADR コマンドを発行すると、エラー・メッセージが戻されます。元の 1 次データベースが HADR ペアに正常に再度加わることができた場合、 BY FORCE オプションを指定しない TAKEOVER HADR コマンドを出すことにより、データベースのフェイルバックを実現できます。元の 1 次データベースが HADR ペアに再度加わることができない場合、新しい 1 次データベースのバックアップ・イメージをリストアすることで、スタンバイ・データベースとして再初期設定することができます。

NEARSYNC (準同期)

このモードは同期モードよりもトランザクション応答時間が短いですが、トランザクション消失に対する保護が若干劣ります。

このモードでは、ログ書き込みは、ログ・レコードが 1 次データベースのログ・ファイルに書き込まれ、ログがスタンバイ・システム上のメイン・メモリーにも書き込まれたことの確認通知をスタンバイ・システムから 1 次データベースが受信した場合にのみ、成功したものと考えられます。データの消失は、両方のサイトに同時に障害が発生し、ターゲット・サイトが、受信したすべてのログ・データを不揮発性ストレージに転送していない場合にのみ生じます。

スタンバイ・データベースがログ・レコードをメモリーからディスクへコピーする前に破損する場合、スタンバイ・データベースのログ・レコードが失われます。通常は、スタンバイ・データベースは、再始動時に消失したログ・レコードを 1 次データベースから入手できます。しかし、1 次データベースまたはネットワークでの障害によって、検索が不可能になりフェイルオーバーが必要になる場合、ログ・レコードはスタンバイ・データベースに送られず、そのログ・レコードに関連したトランザクションはスタンバイ・データベースに送られません。

トランザクションが失われる場合、新しい 1 次データベースは、フェイルオーバー操作後の元の 1 次データベースと同じではありません。クライアント・アプリケーションは、これらのトランザクションを再サブミットして、アプリケーションの状態を最新にする必要があります。

1 次データベースとスタンバイ・データベースがピア状態のときに 1 次データベースに障害が発生する場合、完全なリストア操作を使用して再初期設定しないと、元の 1 次データベースはスタンバイ・データベースとして HADR ペアに再度加わることができない可能性があります。フェイルオーバーに、(1 次データベースとスタンバイ・データベースの両方に障害が発生したために) 消失したログ・レコードが関係する場合、 1 次データベー

スおよびスタンバイ・データベース両方のログ・シーケンスは異なり、先にリストア操作を実行せずに、元の 1 次データベースをスタンバイ・データベースとして再始動しようとすると失敗します。元の 1 次データベースが HADR ペアに正常に再度加わることができた場合、 BY FORCE オプションを指定しない TAKEOVER HADR コマンドを出すことにより、データベースのフェイルバックを実現できます。元の 1 次データベースが HADR ペアに再度加わることができない場合、新しい 1 次データベースのバックアップ・イメージをリストアすることで、スタンバイ・データベースとして再初期設定することができます。

ASYNC (非同期)

このモードは、1 次システムに障害が発生する場合に、トランザクションを消失する危険性が一番高いものです。また、3 つのモードの中では、トランザクションの応答時間が最短でもあります。

このモードでは、ログ書き込みは、ログ・レコードが 1 次データベース上のログ・ファイルに書き込まれ、 1 次システムのホスト・マシンの TCP レイヤーに送信される場合にのみ、成功したものと考えられます。 1 次システムはスタンバイ・システムからの確認通知を待たないため、トランザクションがスタンバイ・システムへの送信途上であっても、コミット済みであると見なされる場合があります。

1 次データベース・ホスト・マシン、ネットワーク、またはスタンバイ・データベース上の障害により、転送中のログ・ファイルが消失してしまうことがあります。 1 次データベースが使用可能な場合、ペアのデータベースが接続を再確立するときに、消失しているログ・ファイルをスタンバイ・データベースへ再送できます。しかし、ログ・ファイルが消失しているときにフェイルオーバー操作が必要な場合は、ログ・ファイルと関連するトランザクションはスタンバイ・データベースへ到達しません。 1 次データベースでログ・ファイルが消失したり障害が発生したりする場合、トランザクションは永続的に消失します。

トランザクションが失われる場合、新しい 1 次データベースは、フェイルオーバー操作後の元の 1 次データベースとまったく同じではありません。クライアント・アプリケーションは、これらのトランザクションを再サブミットして、アプリケーションの状態を最新にする必要があります。

1 次データベースとスタンバイ・データベースがピア状態のときに 1 次データベースに障害が発生する場合、完全なリストア操作を使用して再初期設定しないと、元の 1 次データベースはスタンバイ・データベースとして HADR ペアに再度加わることができない可能性があります。フェイルオーバーに、消失したログ・レコードが関係する場合、1 次データベースおよびスタンバイ・データベース両方のログ・シーケンスは異なり、元の 1 次データベースをスタンバイ・データベースとして再始動しようとすると失敗します。非同期モードでフェイルオーバーが生じる場合、ログ・レコードが消失する可能性が高くなるため、 1 次データベースが HADR ペアに再度加わることができなくなる可能性も高くなります。元の 1 次データベースが HADR ペアに正常に再度加わることができた場合、 BY FORCE オプションを指定しない TAKEOVER HADR コマンドを出すことにより、データベースのフェイルバックを実現できます。元の 1 次データベースが HADR

ペアに再度加わることができない場合、新しい 1 次データベースのバックアップ・イメージをリストアすることで、スタンバイ・データベースとして再初期設定することができます。

関連概念:

- 208 ページの『高可用性災害時リカバリーでのスタンバイ・データベースの状態』

関連タスク:

- 234 ページの『テークオーバー操作後のデータベースの再統合』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『hadr_syncmode - 対等状態にあるログ書き込みのための HADR 同期モード構成パラメーター』

自動クライアント転送および高可用性災害時リカバリー

自動クライアント転送機能を高可用性災害時リカバリーと共に使用することにより、クライアント・アプリケーションは、サーバーとの通信の消失からリカバリーし、最小限の中断で作業を続けられます。転送は、サーバーで代替データベース・ロケーションが指定されている場合にのみ可能です。自動クライアント転送は、TCP/IP プロトコルでのみサポートされています。

自動クライアント転送を HADR と共に使用することにより、クライアント・アプリケーションは、テークオーバー操作後に新しい 1 次データベースへ接続できます。自動クライアント転送が使用可能になっていない場合、クライアント・アプリケーションは、エラー・メッセージ SQL30081 を受け取り、サーバーとの接続を確立するための処置は行われません。次の例では、UPDATE ALTERNATE SERVER FOR DATABASE コマンドを使用して、自動クライアント転送と HADR をセットアップする方法が説明されています。

例

システムは、以下のようにしてセットアップします。

- データベース MUSIC がホスト HORNET に存在するものとしてカタログされているクライアントがあります。
- データベース MUSIC は 1 次データベースであり、対応するスタンバイ・データベース MUSIC は、ポート番号 456 でホスト MONTERO 上に存在します。

自動クライアント転送を使用可能にするために、以下のようにしてホスト HORNET のデータベース MUSIC の代替サーバーを更新します。

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

このコマンドの発行後、クライアントは、ホスト HORNET へ正常に接続して、代替サーバー情報を入手することになります。次に、クライアントとホスト HORNET のデータベース MUSIC との間で通信エラーが発生する場合、クライアントは、まずホスト HORNET のデータベース MUSIC へ再接続しようとします。これが失敗する場合、次にクライアントは、ホスト MONTERO のスタンバイ・データベース MUSIC との接続を確立しようとします。

注:

1. 代替ホストのロケーションは、サーバーのシステム・データベース・ディレクトリー・ファイルに保管されています。
2. 自動クライアント転送機能を使用可能にするには、UPDATE ALTERNATE SERVER FOR DATABASE コマンドを使用する必要があります。自動クライアント転送では、HADR_REMOTE_HOST および HADR_REMOTE_SVC データベース構成パラメーターを使用しません。

関連概念:

- 201 ページの『高可用性災害時リカバリーの概要』
- 「*管理ガイド: インプリメンテーション*」の『クライアントの自動転送のインプリメンテーション』

関連資料:

- 「*コマンド・リファレンス*」の『UPDATE ALTERNATE SERVER FOR DATABASE コマンド』
- 「*管理 API リファレンス*」の『db2UpdateAlternateServerForDB - データベースの代替サーバーの更新』

索引ロギングおよび高可用性災害時リカバリー

高可用性災害時リカバリー (HADR) データベースに構成パラメーターを設定する際には、以下の推奨事項を考慮してください。

LOGINDEXBUILD データベース構成パラメーターの使用

推奨: HADR データベースについては、 LOGINDEXBUILD データベース構成パラメーター ON に設定し、索引作成、再作成、および再編成のために完全な情報がログに記録されるようにしてください。索引作成は 1 次システムで時間がかかることがあり、より多くのログ・スペースが必要になる場合がありますが、索引は、HADR ログの再生時にスタンバイ・システムで再作成され、フェイルオーバーが生じるときに使用できるようになります。 1 次システムでの索引作成がログに記録されていない場合で、フェイルオーバーが生じた場合、フェイルオーバーの完了後に残った無効な索引を、アクセスされる前に再作成する必要があります。索引の再作成中には、アプリケーションでアクセスすることはできません。

注: LOG INDEX BUILD 表属性がデフォルト値である NULL に設定される場合、DB2® は、 LOGINDEXBUILD データベース構成パラメーターに指定された値を使用します。 LOG INDEX BUILD 表属性が ON または OFF に設定される場合、 LOGINDEXBUILD データベース構成パラメーターに指定された値は無視されます。

以下のいずれかの理由により、1 つ以上の表で LOG INDEX BUILD 表属性を OFF に設定するよう選択できます。

- 索引作成のロギングをサポートするための十分なアクティブ・ログ・スペースがない。
- 索引データが非常に大きく、表が頻繁にアクセスされないため、テークオーバー操作の終了時に索引を再作成できる。この場合、INDEXREC 構成パラメーターを

RESTART に設定します。表が頻繁にアクセスされないため、この設定では、システムは、テークオーバー操作後に初めて表がアクセスされることを待つのではなく、テークオーバー操作の終了時に索引を再作成します。

1 つ以上の表で LOG INDEX BUILD 表属性が OFF に設定される場合、それらの表に索引作成操作を実行すると、テークオーバー操作を実行するときに索引が再作成されます。同様に、LOG INDEX BUILD 表属性がデフォルト値の NULL に設定され、LOGINDEXBUILD データベース構成パラメーターが OFF に設定される場合、表に対して索引作成操作を実行すると、その表の索引はテークオーバー操作を実行するときに再作成されます。以下のいずれかのアクションを実行することで、索引が再作成されることを防げます。

- 新しい 1 次データベースですべての無効な索引が再作成された後で、データベースのバックアップをとり、スタンバイ・データベースに適用します。 1 次データベース上における無効な索引は通常スタンバイ・データベースで再作成が必要であるとみなされますが、このように 1 次データベースのバックアップをスタンバイ・データベースに適用することで、スタンバイ・データベースは、1 次データベース上で無効な索引作成するために使用するログを適用する必要がなくなります。
- LOG INDEX BUILD 表属性を ON に設定するか、LOG INDEX BUILD 表属性を NULL に設定し、スタンバイ・データベースで LOGINDEXBUILD 構成パラメーターを ON に設定し、索引再作成がログに記録されるようにします。

INDEXREC データベース構成パラメーターの使用

推奨: 1 次データベースとスタンバイ・データベースの両方で、 INDEXREC データベース構成パラメーターを RESTART (デフォルト) に設定してください。これにより、テークオーバー操作の完了後に、無効な索引が再作成されます。索引作成がログに記録されていない場合、この設定により、 DB2 は、無効な索引を調べて再作成できます。このプロセスは、バックグラウンドで生じるものであり、テークオーバー操作が正常に完了した後でも、データベースにアクセスできます。

索引がバックグラウンド再作成索引プロセスによって再作成される前に、特定のトランザクションが無効な索引を含む表にアクセスする場合、無効な索引は、それにアクセスした最初のトランザクションによって再作成されます。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『indexrec - 「索引再作成時点」構成パラメーター』
- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*管理ガイド: パフォーマンス*」の『logindexbuild - 作成済み索引ページのログ構成パラメーター』

高可用性災害時リカバリー用に複製された操作

高可用性災害時リカバリー (HADR) では、以下の操作が、 1 次データベースからスタンバイ・データベースへ複製されます。

- データ定義言語 (DDL)
- データ操作言語 (DML)

- バッファー・プール操作
- 表スペース操作
- オンライン再編成
- オフライン再編成
- ストアード・プロシージャーおよびユーザー定義関数 (UDF) のメタデータ (ただし、関連オブジェクトまたはライブラリー・ファイルではない)

オンライン再編成時には、すべての操作が詳細にログに記録されます。そのため、HADR は、より標準的なデータベース更新の場合よりも、スタンバイ・データベースを遅れさせることなく、操作を複製できます。しかし、この動作は、大量のログ・レコードが生成されるため、システムに大きな影響を与える可能性があります。

オフライン再編成は、オンライン再編成ほど徹底的にログに記録されませんが、一般的に操作は、影響を受ける数百か数千の行ごとにログに記録されます。つまり、スタンバイ・データベースは、各ログ・レコードを待機してから、多数の更新を一度に再生するため、遅れる可能性があるということです。オフライン再編成がクラスター化されていない場合、再編成操作全体の完了後に、1 つのログ・レコードが生成されます。この方法は、スタンバイ・データベースが 1 次データベースに遅れをとらずついて行く能力に大きな影響を与えます。スタンバイ・データベースは、1 次データベースからログ・レコードを受け取った後に再編成全体を実行します。

HADR は、ストアード・プロシージャー、UDF オブジェクト、およびライブラリー・ファイルを複製しません。 1 次データベースとスタンバイ・データベースの両方で、同じパスにファイルを作成する必要があります。スタンバイ・データベースが、参照されているオブジェクトまたはライブラリー・ファイルを検出できない場合、スタンバイ・データベースでのストアード・プロシージャーまたは UDF の呼び出しは失敗します。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『表の再編成』
- 「*管理ガイド: パフォーマンス*」の『索引の再編成』
- 204 ページの『高可用性災害時リカバリー用のデータベース構成』
- 217 ページの『高可用性災害時リカバリー用の複製されない操作』

高可用性災害時リカバリー用の複製されない操作

高可用性災害時リカバリー (HADR) では、データベース・ログを使用して、データをスタンバイ・データベースに複製します。ログに記録されない操作は、1 次データベースで可能ですが、スタンバイ・データベースに複製されません。複製されない操作には、以下の操作が含まれます。ただし、それらの操作に限定されるわけではありません。

- NOT LOGGED INITIALLY オプションを指定して作成された表。
- BLOB および CLOB は複製されません。しかし、それらのスペースは、スタンバイ・データベースに割り振られます。

- データ・リンクは、HADR データベースではサポートされていません。 START HADR または ACTIVATE DATABASE コマンドが発行される場合、または最初のクライアント接続によって、 HADR 役割のデータベース (1 次でもスタンバイでも) が設定される場合で、 DATALINKS データベース・マネージャー構成パラメーターが YES に設定される場合、その操作は失敗します。 HADR データベースを使用するには、DATALINKS 構成パラメーターを NO に設定して、インスタンスをシャットダウンして再始動します。

  注: データベースが標準データベースから 1 次データベースかスタンバイ・データベースに変換されるときに、既存のデータ・リンク列は影響を受けません。 DATALINKS 構成パラメーターが NO に設定されている場合でも、HADR データベースで新しいデータ・リンク列を作成できます。しかし、データ・リンク列を挿入、更新、または選択することはできません。
- UPDATE DATABASE CONFIGURATION および UPDATE DATABASE MANAGER CONFIGURATION コマンドを使用したデータベース構成に対する更新は、複製されません。
- リカバリー履歴ファイルとそれに対する変更は、 1 次データベースからスタンバイ・データベースへ自動的に送られるわけではありません。

  REPLACE HISTORY FILE オプションを指定した RESTORE DATABASE コマンドを発行することにより、 (1 次データベースのバックアップ・イメージから入手した) 履歴ファイルの初期コピーを、スタンバイ・データベースに置くことができます。

  ```
  RESTORE DB KELLY REPLACE HISTORY FILE
  ```

  HADR が初期設定され、1 次データベースでバックアップ・アクティビティーが実行されたら、スタンバイ・データベースの履歴ファイルは古くなってしまいます。次のコマンドを発行して、1 次データベースでバックアップ操作が行われるときに、スタンバイ・データベースの履歴ファイルを更新することができます。

  ```
  RESTORE DB KELLY HISTORY FILE
  ```

  テークオーバー操作が行われ、スタンバイ・データベースに最新の履歴ファイルがある場合、新しい 1 次データベースでのバックアップおよびリストア操作により、履歴ファイルに新しいレコードが生成され、元の 1 次データベースで生成されたレコードとシームレスに混合させられます。履歴ファイルが古いか、項目が欠落している場合、自動増分リストアはできない可能性があります。代わりに、手動での増分リストア操作が必要になります。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『表の再編成』
- 「*管理ガイド: パフォーマンス*」の『索引の再編成』
- 60 ページの『リカバリー履歴ファイルについて』
- 204 ページの『高可用性災害時リカバリー用のデータベース構成』
- 216 ページの『高可用性災害時リカバリー用に複製された操作』

クラスター・マネージャーおよび高可用性災害時リカバリー

高可用性災害時リカバリー (HADR) をクラスター・マネージャーと共に使用して、DBMS の可用性を拡張することができます。 HADR を構成して可用性を拡張するには、次の 2 つの方法があります。

- 1 次データベースとスタンバイ・データベースが同じクラスター・マネージャーによって処理される HADR ペアを設定します。

  この構成は、1 次データベースとスタンバイ・データベースが同じサイトに存在し、できる限り最速のフェイルオーバーが必要とされる環境に最適です。このような環境では、クラッシュ・リカバリーや別のリカバリー方式を使用するよりも、 HADR を使用して DBMS 可用性を保守する方が利点があります。

  クラスター・マネージャーを使用すると、問題をすぐに検出し、テークオーバー操作を開始することができます。 HADR では、DBMS に個別のストレージが必要であるため、クラスター・マネージャーは、個別のボリューム制御で構成する必要があります。この構成により、クラスター・マネージャーは、スタンバイ・システムで DBMS を使用する前に、ボリュームでフェイルオーバーが発生することを待たずに済みます。自動クライアント転送機能を使用して、クライアント・アプリケーションを新しい 1 次データベースへリダイレクトすることができます。
- 1 次データベースとスタンバイ・データベースが同じクラスター・マネージャーによって処理されない HADR ペアを設定します。

  この構成は、1 次データベースとスタンバイ・データベースが別々のサイトに存在し、完全なサイト障害時の災害時リカバリーに高可用性が求められる環境に最適です。この構成をインプリメントする方法はいくつかあります。 HADR 1 次データベースおよびスタンバイ・データベースがクラスターの一部であるときには、 2 つのフェイルオーバー・シナリオが考えられます。
  - 部分サイト障害が発生する場合で、DBMS がフェイルオーバーできるノードがまだ使用可能な場合、クラスターのフェイルオーバーを選択できます。この場合、IP アドレスとボリュームのフェイルオーバーは、クラスター・マネージャーを使用して実行されます。 HADR は影響を受けません。
  - 1 次データベースが存在するサイトで完全なサイト障害が発生する場合、HADR を使用し、テークオーバー操作を開始することで、DBMS 可用性を維持できます。スタンバイ・データベースが存在するサイトで完全なサイト障害が発生する場合、サイトを修復するか、スタンバイ・データベースを別のサイトへ移動できます。

関連概念:

- 201 ページの『高可用性災害時リカバリーの概要』
- 214 ページの『自動クライアント転送および高可用性災害時リカバリー』

関連タスク:

- 230 ページの『高可用性災害時リカバリーでのデータベース役割の切り替え』

高可用性災害時リカバリーの初期設定

1 次データベースとスタンバイ・データベースを高可用性災害時リカバリー (HADR) 用にセットアップして初期設定するには、以下の手順を使用します。

手順:

HADR は、コマンド行プロセッサー (CLP)、コントロール・センターの「高可用性災害時リカバリー (HADR) データベースのセットアップ」ウィザードを使用するか、または対応するアプリケーション・プログラミング・インターフェース (API) を呼び出して初期設定できます。

CLP を使用して、初めてシステム上で HADR を初期設定する場合、次のようにします。

1. 各 HADR データベースのホスト名、ホスト IP アドレス、およびサービス名かポート番号を判別します。

   ホストに複数のネットワーク・インターフェースがある場合、 HADR ホスト名または IP アドレスが目的のインターフェースにマッピングされるようにします。ホスト名は、1 つの IP アドレスにだけマッピング可能です。

   注: 1 次データベースとスタンバイ・データベースのインスタンス名は、同じである必要はありません。
2. 1 次データベースとして設定する予定の既存のデータベースに基づき、バックアップ・イメージをリストアするか、スプリット・ミラーを初期設定して、スタンバイ・データベースを作成します。

   次の例では、BACKUP DATABASE および RESTORE DATABASE コマンドが使用され、データベース SOCKS がスタンバイ・データベースとして初期設定されます。この場合、NFS がマウントされたファイル・システムは、両方のサイトでアクセス可能です。

   1 次データベースで次のコマンドを発行します。

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

   スタンバイ・データベースで次のコマンドを発行します。

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

   次の例には、**db2inidb** ユーティリティーを使用して、 1 次データベースのスプリット・ミラーを使用したスタンバイ・データベースを初期設定する方法が示されています。この手順は、前述のバックアップおよびリストアの手順の代わりに実行できるものです。

   スタンバイ・データベースで次のコマンドを発行します。

   ```
   db2inidb socks as standby
   ```

   注:

   a. 1 次データベースとスタンバイ・データベースのデータベース名は、同じでなければなりません。

b. リストア操作後またはスプリット・ミラー初期設定後は、スタンバイ・データベースで ROLLFORWARD DATABASE コマンドを発行しないことをお勧めします。ロールフォワード操作を使用したときの結果は、スタンバイ・データベースで HADR を使用してログを再生する場合とは、若干異なる場合があります。データベースが同一でない場合、AS STANDBY オプションを指定した START HADR コマンドを発行すると失敗します。

c. RESTORE DATABASE コマンドを使用するときに、REPLACE HISTORY FILE オプションを使用することをお勧めします。

d. スタンバイ・データベースを設定するときには、以下の RESTORE DATABASE コマンド・オプションを避ける必要があります。TABLESPACE、INTO、REDIRECT、および WITHOUT ROLLING FORWARD。

e. **db2inidb** ユーティリティーを使用してスタンバイ・データベースを設定する場合、 SNAPSHOT または MIRROR オプションは使用しないでください。構成属性である、インスタンス名、ログ・パス、およびデータベース・パスの 1 つ以上を変更するには、 RELOCATE USING オプションを指定できます。しかし、データベース名または表スペース・コンテナー・パスを変更してはなりません。

3. 1 次データベースおよびスタンバイ・データベースで、HADR 構成パラメーターを設定します。

   注: スタンバイ・データベースの作成後に、以下の構成パラメーターを設定することは非常に重要です。

   - HADR_LOCAL_HOST
   - HADR_LOCAL_SVC
   - HADR_REMOTE_HOST
   - HADR_REMOTE_SVC
   - HADR_REMOTE_INST

   これらを、スタンバイ・データベースの作成前に設定する場合、スタンバイ・データベース側での設定は、 1 次データベースでの設定内容を反映します。

4. スタンバイ・インスタンスに接続し、次の例のように、スタンバイ・データベースで HADR を開始します。

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

   注: 通常は、スタンバイ・データベースが最初に開始されます。1 次データベースを最初に開始する場合、スタンバイ・データベースが HADR_TIMEOUT データベース構成パラメーターで指定した時間間隔内に開始されなければ、この開始手順は失敗します。

5. 1 次インスタンスに接続し、次の例のように、1 次データベースで HADR を開始します。

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

6. これで、1 次データベースおよびスタンバイ・データベースで HADR が開始されました。

「高可用性災害時リカバリー (HADR) データベースのセットアップ」ウィザードをオープンするには、次のようにします。

1. コントロール・センターから、HADR を構成する対象のデータベースが見つかるまでオブジェクト・ツリーを展開します。
2. データベースを右クリックし、ポップアップ・メニューで、「高可用性災害時リカバリー」→「セットアップ」をクリックします。「高可用性災害時リカバリー (HADR) データベースのセットアップ」ウィザードがオープンします。

詳しい情報については、コントロール・センターのコンテキスト・ヘルプ機能をご覧ください。

関連概念:

- 204 ページの『高可用性災害時リカバリー用のデータベース構成』

関連タスク:

- 96 ページの『リストアの使用』
- 194 ページの『スプリット・ミラーをバックアップ・イメージとして使用する』

関連資料:

- 222 ページの『START HADR』
- 227 ページの『STOP HADR』

START HADR

データベースの HADR 操作を開始します。

権限:

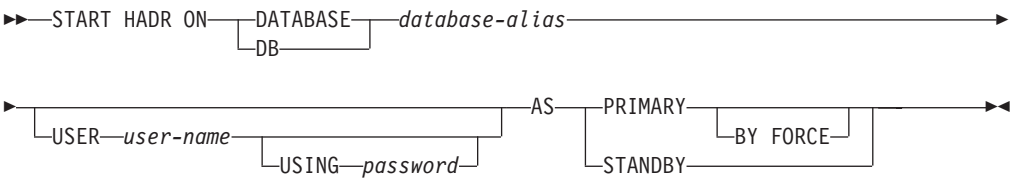
以下のいずれかが必要です。

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

必要な接続:

インスタンス。データベース接続が存在しない場合、このコマンドでデータベース接続が確立され、コマンド完了時にそのデータベース接続がクローズされます。

コマンド構文:



コマンド・パラメーター:

DATABASE *database-alias*
HADR 操作を開始するデータベース。

USER *user-name*
HADR 操作を開始するために使用するユーザー名。

USING *password*
user-name の認証パスワード。

AS PRIMARY
データベースに対して HADR 1 次操作を開始することを指定します。

BY FORCE
HADR 1 次データベースにおいて、スタンバイ・データベースがそれに接続するまで待機しないことを指定します。 BY FORCE オプションによる開始の後も 1 次データベースは、後でスタンバイ・データベースが利用可能になった時点で、スタンバイ・データベースからの有効な接続を受け付けます。

注意: START HADR コマンドに AS PRIMARY BY FORCE オプションを指定する場合には、十分な注意が必要です。スタンバイ・データベースが 1 次データベースに変更されてから、AS PRIMARY BY FORCE オプションを指定した START HADR コマンドの発行により元のデータベースが再始動した場合、データベースの 2 つのコピーが 1 次として独立して動作することになります。 (これは*分割ブレイン*、または*二重 1 次* と呼ばれることがあります)。この場合、各 1 次データベースは複数の接続を受け入れたり複数のトランザクションを実行したりできますが、もう一方のデータベースによる更新は受け取ることも再生することもできません。そのため、データベースのそれら 2 つのコピーは、互いに矛盾することになります。

AS STANDBY
データベースに対して HADR スタンバイ操作を開始することを指定します。スタンバイ・データベースは、接続が正常に確立されるまで、または接続試行が 1 次データベースによって明示的に拒否されるまで、 HADR 1 次データベースへの接続を試行します。 (1 次データベースが接続を拒否する場合として考えられるのは、 HADR 構成パラメーターが正しく設定されていない場合、またはデータベースのコピーが矛盾している場合であり、いずれにしても接続再試行を続けることは適当ではありません。)

使用上の注意:

さまざまな条件におけるデータベースの動作を、次の表に示します。

データベースの状況	START HADR コマンド (AS PRIMARY オプション) での動作	START HADR コマンド (AS STANDBY オプション) での動作
非アクティブ標準データベース	HADR 1 次データベースとして活動化されます。	データベースがロールフォワード・ペンディング・モード (リストアまたはスプリット・ミラーなどの結果)、またはロールフォワード進行中モードの場合、データベースはスタンバイ・データベースとして開始します。それ以外の場合、エラーが戻されます。

データベースの状況	START HADR コマンド (AS PRIMARY オプション) での動作	START HADR コマンド (AS STANDBY オプション) での動作
アクティブ標準データベース	データベースは HADR 1 次の役割になります。	エラー・メッセージが戻されます。
非アクティブ 1 次データベース	HADR 1 次データベースとして活動化されます。	フェイルオーバーの後、障害の発生した 1 次を新しいスタンバイ・データベースとして HADR ペアに再び組み入れます。いくらかの制限があります。
アクティブ 1 次データベース	警告メッセージが発行されます。	エラー・メッセージが戻されます。
非アクティブ・スタンバイ・データベース	エラー・メッセージが戻されます。	データベースをスタンバイ・データベースとして開始します。
アクティブ・スタンバイ・データベース	エラー・メッセージが戻されます。	警告メッセージが発行されます。

db2HADRStart - HADR の開始

データベースで HADR 操作を開始します。

権限:

以下のいずれかが必要です。

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

必要な接続:

インスタンス。データベース接続が存在しない場合、この API でデータベース接続が確立され、 API 完了時にそのデータベース接続がクローズされます。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2HADRStart */
/* ... */
SQL_API_RC SQL_API_FN
  db2HADRStart (
        db2Uint32 versionNumber,
        void * pParmStruct,
        struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRStartStruct
{
  char                          *piDbAlias;
  char                          *piUserName;
```

```
  char                        *piPassword;
  db2Uint32                   iDbRole;
  db2Uint16                   iByForce;
} db2HADRStartStruct;
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2gHADRStart */
/* ... */
SQL_API_RC SQL_API_FN
  db2gHADRStart (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRStartStruct
{
  char                        *piDbAlias;
  db2Uint32                   iAliasLen;
  char                        *piUserName;
  db2Uint32                   iUserNameLen;
  char                        *piPassword;
  db2Uint32                   iPasswordLen;
  db2Uint32                   iDbRole;
  db2Uint16                   iByForce;
} db2gHADRStartStruct;
```

API パラメーター:

versionNumber
入力。 2 番目のパラメーター *pParmStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pParmStruct
入力。 *db2HADRStartStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

piDbAlias
入力。データベース別名を指すポインター。

iAliasLen
入力。データベース別名の長さ (バイト単位) を指定します。

piUserName
入力。コマンドを実行するときのユーザー名へのポインター。

iUserNameLen
入力。ユーザー名の長さ (バイト単位) を指定します。

piPassword
入力。パスワードを含むストリングを指すポインター。

iPasswordLen
入力。パスワードの長さ (バイト単位) を指定します。

iDbRole
入力。データベースを HADR 1 次データベースとして開始するか、スタンバイ・データベースとして開始するかを指定します。有効な値は以下のとおりです。

DB2HADR_DB_ROLE_PRIMARY

DB2HADR_DB_ROLE_STANDBY

iByForce

入力。 *iDbRole* が DB2HADR_DB_ROLE_STANDBY に設定される場合、この引き数は無視されます。有効な値は以下のとおりです。

DB2HADR_NO_FORCE

スタンバイ・データベースが指定した時間制限内に 1 次データベースへ接続する場合にのみ、 HADR が 1 次データベースで開始されることを指定します。

DB2HADR_FORCE

スタンバイ・データベースが 1 次データベースへ接続することを待たずに、 HADR を強制的に開始することを指定します。

関連資料:

- 229 ページの『db2HADRStop - HADR の停止』
- 239 ページの『db2HADRTakeover - 1 次データベースとしてのテークオーバー』
- 222 ページの『START HADR』

高可用性災害時リカバリーの停止

1 次データベースまたはスタンバイ・データベースの高可用性災害時リカバリー (HADR) 操作を停止するには、 STOP HADR コマンドを使用します。片方または両方のデータベースで HADR を停止することを選択できます。スタンバイ・システムで保守を実行している場合、スタンバイ・データベースで HADR を停止するだけで構いません。 HADR の使用を完全に停止する場合、両方のデータベースで HADR を停止できます。

制限:

1 次データベースまたはスタンバイ・データベースのどちらか一方だけに STOP HADR コマンドを発行できます。標準のデータベースにこのコマンドを発行すると、エラーが戻されます。

手順:

HADR は、コマンド行プロセッサー (CLP)、コントロール・センターの「高可用性災害時リカバリー (HADR) の管理」ウィンドウ、または **db2HADRStop** アプリケーション・プログラミング・インターフェース (API) を使用して停止できます。

1 次データベースまたはスタンバイ・データベースで CLP を使用して HADR 操作を停止するには、 HADR 操作を停止するデータベースで STOP HADR コマンドを発行します。

次の例では、HADR 操作はデータベース SOCKS で停止します。

```
STOP HADR ON DATABASE SOCKS
```

このコマンドを非アクティブの 1 次データベースに対して発行する場合、データベースは標準データベースに切り替わり、オフラインのままになります。

このコマンドを非アクティブのスタンバイ・データベースに対して発行する場合、データベースは標準データベースに切り替わり、ロールフォワード・ペンディング状態となって、オフラインのままになります。

このコマンドをアクティブな 1 次データベースで発行する場合、スタンバイ・データベースへのログの送信が停止し、 1 次データベースで HADR エンジン・ディスパッチ可能単位 (EDU) がすべてシャットダウンされます。データベースは、標準データベースに切り替わり、オンラインのままになります。トランザクション処理は続行可能です。データベースの役割を 1 次データベースに戻す場合は、START HADR AS PRIMARY コマンドを発行できます。

このコマンドをアクティブなスタンバイ・データベースに対して発行する場合、エラー・メッセージが戻され、スタンバイ・データベースを標準データベースに変更する前にそれを非アクティブにする必要があることが示されます。

「高可用性災害時リカバリー (HADR) の管理」ウィンドウをオープンするには、次のようにします。

1. コントロール・センターから、HADR を管理する対象のデータベースが見つかるまでオブジェクト・ツリーを展開します。
2. データベースを右クリックし、ポップアップ・メニューで、「高可用性災害時リカバリー」→「管理」をクリックします。「高可用性災害時リカバリー (HADR) の管理」ウィンドウがオープンします。

詳しい情報については、コントロール・センターのコンテキスト・ヘルプ機能をご覧ください。

関連概念:

- 201 ページの『高可用性災害時リカバリーの概要』

関連資料:

- 222 ページの『START HADR』
- 227 ページの『STOP HADR』

STOP HADR

データベースの HADR 操作を停止します。

権限:

以下のいずれかが必要です。

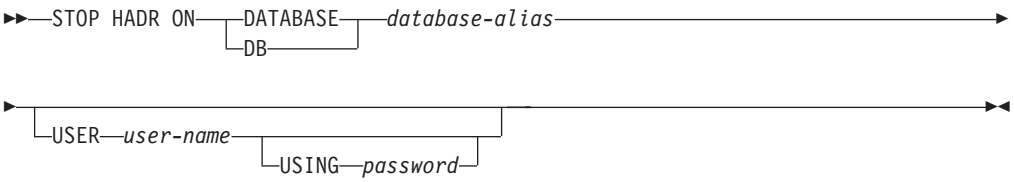
- *sysadm*
- *sysctrl*
- *sysmaint*

必要な接続:

インスタンス。データベース接続が存在しない場合、このコマンドでデータベース接続が確立され、コマンド完了時にそのデータベース接続がクローズされます。

コマンド構文:

STOP HADR



コマンド・パラメーター:

DATABASE *database-alias*
HADR 操作を停止するデータベース。

USER *user-name*
HADR 操作を停止するために使用するユーザー名。

USING *password*
user-name の認証パスワード。

使用上の注意:

さまざまな条件におけるデータベースの動作を、次の表に示します。

データベースの状況	STOP HADR コマンドでの動作
非アクティブ標準データベース	エラー・メッセージが戻されます。
アクティブ標準データベース	エラー・メッセージが戻されます。
非アクティブ 1 次データベース	データベースの役割は標準に変更されます。データベース構成パラメーター *hadr_db_role* は STANDARD に更新されます。データベースはオフラインのままです。次回再始動時に標準の役割になります。
アクティブ 1 次データベース	HADR スタンバイ・データベースへのログ出力を停止し、HADR 1 次データベース上のすべての HADR EDU をシャットダウンします。データベースの役割は標準に変更され、データベースはオンラインのままです。 AS PRIMARY オプションを指定した明示的な START HADR コマンドが発行されるまで、データベースは標準の役割のままです。STOP HADR コマンドは、オープンされているセッションとトランザクションには影響しません。データベースがオンラインのまま、 STOP HADR コマンドと START HADR コマンドを繰り返して発行できます。それらのコマンドは動的に影響を及ぼします。
非アクティブ・スタンバイ・データベース	データベースの役割は標準に変更されます。データベース構成パラメーター *hadr_db_role* は STANDARD に更新されます。データベースはオフラインのままです。データベースはロールフォワード・ペンディング・モードになります。
アクティブ・スタンバイ・データベース	エラー・メッセージが戻されます。スタンバイ・データベースを標準データベースに変更する前にそれを非アクティブにします。

db2HADRStop - HADR の停止

データベースで HADR 操作を停止します。

権限:

以下のいずれかが必要です。

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

必要な接続:

インスタンス。データベース接続が存在しない場合、この API でデータベース接続が確立され、 API 完了時にそのデータベース接続がクローズされます。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2HADRStop */
/* ... */
SQL_API_RC SQL_API_FN
  db2HADRStop (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRStopStruct
{
  char                        *piDbAlias;
  char                        *piUserName;
  char                        *piPassword;
} db2HADRStopStruct;
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2gHADRStop */
/* ... */
SQL_API_RC SQL_API_FN
  db2gHADRStop (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRStopStruct
{
  char                        *piDbAlias;
  db2Uint32                   iAliasLen;
  char                        *piUserName;
  db2Uint32                   iUserNameLen;
  char                        *piPassword;
  db2Uint32                   iPasswordLen;
} db2gHADRStopStruct;
```

API パラメーター:

versionNumber
入力。 2 番目のパラメーター *pParmStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pParmStruct
入力。 *db2HADRStopStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

piDbAlias
入力。データベース別名を指すポインター。

iAliasLen
入力。データベース別名の長さ (バイト単位) を指定します。

piUserName
入力。コマンドを実行するときのユーザー名へのポインター。

iUserNameLen
入力。ユーザー名の長さ (バイト単位) を指定します。

piPassword
入力。パスワードを含むストリングを指すポインター。

iPasswordLen
入力。パスワードの長さ (バイト単位) を指定します。

関連資料:

- 224 ページの『db2HADRStart - HADR の開始』
- 239 ページの『db2HADRTakeover - 1 次データベースとしてのテークオーバー』
- 227 ページの『STOP HADR』

高可用性災害時リカバリーでのデータベース役割の切り替え

高可用性災害時リカバリー (HADR) 時に、1 次データベースとスタンバイ・データベースの役割を切り替えるには、 TAKEOVER HADR コマンドを使用します。

警告: この手順を実行する前に、1 次データベースがデータベース・トランザクションを処理していないことを確認してください。

制限:

- TAKEOVER コマンドは、スタンバイ・データベース上でのみ発行できます。コマンドの発行時に 1 次データベースがスタンバイ・データベースに接続されていない場合、テークオーバー操作は失敗します。
- TAKEOVER HADR コマンドは、1 次データベースとスタンバイ・データベースがピア状態の場合にだけ、それらのデータベース間の役割を切り替えるために使用できます。スタンバイ・データベースが他の状態の場合、エラー・メッセージが戻されます。

手順:

HADR データベースの役割は、コマンド行プロセッサー (CLP)、コントロール・センターの「高可用性災害時リカバリー (HADR) の管理」ウィンドウ、

または **db2HADRTakeover** アプリケーション・プログラミング・インターフェース (API) を使用して、切り替えることができます。

CLP を使用して、スタンバイ・データベースでテークオーバー操作を開始するには、スタンバイ・データベース上で BY FORCE オプションを指定しない TAKEOVER HADR コマンドを発行します。

次の例では、スタンバイ・データベース LEAFS でテークオーバー操作が行われます。

```
TAKEOVER HADR ON DB LEAFS
```

「高可用性災害時リカバリー (HADR) の管理」ウィンドウをオープンするには、次のようにします。

1. コントロール・センターから、HADR を管理する対象のデータベースが見つかるまでオブジェクト・ツリーを展開します。
2. データベースを右クリックし、ポップアップ・メニューで、「高可用性災害時リカバリー」→「管理」をクリックします。「高可用性災害時リカバリー (HADR) の管理」ウィンドウがオープンします。

詳しい情報については、コントロール・センターのコンテキスト・ヘルプ機能をご覧ください。

関連概念:

- 201 ページの『高可用性災害時リカバリーの概要』
- 208 ページの『高可用性災害時リカバリーでのスタンバイ・データベースの状態』

関連資料:

- 237 ページの『TAKEOVER HADR』

フェイルオーバー時の HADR テークオーバーの使用

現在の 1 次データベースが使用可能でないために、現在のスタンバイ・データベースを新しい 1 次データベースにすることを希望する場合、フェイルオーバーを実行することができます。

警告: この手順では、データが消失する可能性があります。この非常手順を実行する前に、次の情報を検討してください。

- 1 次データベースがデータベース・トランザクションを処理していないことを確認してください。 1 次データベースが実行中だが、スタンバイ・データベースと通信できない場合、(BY FORCE オプションを指定した TAKEOVER HADR コマンドを発行して) 強制的にテークオーバー操作を実行すると、 1 次データベースが 2 つできる可能性があります。 1 次データベースが 2 つ存在するときには、それぞれのデータベースには異なるデータが存在し、これらの 2 つのデータベースが自動的に同期化されることはなくなります。
  - 1 次データベースを非活動化するか、可能であれば、そのインスタンスを停止してください。 (1 次システムが、ハング、破損、またはアクセス不能である場合、これは不可能である可能性があります。) テークオ

ーバー操作の実行後、障害が発生したデータベースが後で再始動される場合、自動的に 1 次データベースの役割であると見なされることはありません。

- トランザクション消失の可能性と程度は、それぞれ特定の構成および環境に応じて異なります。
  - ピア状態のときに 1 次データベースに障害が発生する場合で、同期モードが同期 (SYNC) である場合、スタンバイ・データベースは、 1 次データベースの障害発生前にアプリケーションへコミットされたと報告のあったトランザクションを消失することはありません。
  - ピア状態のときに 1 次データベースに障害が発生する場合で、同期モードが準同期 (NEARSYNC) である場合、スタンバイ・データベースは、1 次データベースとスタンバイ・データベースの両方に同時に障害が発生する場合に、1 次データベースによってコミットされたトランザクションだけを消失する可能性があります。
  - ピア状態のときに 1 次データベースに障害が発生する場合で、同期モードが非同期 (ASYNC) である場合、スタンバイ・データベースは、スタンバイ・データベースがテークオーバー操作の実行前にトランザクションの全ログ・レコードを受け取らなかった場合に、 1 次データベースによってコミットされたトランザクションを消失する可能性があります。スタンバイ・データベースは、1 次データベースとスタンバイ・データベースの両方に同時に障害が発生する場合に、 1 次データベースによってコミットされたトランザクションも消失する可能性があります。
  - リモート・キャッチアップ・ペンディング状態のときに 1 次データベースに障害が発生する場合、スタンバイ・データベースが受け取って処理していないトランザクションは消失します。

    注: データベース・スナップショットにログのギャップが示される場合、それは、 1 次データベースとスタンバイ・データベースが最後に相互に通信した時点でのギャップです。 1 次データベースは、その時点以降、非常に大量のトランザクションを処理した可能性があります。

制限:

- TAKEOVER HADR コマンドは、スタンバイ・データベース上でのみ発行できます。
- HADR は、障害が発生したデータベースを自動的に再始動する際に使用できる、DB2® 障害モニター (db2fm) とのインターフェースはありません。障害モニターが使用可能な場合、障害が発生したと思われる 1 次データベースでの、行われる可能性のある障害モニター・アクションに注意する必要があります。
- テークオーバー操作は、1 次データベースとスタンバイ・データベースがピア状態であるか、スタンバイ・データベースがリモート・キャッチアップ・ペンディング状態の場合のみ行えます。スタンバイ・データベースが他の状態である場合、エラーが戻されます。

  注: ローカル・キャッチアップ状態のスタンバイ・データベースを、標準データベースに変換することにより、通常の使用で使用可能にすることができます。このためには、DEACTIVATE DATABASE コマンドを発行してデータベースをシャットダウンしてから、 STOP HADR コマンドを発行します。

HADR が停止したら、以前のスタンバイ・データベースを使用可能にする前に、以前のスタンバイ・データベースでロールフォワード操作を完了する必要があります。データベースをスタンバイ・データベースから標準データベースへ変換した後は、そのデータベースを HADR ペアに再統合することはできません。 2 つのサーバーで HADR を再始動するには、HADR を初期設定するための次の手順に従ってください。

手順:

フェイルオーバー・シナリオでは、テークオーバー操作は、コマンド行プロセッサー (CLP)、コントロール・センターの「高可用性災害時リカバリーの管理」ウィンドウ、または **db2HADRTakeover** アプリケーション・プログラミング・インターフェース (API) を使用して実行できます。

次の手順では、CLP を使用して、1 次データベースまたはスタンバイ・データベースでフェイルオーバーを開始する方法を示します。

1. 障害が発生した 1 次データベースを完全に使用不可にします。データベースに内部エラーが生じる場合、通常シャットダウン・コマンドでは、 1 次データベースを完全にシャットダウンできません。プロセス、共用メモリー、またはネットワーク接続などのリソースを除去するには、オペレーティング・システム・コマンドを使用しなければならない可能性があります。
2. スタンバイ・データベースで BY FORCE オプションを指定した TAKEOVER HADR コマンドを発行します。次の例では、フェイルオーバーはデータベース LEAFS で行われます。

   ```
   TAKEOVER HADR ON DB LEAFS BY FORCE
   ```

   1 次データベースはオフラインになるものと予想されるため、 BY FORCE オプションが必要になります。

   1 次データベースを完全に使用不可にしない場合、スタンバイ・データベースは、1 次データベースに接続されたままになり、シャットダウンするようにというメッセージを 1 次データベースに送信します。スタンバイ・データベースは、1 次データベースがシャットダウンされたことを示す確認を受け取るかどうかにかかわらず、 1 次データベースの役割に切り替えられます。

「高可用性災害時リカバリー (HADR) の管理」ウィンドウをオープンするには、次のようにします。

1. コントロール・センターから、HADR を管理する対象のデータベースが見つかるまでオブジェクト・ツリーを展開します。
2. データベースを右クリックし、ポップアップ・メニューで、「高可用性災害時リカバリー」→「管理」をクリックします。「高可用性災害時リカバリーの管理」ウィンドウがオープンします。

詳しい情報については、コントロール・センターのオンライン・ヘルプ機能をご覧ください。

関連概念:

- 201 ページの『高可用性災害時リカバリーの概要』
- 211 ページの『高可用性災害時リカバリーの同期モード』

- 208 ページの『高可用性災害時リカバリーでのスタンバイ・データベースの状態』

関連タスク:

- 220 ページの『高可用性災害時リカバリーの初期設定』
- 230 ページの『高可用性災害時リカバリーでのデータベース役割の切り替え』

関連資料:

- 143 ページの『ROLLFORWARD DATABASE』
- 「コマンド・リファレンス」の『DEACTIVATE DATABASE コマンド』
- 227 ページの『STOP HADR』
- 237 ページの『TAKEOVER HADR』

テークオーバー操作後のデータベースの再統合

1 次データベースに障害が発生したために、テークオーバー操作を高可用性災害時リカバリー (HADR) 環境で実行した場合、障害の発生したデータベースをオンラインに戻してスタンバイ・データベースとして使用するか、1 次データベースとしての状況に戻すことができます、

障害の発生した 1 次データベースを新しいスタンバイ・データベースとして HADR ペアに再統合するには、次のようにします。

1. 元の 1 次データベースが存在したシステムを修復します。このことは、破損したハードウェアを修復することや、障害の発生したオペレーティング・システムをリブートすることを意味する場合があります。
2. 障害の発生した 1 次データベースをスタンバイ・データベースとして再始動します。次の例では、データベース LEAFS がスタンバイ・データベースとして開始されます。

   ```
   START HADR ON DB LEAFS AS STANDBY
   ```

   注: データベースの 2 つのコピーが非互換のログ・ストリームを持つ場合には、このコマンドは失敗します。特に、HADR は、元のスタンバイ・データベースが新しい 1 次データベースになるまでは、元の 1 次データベースが、ログに記録されていても元のスタンバイ・データベースで反映されていない操作を適用しないように要求します。この状況が生じると、新しい 1 次データベースのバックアップ・イメージをリストアするか、スプリット・ミラーを初期設定することにより、元の 1 次データベースをスタンバイ・データベースとして再始動できます。

元の 1 次データベースがスタンバイ・データベースとして HADR ペアに再結合されたら、フェイルバック操作を実行することを選択し、データベースの役割を切り替えて、元の 1 次データベースをもう一度 1 次データベースにすることができます。このフェイルバック操作を実行するには、スタンバイ・データベースで次のコマンドを発行します。

```
TAKEOVER HADR ON DB LEAFS
```

注:

1. HADR データベースがピア状態ではないか、ペアが接続されていない場合、このコマンドは失敗します。
2. 1 次データベースのオープン・セッションは強制的にクローズされ、処理中のトランザクションはロールバックされます。
3. 1 次データベースの役割とスタンバイ・データベースの役割を切り替える場合、TAKEOVER HADR コマンドの BY FORCE オプションは指定できません。

関連概念:

- 211 ページの『高可用性災害時リカバリーの同期モード』
- 208 ページの『高可用性災害時リカバリーでのスタンバイ・データベースの状態』

関連タスク:

- 230 ページの『高可用性災害時リカバリーでのデータベース役割の切り替え』

関連資料:

- 237 ページの『TAKEOVER HADR』

高可用性災害時リカバリー環境での回転アップグレードの実行

ソフトウェア (オペレーティング・システムまたは DB2® UDB) またはハードウェアをアップグレードするとき、またはデータベース構成パラメーターに変更を加えるときには、高可用性災害時リカバリー (HADR) 環境でこの手順を使用します。この手順を使用すると、アップグレード・プロセスの間、データベース・サービスはずっと使用可能な状態になります。ただし、処理があるデータベースから別のデータベースへ切り替えられるときには、そのときだけ一時的にサービスが中断します。 1 次データベースとスタンバイ・データベースの両方が同等のシステムにある場合に、HADR が適切に実行されるため、できるだけ迅速に、変更を両方のシステムに適用する必要があります。

注: DB2 UDB FixPak およびアップグレードはすべて、実動システムに適用する前に、テスト環境にインプリメントする必要があります。

前提条件:

アップグレードを開始する前に、HADR ペアをピア状態にしておく必要があります。

制限:

この手順は、DB2 UDB のメジャー・バージョンからそれ以降のバージョンにアップグレードするときには機能しません。

手順:

HADR 環境でアップグレードを実行するには、次のようにします。

1. スタンバイ・データベースが存在するシステムをアップグレードします。
   a. DEACTIVATE DATABASE コマンドを使用して、スタンバイ・データベースをシャットダウンします。

b. 必要な場合、スタンバイ・データベースのインスタンスをシャットダウンします。
c. ソフトウェア、ハードウェア、または DB2 構成パラメーターの 1 つ以上を変更します。
d. 必要な場合、スタンバイ・データベースのインスタンスを再始動します。
e. ACTIVATE DATABASE コマンドを使用して、スタンバイ・データベースを再始動します。
f. スタンバイ・データベースがピア状態になったことを確認します。このことを確認するには、GET SNAPSHOT コマンドを使用します。

2. 1 次データベースの役割とスタンバイ・データベースの役割を切り替えます。
   a. スタンバイ・データベースで TAKEOVER HADR コマンドを発行します。
   b. クライアントを新しい 1 次データベースに誘導します。これは自動クライアント転送を使用して実行できます。

   注: スタンバイ・データベースが 1 次データベースとしてテークオーバーするため、ここで新しい 1 次データベースがアップグレードされます。 DB2 UDB FixPak を適用している場合、TAKEOVER HADR コマンドを発行すると、元の 1 次データベースの役割がスタンバイ・データベースに変更されます。しかし、このコマンドでは、新しいスタンバイ・データベースは新しくアップグレードした 1 次データベースに接続しません。新しいスタンバイ・データベースは以前のバージョンの DB2 UDB を使用するため、アップグレードされた 1 次データベースによって生成される新しいログ・レコードが理解されない場合があり、その場合にはシャットダウンしてしまいます。 1 次データベースに接続するには、スタンバイ・データベースもアップグレードする必要があります。

3. 前述のステップ 1 と同じ手順を使用して、元の 1 次データベース (つまり、現在のスタンバイ・データベース) をアップグレードします。これが完了したら、両方のデータベースがアップグレードされ、 HADR ピア状態で相互に接続されます。 HADR システムには、完全なデータベース・サービスと、完全な高可用性保護機能が備えられています。
4. オプション。元の構成に戻すには、ステップ 2 のように、1 次データベースの役割とスタンバイ・データベースの役割を切り替えます。

関連概念:

- 214 ページの『自動クライアント転送および高可用性災害時リカバリー』
- 208 ページの『高可用性災害時リカバリーでのスタンバイ・データベースの状態』

関連資料:

- 「コマンド・リファレンス」の『GET SNAPSHOT コマンド』
- 「コマンド・リファレンス」の『ACTIVATE DATABASE コマンド』
- 「コマンド・リファレンス」の『DEACTIVATE DATABASE コマンド』
- 237 ページの『TAKEOVER HADR』

TAKEOVER HADR

HADR スタンバイ・データベースに対して、 HADR ペアのための新しい HADR 1 次データベースとなるように指示します。

権限:

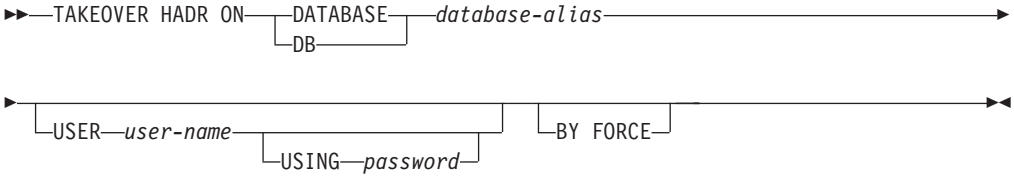
以下のいずれかが必要です。

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

必要な接続:

インスタンス。データベース接続が存在しない場合、このコマンドでデータベース接続が確立され、コマンド完了時にそのデータベース接続がクローズされます。

コマンド構文:



コマンド・パラメーター:

DATABASE *database-alias*
HADR 1 次データベースにする現在の HADR スタンバイ・データベース。

USER *user-name*
テークオーバー操作を開始するために使用するユーザー名。

USING *password*
user-name の認証パスワード。

BY FORCE
データベースが、元の HADR 1 次データベースがシャットダウンされたという確認を待たないことを指定します。 HADR ペアがピア状態でない場合には、このオプションが必要です。

使用上の注意:

TAKEOVER HADR コマンドをアクティブ・スタンバイに対して発行した場合に、可能性のある状態とオプションの各組み合わせごとにどんな動作になるかを、次の表に示します。このコマンドを非アクティブ・スタンバイ・データベースに対して発行すると、エラー・メッセージが戻されます。

スタンバイ状態	BY FORCE オプション	テークオーバーの動作
ローカル・キャッチアップまたはリモート・キャッチアップ	使用しない	エラー・メッセージが戻されます。
ローカル・キャッチアップまたはリモート・キャッチアップ	使用する	エラー・メッセージが戻されます。
ピア	使用しない	1 次データベースとスタンバイ・データベースがその役割を切り替えます。 テークオーバー中に障害が発生しなければ、データの損失はありません。しかし、テークオーバー中に障害が発生すると、データ損失の可能性があります。 1 次とスタンバイの役割は変更される場合と変更されない場合があります。テークオーバー中に障害が発生して、1 次とスタンバイの役割が切り替えられた場合には、以下のガイドラインに従ってその障害を処理してください。 1. テークオーバー中に障害が発生した場合、 HADR システムの役割は変更される場合と変更されない場合があります。可能なら、2 つのデータベースがオンラインであることを確認してください。使用可能なデータベースの HADR 役割を調べるには、スナップショット・モニターを使用するか、またはデータベース構成パラメーター *hadr_db_role* の値を確認してください。 2. 意図された新しい 1 次データベースがまだスタンバイの役割であり、それでもテークオーバーを発行したい場合は、TAKEOVER HADR コマンドを再発行してください (BY FORCE オプションについての下記のガイドラインを参照)。 3. 2 つのデータベースが両方共スタンバイの役割のまま終わることもあり得ます。その場合、1 次になっているほうのノードで BY FORCE オプションを発行できます。その場合には、 2 つのスタンバイ・データベースが通常の HADR 1 次スタンバイ接続を確立することはできないため、 BY FORCE オプションが必要です。
ピア	使用する	スタンバイ側は 1 次側に対して、それ自体 (1 次) をシャットダウンするよう通知を送ります。スタンバイ側は 1 次側からのログ受信を停止し、既に受信したログの再生を終了してから、1 次になります。スタンバイ側は、 1 次側がテークオーバー通知を受け取ったこと、またはそれがシャットダウンされたことを確認するために、 1 次側から確認通知が送られるのを待ちません。

スタンバイ状態	BY FORCE オプション	テークオーバーの動作
リモート・キャッチアップ・ペンディング	使用しない	エラー・メッセージが戻されます。
リモート・キャッチアップ・ペンディング	使用する	スタンバイ・データベースが 1 次データベースになります。

db2HADRTakeover - 1 次データベースとしてのテークオーバー

スタンバイ・データベースが 1 次データベースとしてテークオーバーすることを指示します。

権限:

以下のいずれかが必要です。

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

必要な接続:

インスタンス。データベース接続が存在しない場合、この API でデータベース接続が確立され、 API 完了時にそのデータベース接続がクローズされます。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2HADRTakeover */
/* ... */
SQL_API_RC SQL_API_FN
  db2HADRTakeover (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRTakeoverStruct
{
  char                        *piDbAlias;
  char                        *piUserName;
  char                        *piPassword;
  db2Uint16                   iByForce;
} db2HADRTakeoverStruct;
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2gHADRTakeover */
/* ... */
SQL_API_RC SQL_API_FN
  db2gHADRTakeover (
```

```
        db2Uint32 versionNumber,
        void * pParmStruct,
        struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRTakeoverStruct
{
  char                        *piDbAlias;
  db2Uint32                   iAliasLen;
  char                        *piUserName;
  db2Uint32                   iUserNameLen;
  char                        *piPassword;
  db2Uint32                   iPasswordLen;
  db2Uint16                   iByForce;
} db2gHADRTakeoverStruct;
```

API パラメーター:

versionNumber
入力。 2 番目のパラメーター *pParmStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pParmStruct
入力。 *db2HADRStartStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

piDbAlias
入力。データベース別名を指すポインター。

iAliasLen
入力。データベース別名の長さ (バイト単位) を指定します。

piUserName
入力。コマンドを実行するときのユーザー名へのポインター。

iUserNameLen
入力。ユーザー名の長さ (バイト単位) を指定します。

piPassword
入力。パスワードを含むストリングを指すポインター。

iPasswordLen
入力。パスワードの長さ (バイト単位) を指定します。

iByForce
入力。有効な値は以下のとおりです。

DB2HADR_NO_FORCE
2 つのシステムが、通信の確立された対等状態である場合にのみ、テークオーバーが発生することを指定します。これにより、HADR 1 次データベースと HADR スタンバイ・データベースとの間で役割の逆転が生じます。

DB2HADR_FORCE
スタンバイ・データベースが、元の 1 次データベースがシャットダウンされたことの確認を待たずに、 1 次データベースとしてテークオーバーすることを指定します。

関連資料:

- 224 ページの『db2HADRStart - HADR の開始』
- 229 ページの『db2HADRStop - HADR の停止』
- 237 ページの『TAKEOVER HADR』

第 8 章 AIX でのクラスター・サポート

High Availability Cluster Multi-Processing のサポート

拡張スケーラビリティー (ES) は、 High Availability Cluster Multi-Processing (HACMP) for AIX® の機能です。この機能は、フェイルオーバー・リカバリーを提供し、HACMP と同じイベント構造を持ちます。また、拡張スケーラビリティーは以下のものを提供します。

- より大きな HACMP クラスター。
- ユーザー定義イベント による追加エラー保守。モニター・エリアでユーザー定義イベントが生成される可能性があり、その場合には処理が停止したり、ページ・スペースが容量の限界に近づいたりすることがあります。そのようなイベントにはイベント前とイベント後があり、必要に応じて、フェイルオーバー・リカバリー処理に追加できます。 HACMP のイベント前ストリームとイベント後ストリームの中で、異なるインプリメンテーションに特有の追加機能を加えることができます。

  規則ファイル (`/usr/sbin/cluster/events/rules.hacmprd`) には、 HACMP イベントが含まれています。ユーザー定義イベントはこのファイルに追加されます。その定義の一部として、イベント発生時に実行されるスクリプト・ファイルがあります。
- HACMP クライアント・ユーティリティー。 HACMP クラスター外の AIX 物理ノードから (1 つまたは複数のクラスターにおける) 状況変更をモニターし、検出します。

HACMP ES クラスター内のノードは、*ハートビート* またはキープアライブ・パケット というメッセージを交換して、自身の可用性についての情報を他のノードに通知します。応答しなくなったノードがあると、クラスター内の残りのノードはリカバリーを呼び出します。リカバリー処理は *node_down イベント* と呼ばれ、*フェイルオーバー* ということもあります。リカバリー処理が完了すると、ノードをクラスターに再統合されます。この処理を *node_up イベント* といいます。

イベントには 2 つのタイプ、つまり、HACMP ES の操作中に予期される標準イベントと、ハードウェアおよびソフトウェア・コンポーネントのモニターに関連したユーザー定義イベントがあります。

標準イベントの 1 つに、node_down イベントがあります。リカバリー処理の一部として実行する処理を計画する場合、 HACMP では 2 つのフェイルオーバー・オプション、つまり「ホット (またはアイドル) スタンバイ」と「相互テークオーバー」を指定できます。

注: HACMP を使用している時は、**db2iauto** ユーティリティーを以下のように使用して、 DB2® インスタンスがブート時に開始されないようにしてください。

```
db2iauto -off InstName
```

ここで

InstName は、インスタンスのログイン名。

Cluster Configuration

ホット・スタンバイ 構成では、テークオーバー・ノードではない AIX プロセッサー・ノードは他のワークロードを一切実行していません。*相互テークオーバー* 構成では、引き受けノードである AIX プロセッサー・ノードは他のワークロードを実行しています。

一般的には、パーティション・データベース環境では、 DB2 Universal Database は、パーティションが各ノードにある状態で相互テークオーバー・モードで実行されます。 1 つの例外として、カタログ・ノード部品がホット・スタンバイ構成の一部となるシナリオがあります。

HACMP ES を使用する RS/6000® SP™ で大規模な DB2 インストールを計画する場合は、 RS/6000 SP フレームの内部またはフレーム間でクラスターのノードを分割する方法を考慮する必要があります。異なる SP フレームにノードおよびそのバックアップを指定すると、 1 つのフレームがダウン (つまり、フレームの電源/スイッチ・ボードが故障) するというイベントでも引き受けが可能です。しかし、こうした故障は非常にまれだと考えられます。各 SP フレームには *N* +1 の電源機構があり、各 SP スイッチには *N* +1 のファンと電源を含む冗長パスがあるからです。フレームが故障すると、手作業で介入して、残りのフレームをリカバリーしなければならない場合があります。このリカバリー手順については、SP 管理ガイドで説明されています。 HACMP ES では、SP ノード障害をリカバリーできます。その場合、フレーム障害のリカバリーは、 1 つまたは複数の SP フレーム内のクラスターの適正なレイアウトに依存しています。

計画時に考慮するべき別の点として、大きなクラスターの管理方法があります。大きなクラスターよりも小さなクラスターのほうが管理は容易ですが、多くの小さなクラスターよりも 1 つの大きなクラスターのほうが管理はやはり容易です。計画時には、実際のアプリケーションがクラスター環境で使用される方法を考慮してください。たとえば、16 ノードで単一の大きな同類のアプリケーションが実行されていれば、構成を 8 個の 2 ノード・クラスターではなく、単一のクラスターとして管理するほうが容易でしょう。同じ 16 ノードでも、異なるネットワーク、ディスク、およびノード関係を持つ多数の異なるアプリケーションが含まれているのであれば、ノードを小さめのクラスターにグループ化したほうが得策と思われます。各ノードは一度に 1 つずつ HACMP クラスターに統合される点に留意してください。つまり、1 つの大きなクラスターよりも複数のクラスター構成のほうが始動速度は向上します。 HACMP ES は、ノードおよびそのバックアップが同じクラスターにある限り、単一のクラスターも複数のクラスターもサポートします。

HACMP ES フェイルオーバー・リカバリーでは、物理ノードにリソース・グループを事前定義 (*カスケード* ともいう) によって割り当てることができます。フェイルオーバーのリカバリー手順では、物理ノードにリソース・グループを浮動 (*回転* ともいう) によって割り当てることもできます。 IP アドレス、外部ディスク・ボリューム・グループ、ファイル・システム、 NFS ファイル・システム、および各リソース・グループ内のアプリケーション・サーバーは、アプリケーションまたはアプリケーション・コンポーネントのいずれかを指定します。これは、フェイルオーバーまたは再統合によって、物理ノード間で HACMP ES が操作するものです。フェ

イルオーバーおよび再統合の操作は、作成されるリソース・グループのタイプ、およびリソース・グループに入れられるノード数によって指定されます。

たとえば、DB2 データベース・パーティション (論理ノード) を考慮してみます。そのログと表スペースのコンテナーが外部ディスクに置かれ、他のノードがそのディスクにリンクされていた場合、それら他のノードは外部ディスクにアクセスし、(引き受けノード上にある) データベース・パーティションを再始動することができます。 HACMP では、このような操作が自動化されます。 HACMP ES は、 DB2 インスタンスの主要なユーザー・ディレクトリーが使用する NFS ファイル・システムをリカバリーする場合にも使用できます。

パーティション・データベース環境で DB2 UDB のリカバリーを計画する場合には、その一環として HACMP ES 資料を精読してください。概念、計画、インストール、管理の手引きに目を通した後、環境に合ったリカバリー・アーキテクチャーを構築してください。知られている障害点に基づいて識別した、リカバリーを要する各サブシステムについては、必要な HACMP クラスターとリカバリー・ノード (ホット・スタンバイまたは相互引き受け) を識別してください。

ディスクとアダプターはどちらも、外部ディスク構成でミラーリングすることをぜひお勧めします。 HACMP 用に構成する DB2 物理ノードの場合、ボリューム・グループ上のノードを共有外部ディスクから構成変更できるように配慮する必要があります。相互引き受け構成でこのような設定を行う場合、対になったノードが競合することなく互いのボリューム・グループにアクセスできるよう、計画を立てることが必要です。パーティション・データベース環境では、これは、すべてのコンテナー名が、全データベースをまたがってユニークでなければならないことを意味します。

固有のものにする 1 つの方法は、名前の一部にパーティション番号を含めることです。 SMS または DMS コンテナーを作成するときに、コンテナーのストリング構文にノード式を指定できます。式を指定するときは、ノード番号をコンテナー名の一部とすることができます。また、追加の引き数を指定する場合は、これらの引き数の結果をコンテナー名の一部とすることができます。ノード式を指定するには、引き数「 $N」([ブランク]$N) を使用します。引き数は必ずコンテナー・ストリングの最後に指定するようにし、以下のいずれかの形式だけを使用できます。

表 1. コンテナーを作成するための引き数 : ノード番号を 5 と仮定します。

構文	例	値
[ブランク]$N	「 $N」	5
[ブランク]$N+[番号]	「 $N+1011」	1016
[ブランク]$N%[番号]	「 $N%3」	2
[ブランク]$N+[番号]%[番号]	「 $N+12%13」	4
[ブランク]$N%[番号]+[番号]	「 $N%3+20」	22
注: 1. % はモジュラスです。 2. どの場合でも、演算子は左から右に向かって評価されます。		

次に、この特殊な引き数を使用してコンテナーを作成する方法の例をいくつか示します。

- 2 ノード・システムで使用するためのコンテナーの作成。

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
   (device '/dev/rcont $N' 20000)
```

  次のコンテナーが使用されます。

```
/dev/rcont0   - on Node 0
/dev/rcont1   - on Node 1
```

- 4 ノード・システムで使用するためのコンテナーの作成。

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
   (file '/DB2/containers/TS2/container $N+100' 10000)
```

  次のコンテナーが使用されます。

```
/DB2/containers/TS2/container100   - on Node 0
/DB2/containers/TS2/container101   - on Node 1
/DB2/containers/TS2/container102   - on Node 2
/DB2/containers/TS2/container103   - on Node 3
```

- 2 ノード・システムで使用するためのコンテナーの作成。

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
   ('/TS3/cont $N%2, '/TS3/cont $N%2+2')
```

  次のコンテナーが使用されます。

```
/TS3/cont0   - on Node 0
/TS3/cont2   - on Node 0
/TS3/cont1   - on Node 1
/TS3/cont3   - on Node 1
```

HACMP ES 用 DB2 データベース・パーティションの構成

構成が済むと、インスタンス中の各データベース・パーティションは HACMP ES によって物理ノードごとに始動されます。 4 ノードよりも大きい並列 DB2 構成を始動する場合は、複数のクラスターをお勧めします。 64 ノードの並列 DB2 構成では、 4 個の 16 ノード・クラスターよりも、 32 個の 2 ノード HACMP クラスターのほうが始動速度が向上することに注意してください。

スクリプト・ファイル rc.db2pe は、ホット・スタンバイまたは相互引き受けノードのいずれかで HACMP ES のフェイルオーバーまたはリカバリーを構成するための補助機構として、 DB2 UDB Enterprise Server Edition にパッケージされています (/usr/bin の各ノードにインストールされます)。また、rc.db2pe の内部から、フェイルオーバーまたは相互引き受け構成で DB2 バッファー・プールのサイズをカスタマイズできます。 (バッファー・プール・サイズは、1 つの物理ノードで 2 つのデータベース・パーティションを稼働する場合に適切なリソース割り振りを確保できるように構成できます。)

HACMP ES イベント・モニターおよびユーザー定義イベント

特定のノードでプロセスが止まってしまう場合にフェイルオーバー操作を開始することも、ユーザー定義イベントの 1 例です。データベース・パーティションのシャットダウン、およびページング・スペースを解放するためのトランザクションのアボートなどのユーザー定義イベントを説明する例が、samples/hacmp/es サブディレクトリーにあります。

/user/sbin/cluster/events/rules.hacmprd 規則ファイルには HACMP イベントが含まれています。このファイルの各イベント記述には、以下の 9 つの構成要素があります。

- イベント名。この名前はユニークでなければなりません。
- 状態。これは、イベントの修飾子です。イベント名および状態は、規則トリガーです。 HACMP ES クラスター・マネージャーは、規則トリガーがイベント名および状態に対応する規則を検出した場合にのみ、リカバリー処理を開始します。
- リソース・プログラム・パス。リカバリー・プログラムが入った *xxx*.rp ファイルを指定する絶対パスです。
- リカバリー・タイプ。これは、将来の利用のために予約されています。
- リカバリー・レベル。これは、将来の利用のために予約されています。
- リソース変数名。イベント・マネージャーのイベントで使用します。
- インスタンス・ベクトル。イベント・マネージャーのイベントで使用します。これは、形式 ″name=value″ のエレメント・セットです。各値は、システム内のリソースのコピーと、拡張子を使用してリソース変数のコピーを一意的に識別します。
- 述部。イベント・マネージャーのイベントで使用します。これは、リソース変数と他のエレメントとの関係式です。この式が真の場合、イベント管理サブシステムはイベントを生成して、クラスター管理や該当するアプリケーションに通知します。
- リアーム述部。イベント・マネージャーのイベントで使用します。この述部を使用して、基本述部の状況を変更するイベントを生成します。この述部は一般に、基本述部の反対です。また、問題となる条件の上限および下限の境界を設定するためのイベント述部でも使用できます。

たとえ行を使用しない場合でも、イベント定義では各オブジェクトに 1 行が必要となります。これらの行が削除されると、 HACMP ES クラスター・マネージャーはイベント定義を適切に解析できなくなり、これが原因でシステムがハングする可能性があります。「#」で始まる行は注釈行として扱われます。

注: 規則ファイルは、注釈行を数えないで、イベント定義ごとに 9 行だけを必要とします。規則ファイルの最後にユーザー定義のイベントを追加する場合、ファイルの末尾の不要な空白行を削除することが重要です。削除しなければ、ノードがハングします。

HACMP ES は PSSP イベント検出を使用して、ユーザー定義イベントを取り扱います。 PSSP イベント管理サブシステムは、さまざまなハードウェアおよびソフトウェア・リソースをモニターして、包括的なイベント検出を提供します。

次のようにプロセスを要約することができます。

1. (事前定義イベントの) グループ・サービス/ES または (ユーザー定義イベントの) イベント管理のいずれかが、イベントについて HACMP ES クラスター・マネージャーに通知します。
2. クラスター・マネージャーは rules.hacmprd ファイルを読み取り、イベントにマップされたリカバリー・プログラムを判別します。
3. クラスター・マネージャーは、一連のリカバリー・コマンドから成るリカバリー・プログラムを実行します。

4. リカバリー・プログラムは、シェル・スクリプトまたは バイナリー・コマンドであるリカバリー・コマンドを実行します。 (HACMP for AIX では、リカバリー・コマンドは、HACMP イベント・スクリプトと同じです。)
5. クラスター・マネージャーは、リカバリー・コマンドから戻り状況を受け取ります。予期しない状況により、 (`smit cm_rec_aids` または `/usr/sbin/cluster/utilities/clruncmd` コマンドによって) 手作業の介入が実行されるまで、クラスターは「停止」されます。

AIX における高度に使用可能な IBM® DB2 Universal Database™ 環境のインプリメンテーションと設計の詳細情報については、「DB2 UDB and DB2 Connect™ Support」Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できる以下の白書を参照してください。

- IBM DB2 Universal Database Enterprise Edition for AIX and HACMP/ES
- IBM DB2 Universal Database Enterprise - Extended Edition for AIX and HACMP/ES

関連資料:

- 「コマンド・リファレンス」の『db2start - DB2 の開始コマンド』

第 9 章 Windows オペレーティング・システムでのクラスター・サポート

Microsoft Cluster Server のサポート

概要

Microsoft Cluster Server (MSCS) は、Windows® NT サーバー、Windows 2000 サーバー、および Windows サーバー 2003 オペレーティング・システムの機能です。それは、高可用性およびデータとアプリケーションのより簡単な管理のために、クラスターへの 2 台のサーバー (最大 DataCenter サーバー内の 4 台のサーバー) の接続をサポートするソフトウェアです。 MSCS はさらに、自動的にサーバーまたはアプリケーションの障害を検出し、リカバリーすることもできます。それを使用して、サーバーのワークロードを移動して、マシン使用率の平衡を取り、ダウン時間を取らずに計画保守を行うことが可能になります。

以下の DB2® 製品は、MSCS をサポートします。

- DB2 Universal Database™ Workgroup Server Edition
- DB2 Universal Database Enterprise Server Edition (DB2 ESE)
- DB2 Universal Database Connect Enterprise Edition (DB2 CEE)

DB2 MSCS コンポーネント

クラスターとは、複数のノードの構成であり、各ノードは独立したコンピューター・システムです。クラスターは、ネットワーク・クライアントには単一サーバーのように見えます。

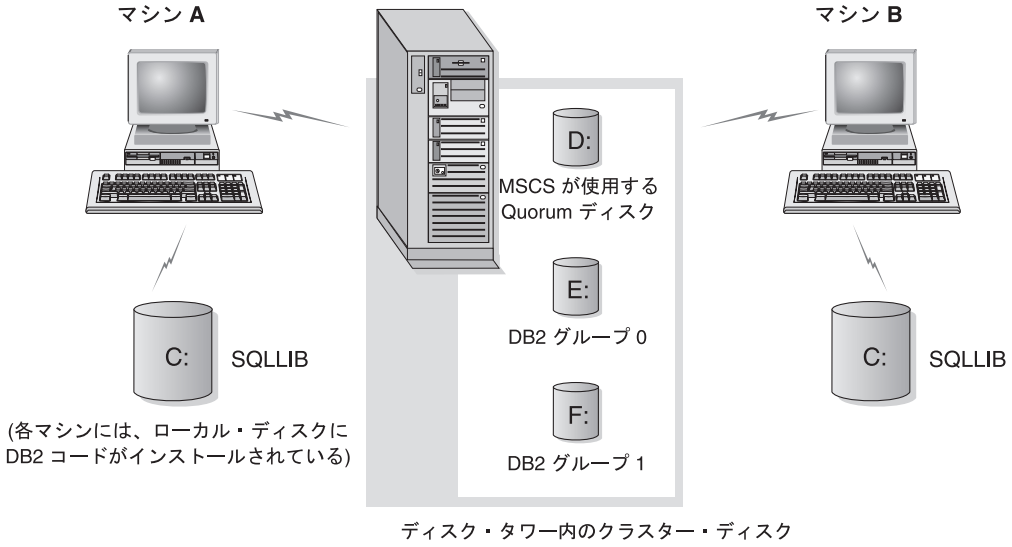


図 17. MSCS 構成の例

MSCS クラスター内のノードは、 1 つ以上の共有ストレージ・バスおよび 1 つ以上の物理的に独立したネットワークを使用して接続されています。サーバーだけを接続していてクライアントをクラスターに接続していないネットワークのことを、プライベート・ネットワークといいます。クライアント接続をサポートするネットワークのことを、公衆 ネットワークといいます。各ノードには、1 つ以上のローカル・ディスクがあります。各共有ストレージ・バスは、1 つ以上のディスクにアタッチします。共有バスの各ディスクは、1 度にクラスターの 1 つのノードだけが所有します。 DB2 ソフトウェアは、ローカル・ディスクに常駐します。 DB2 データベース・ファイル (表、索引、ログ・ファイルなど) は、共有ディスクに常駐します。 MSCS はクラスター内のロー・パーティションの使用をサポートしないため、DB2 を構成して MSCS 環境でロー・デバイスを使用することは不可能です。

DB2 リソース

MSCS 環境では、リソースはクラスタリング・ソフトウェアにより管理されるエンティティーです。たとえば、ディスク、IP アドレス、または汎用サービスは、リソースとして管理することができます。 DB2 は、DB2 と呼ばれるそれ自体のリソース・タイプを作成することにより MSCS と統合します。それぞれの DB2 リソースは、DB2 インスタンスを管理し、パーティション・データベース環境で実行される時、それぞれの DB2 リソースはデータベース・パーティションを管理します。その DB2 リソースの名前は、そのインスタンス名であり、パーティション・データベース環境ではあっても、その DB2 リソースの名前は、インスタンス名およびパーティション (またはノード) 番号の両方によって構成されます。

オンライン接続前およびオンライン接続後のスクリプト

スクリプトは、DB2 リソースがオンラインになった前でも、後でも実行できます。それらのスクリプトはそれぞれ、オンライン接続前およびオンライン接続後のスクリプトと呼ばれます。オンライン接続前およびオンライン接続後のスクリプトは、DB2 およびシステム・コマンドを実行できる .BAT ファイルです。

DB2 の複数インスタンスを同一のマシン上で実行している可能性のある状況では、オンライン接続前およびオンライン接続後のスクリプトを使用して、両方のインスタンスが正常に開始されるように構成を調整することができます。フェイルオーバーが発生した場合には、オンライン接続後のスクリプトを使用して、手動でデータベース・リカバリーを実行できます。さらに、オンライン接続後のスクリプトは、任意のアプリケーションまたは DB2 に依存するサービスを開始するためにも使用されます。

DB2 グループ

関連する、または従属するリソースは、リソース・グループに編成されます。 1 つのグループのすべてのリソースは、クラスター・ノード間を 1 つの単位として移動します。たとえば、典型的な DB2 単一パーティション・クラスター環境において、以下のリソースを含む DB2 グループがあるでしょう。

1. DB2 リソース。 DB2 リソースは DB2 インスタンス (またはノード) を管理します。
2. IP アドレス・リソース。 IP アドレス・リソースを使用すると、クライアント・アプリケーションが DB2 サーバーに接続することが可能になります。

3. Network Name リソース。 Network Name リソースを使用すると、IP アドレスを使用してではなく名前を使用して、クライアント・アプリケーションが DB2 サーバーに接続することが可能になります。 Network Name リソースは、IP アドレス・リソースと従属関係にあります。 Network Name リソースは、オプションです。 (Network Name リソースを構成すると、フェイルオーバー・パフォーマンスに影響するかもしれません。)
4. 1 つ以上の Physical Disk リソース。それぞれの Physical Disk リソースは、クラスター内の共有ディスクを管理します。

注: 同じグループの他のすべてのリソースがオンラインになってからのみ DB2 サーバーが開始されるように、 DB2 リソースは他のすべてのリソースと従属関係になるように構成されます。

フェイルオーバーの構成

2 つのタイプの構成が使用可能です。

- ホット・スタンバイ
- 相互テークオーバー

パーティション・データベース環境では、必ずしもすべてのクラスターが同タイプの構成を持つ必要はありません。ホット・スタンバイを使用するためにセットアップされるクラスターをいくつか、相互テークオーバーのためにセットアップされる他のクラスターをいくつか持つことができます。たとえば、DB2 インスタンスが 5 つのワークステーションで構成される場合、そのうち 2 つを相互テークオーバーの構成を使用するため、 2 つをホット・スタンバイを使用するため、 1 つのマシンをフェイルオーバー・サポート用に構成しないでおくことができます。

ホット・スタンバイ構成

ホット・スタンバイ構成では、 MSCS クラスターの 1 つのマシンが専用のフェイルオーバー・サポートを提供し、他のマシンがデータベース・システムに参加します。データベース・システムに参加しているマシンに障害が起こると、そのマシン上のデータベース・サーバーはフェイルオーバー・マシンで開始します。パーティション・データベース・システムで、あるマシン上で複数の論理ノードを実行していて、そのマシンに障害が起こる場合、論理ノードはフェイルオーバー・マシンで開始します。 252 ページの図 18 にホット・スタンバイ構成の例を示します。

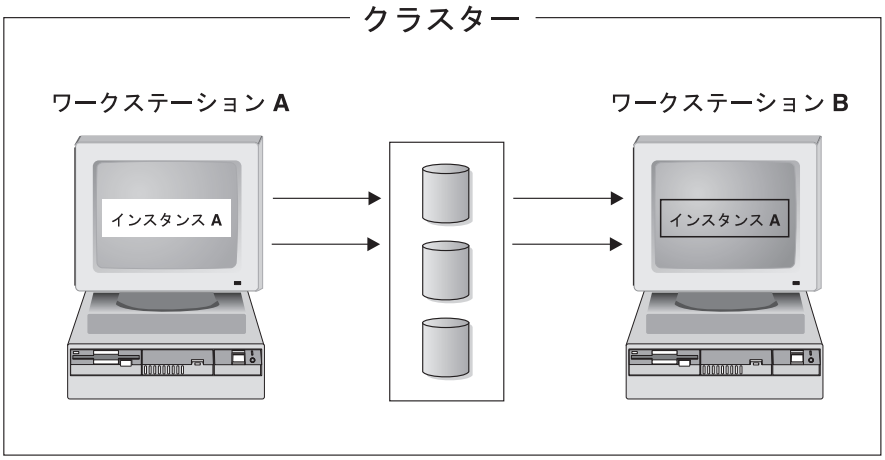


図 18. ホット・スタンバイ構成

相互テークオーバー構成

相互テークオーバー構成では、両方のワークステーションがデータベース・システムに参加します (つまり、各マシンに実行しているデータベース・サーバーが最低 1 つある)。 MSCS クラスターのワークステーションのいずれかに障害が起こると、障害が起きたマシン上のデータベース・サーバーは他のマシンで実行を開始します。相互テークオーバー構成では、あるマシン上のデータベース・サーバーは、別のマシン上のデータベース・サーバーとは別個に障害を起こす場合があります。指定されたどの時点であっても、すべてのデータベース・サーバーはどのマシン上でもアクティブであることができます。 図 19 に相互テークオーバー構成の例を示します。

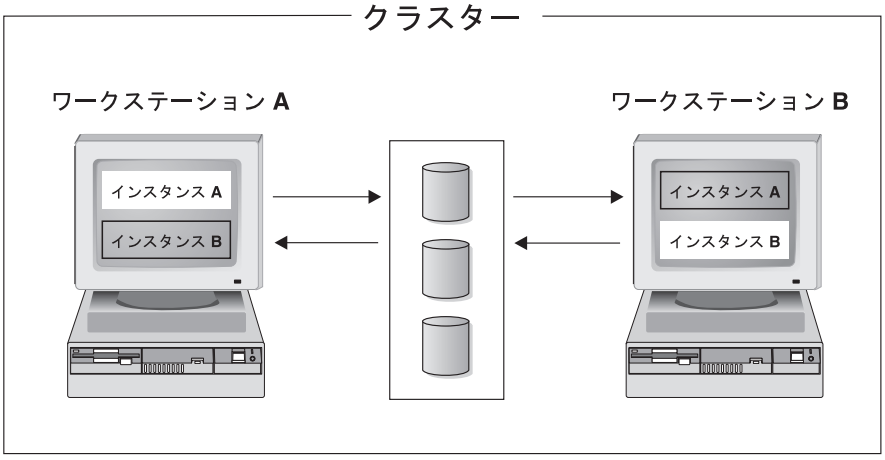


図 19. 相互テークオーバー構成

Windows オペレーティング・システムにおける高度に使用可能な IBM® DB2 Universal Database 環境のインプリメンテーションと設計の詳細情報については、「DB2 UDB and DB2 Connect™ Support」Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できる以下の白書を参照してください。

- 「Implementing IBM DB2 Universal Database Enterprise - Extended Edition with Microsoft® Cluster Server」
- 「Implementing IBM DB2 Universal Database Enterprise Edition with Microsoft Cluster Server」
- 「DB2 Universal Database for Windows: High Availability Support Using Microsoft Cluster Server - Overview」

第 10 章 Solaris オペレーティング環境でのクラスター・サポート

Solaris オペレーティング環境でのクラスター・サポート

Solaris™ オペレーティング環境での高可用性は、 DB2® が Sun™ Cluster または Veritas Cluster Server (VCS) と協働することによって達成されます。 Sun Cluster についての情報は、「DB2 Universal Database and High Availability on Sun Cluster 3.X」というタイトルの白書を参照してください。これは「DB2 UDB and DB2 Connect Online Support」web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できます。 VERITAS Cluster Server についての詳細は、「DB2 and High Availability on VERITAS Cluster Server」というタイトルの白書を参照してください。これは、「IBM® Support and downloads」Web サイト (http://www.ibm.com/support/docview.wss?uid=swg21045033) から入手できます。

注: Sun Cluster 3.0 または Veritas Cluster Server を使用している時は、 **db2iauto** ユーティリティーを以下のように使用して、 DB2 インスタンスがブート時に開始されないようにしてください。

```
db2iauto -off InstName
```

ここで

`InstName` は、インスタンスのログイン名。

高可用性

データ・サービスをホストするコンピューター・システムには、数多くの異なるコンポーネントがあり、各コンポーネントにはそれに関連した「平均故障間隔」(MTBF) があります。 MTBF は、コンポーネントが使用可能な状態にある平均時間です。質の高いハード・ディスクの MTBF は、100 万時間のオーダー (約 114 年) です。これは長期間に思えますが、200 個のディスクのうち 1 つは、 6 か月以内に故障する可能性があります。

データ・サービスの可用性を上げるための方法は多数ありますが、最も一般的なものは HA クラスターです。クラスターは、高可用性で使用される場合、複数のマシン、私設ネットワーク・インターフェースのセット、 1 つまたは複数の公衆ネットワーク・インターフェース、およびいくつかの共用ディスクから成ります。この特別な構成により、データ・サービスを 1 つのマシンから別のマシンに移動させることが可能になります。データ・サービスをクラスター内の別のマシンに移動することによって、引き続きそのデータにアクセスすることができます。データ・サービスを 1 つのマシンから別のマシンに移動することを*フェイルオーバー* といいます。これは、 256 ページの図 20 で図示されています。

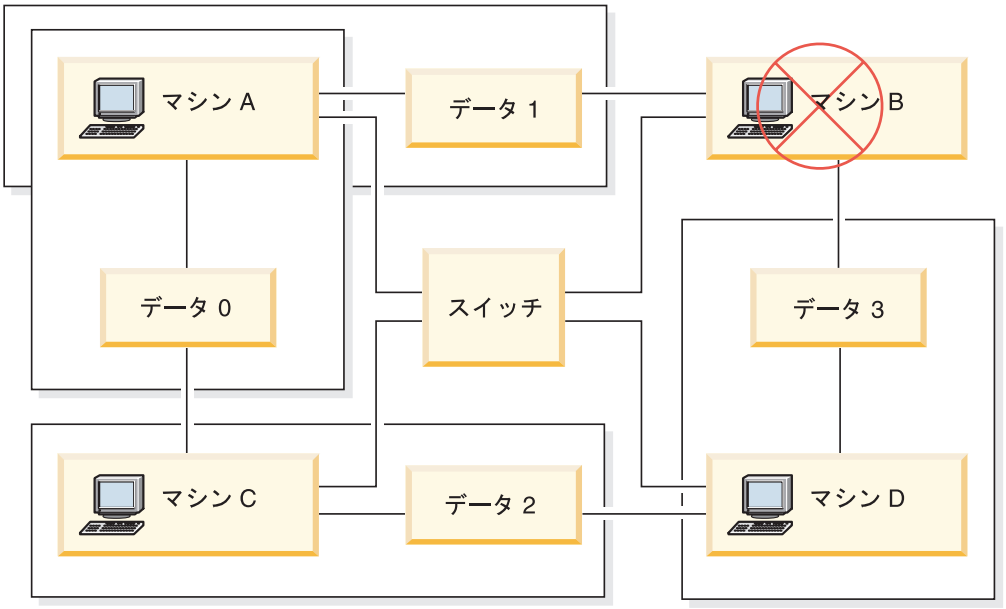


図 20. フェイルオーバー： マシン B のデータに障害が起こると、サービスはそのクラスターの別のマシンに移るため、データは以後もアクセス可能です。

私設ネットワーク・インターフェースは、クラスター内のマシン間でハートビート・メッセージおよび制御メッセージを送信するために使用します。公衆ネットワーク・インターフェースは、 HA クラスターのクライアントと直接通信するために使用します。 HA クラスター内のディスクはクラスター内の 2 つ以上のマシンと接続されており、片方のマシンで障害が起きた場合に別のマシンがそのディスクにアクセスできるようになっています。

HA クラスター上で実行されているデータ・サービスには、 1 つ以上の論理公衆ネットワーク・インターフェースと一連のディスクが関連付けられています。 HA データ・サービスのクライアントは、 TCP/IP 経由でデータ・サービスの論理ネットワーク・インターフェースにのみ接続します。フェイルオーバーが起きると、データ・サービスは、論理ネットワーク・インターフェースおよびディスクのセットとともに、別のマシンに移動します。

HA クラスターの利点の 1 つは、サポート・スタッフの援助なしでデータ・サービスをリカバリーできることと、いつでもそれを行えることです。もう 1 つの利点は、冗長度です。マシン自体を含め、クラスター内のすべての部分に冗長度があります。クラスターは、どんな単一の障害が起きても、存続することができます。

高可用性データ・サービスはそれぞれ性質がかなり異なる場合がありますが、いくつかの共通要件があります。高可用性データ・サービスのクライアントは、データ・サービスがどのマシン上にあっても、データ・サービスのネットワーク・アドレスおよびホスト名が同じであり、同じ方法で要求を出すことができると予期します。

可用性の高い Web サーバーにアクセスしている Web ブラウザーのことを考えてみてください。要求は URL とともに発行されます。 URL には、ホスト名と、web サーバー上のファイルへのパスの両方が含まれます。ブラウザーは、 web サーバーのフェイルオーバー後もホスト名とパスが同じであると予期します。ブラウザーが

web サーバーからファイルをダウンロードしているときに、サーバーがフェイルオーバーされると、ブラウザーは要求を再発行する必要があります。

データ・サービスの可用性は、データ・サービスがユーザーから使用可能である時間の合計により測定されます。可用性の最も一般的な測定単位は、「アップ時間」のパーセンテージです。これはしばしば、複数の「9」によって示されます。

```
99.99% => サービスは、1 年で (最大) 52.6 分ダウンする。
99.999% => サービスは、1 年で (最大) 5.26 分ダウンする。
99.9999% => サービスは、1 年で (最大) 31.5 秒ダウンする。
```

HA クラスターを設計し、テストするときは、

1. クラスターの管理者がシステムに精通しており、フェイルオーバーの発生時に起きることを知っていることを確かめます。
2. クラスターの各部分に正しく冗長度があり、障害が起きたときに素早く置き換えられることを確かめます。
3. 制御環境でテスト・システムに障害を起こし、システムが毎回正確にフェイルオーバーされることを確かめます。
4. それぞれのフェイルオーバーの理由を記録します。これはそれほど頻繁に起きることではありませんが、クラスターを不安定にする問題に対処するために重要です。たとえば、クラスターの一部が 1 か月に 5 回フェイルオーバーを起こした場合、その理由を調べてそれを修正してください。
5. フェイルオーバーが起きたときに、そのことがクラスターのサポート・スタッフに知らされることを確かめます。
6. クラスターが過負荷にならないようにします。フェイルオーバーの後で、残りのシステムが引き続き許容レベルでワークロードを処理できることを確かめてください。
7. よく障害の起きるコンポーネント (ディスクなど) をチェックし、問題が起きる前に置き換えます。

フォールト・トレランス

データ・サービスの可用性を上げる別の方法は、フォールト・トレランスです。フォールト・トレラント・マシンには、すべての冗長度が組み込まれており、 CPU やメモリーを含む任意の部分で起きる単一の障害に対処することができます。フォールト・トレラント・マシンは、すき間産業で最も良く使用され、通常、このマシンをインプリメントするには費用がかかります。地理的に別の場所にあるマシンを含む HA クラスターには、これらの場所のサブセットだけに影響を与える災害から、リカバリーすることができるという利点があります。

HA クラスターは可用性を上げる最も一般的な方法です。なぜなら、拡張が容易で、使いやすく、そして比較的インプリメントするために費用がかからないからです。

関連概念:

- 258 ページの『Sun Cluster 3.0 のサポート』
- 261 ページの『VERITAS Cluster Server のサポート』

Sun Cluster 3.0 のサポート

ここでは、 DB2® が Sun™ Cluster 3.0 と協働して高可用性を実現する方法についての概要を説明しています。また、それら 2 つのソフトウェア製品間で仲介者として機能する、高可用性エージェントについても説明しています (図 21 を参照してください)。

図 21. DB2、Sun Cluster 3.0、および高可用性： DB2、Sun Cluster 3.0 と高可用性エージェントとの間の関係

フェイルオーバー

Sun Cluster 3.0 は、アプリケーション・フェイルオーバーを使用可能にすることにより高可用性を提供します。各ノードは周期的にモニターされ、クラスター・ソフトウェアは、障害が起きた 1 次ノードから指定された 2 次ノードへ自動的にクラスターに依存するアプリケーションを再配置します。フェイルオーバーが発生すると、クライアントにはサービスに短い中断があるかもしれず、サーバーに再接続する必要があるかもしれません。とはいえ、クライアントは、アプリケーションやデータにアクセスするために使用している物理サーバーには気付かないでしょう。クラスター内の別のノードが 1 次ノードの障害時に自動的にワークロードを処理することを許可することにより、 Sun Cluster 3.0 では、大幅にダウン時間を削減し、生産性を上げることができます。

マルチホスト・ディスク

Sun Cluster 3.0 には、マルチホスト・ディスク装置が必須です。これは、ディスクが複数のノードに 1 度に接続されることがあるということです。 Sun Cluster 3.0 環境では、マルチホスト記憶装置により、ディスク装置が高度に使用可能になります。マルチホスト記憶装置にあるディスク装置は、代替サーバー・ノードを経由するデータへの物理パスがあるため、単一ノードの故障を容認できます。マルチホスト・ディスクは、1 次ノードを経由してグローバルにアクセス可能です。クライアント要求があるノードを経由してデータにアクセスしており、そのノードに障害が起こった場合、その要求は、同じディスクへの直接接続を持つ別のノードへ切り替わります。ボリューム・マネージャーは、マルチホスト・ディスクのデータ冗長度のミラーリングされた構成または RAID 5 構成を提供します。現行では、Sun Cluster 3.0 はボリューム・マネージャーとして、 Solstice DiskSuite および VERITAS Volume Manager をサポートします。マルチホスト・ディスクをディスク・ミラーおよびストライピングと結合させるなら、ノード障害および個々のディスク障害の両方を保護します。

グローバル・デバイス

グローバル・デバイスは、装置の物理的な位置にかかわらず、任意のノードからクラスター内の任意の装置への高度に使用可能なアクセスを、クラスター全体にわたり提供するために使用されます。すべてのディスクは、グローバル・ネームスペース内に、割り当てられたデバイス ID (DID) を付けて組み込まれ、グローバル・デバイスとして構成されます。そのため、ディスク自体はすべてのクラスター・ノードから可視状態になります。

ファイル・システム/グローバル・ファイル・システム

クラスター・ファイル・システムまたはグローバル・ファイル・システムは、(1 つのノード上の) カーネルと、 (1 つ以上のディスクへの物理接続を持つノード上の) 元となるファイル・システム・ボリューム・マネージャーとの間のプロキシーです。クラスター・ファイル・システムは、1 つ以上のノードへの物理接続を持つグローバル・デバイスに依存しています。クラスター・ファイル・システムは、元となるファイル・システムおよびボリューム・マネージャーから独立しています。現行では、クラスター・ファイル・システムは、 Solstice DiskSuite または VERITAS Volume Manager のいずれかを使用して UFS 上に作成できます。ディスク上のファイル・システムがクラスター・ファイル・システムとしてグローバルにマウントされた場合にのみ、データは、すべてのノードに対して使用可能になります。

デバイス・グループ

すべてのマルチホスト・ディスクは、Sun Cluster のフレームワークで制御しなければなりません。 Solstice DiskSuite または VERITAS Volume Manager で管理されるディスク・グループは、まずマルチホスト・ディスク上に作成されます。その後、それらは Sun Cluster ディスク・デバイス・グループに登録されます。ディスク・デバイス・グループは、一種のグローバル・デバイスです。マルチホスト・デバイス・グループは、高度に使用可能です。現行でデバイス・グループを管理しているノードに障害が起こった場合には、ディスクは、代替パスを経由してアクセス可能になります。デバイス・グループを管理しているノードの障害により、リカバリーおよび整合性チェックを行うために必要な時間を除いては、デバイス・グループへのアクセスに影響しません。その時間中は、すべての要求は、システムがデバイス・グループを使用可能にするまでブロックされます (このことはアプリケーションには知らされません)。

リソース・グループ・マネージャー (RGM)

RGM は、高可用性のための機構を備え、それぞれのクラスター・ノード上でデーモンとして実行されます。それは、すでに構成されているポリシーに基づいて、選択したノード上のリソースを自動的に開始および停止します。 RGM の使用により、ノード障害の時にリソースを高度に使用可能にしたり、影響を受けたノード上のリソースを停止することによりリブートし、別のノード上で開始したりすることができます。さらに RGM は、リソース障害を検出したり障害が起きたリソースを別のノード上に再配置することができる特定のリソースのモニターを、自動的に開始および停止できます。

データ・サービス

データ・サービスという用語は、単一サーバー上ではなく、クラスター上で実行するように構成されているサード・パーティー・アプリケーションを説明するために用いられます。データ・サービスには、アプリケーションを開始、停止、およびモニターするアプリケーション・ソフトウェアおよび Sun Cluster 3.0 ソフトウェアが含まれます。 Sun Cluster 3.0 は、クラスター内でアプリケーションを制御またはモニターするために使用されるデータ・サービス・メソッドを提供します。これらのメソッドは、リソース・グループ・マネージャー (RGM) の制御下で実行されます。 RGM は、メソッドを使用してクラスター・ノード上のアプリケーションを開始、停止、およびモニターします。それらのメソッドは、クラスター・フレームワーク・ソフトウェアおよびマルチホスト・ディスクとともに、アプリケーションが高度に使用可能なデータ・サービスになるようにします。高度に使用可能なデータ・サービスとして、それらのメソッドは、単一の障害の後にそのクラスター内で発生するかなりの数のアプリケーション中断を防ぎます。それは、障害がノード上、インターフェース・コンポーネント上、またはアプリケーション自体のいずれで起きたかにかかわらず行われます。さらに RGM は、ネットワーク・リソース (論理ホスト名および共有アドレス) およびアプリケーション・インスタンスを含む、クラスター内のリソースを管理します。

リソース・タイプ、リソースおよびリソース・グループ

リソース・タイプは、以下のもので構成されます。

1. クラスター上で実行されるソフトウェア・アプリケーション。
2. アプリケーションをクラスター・リソースとして管理するために RGM によりコールバック・メソッドとして使用されるコントロール・プログラム。
3. クラスターの静的構成の一部を形式するプロパティーのセット。

RGM は、リソース・タイプ・プロパティーを使用して、特定のタイプのリソースを管理します。

リソースは、プロパティーおよびそのリソース・タイプの値を継承します。それは、クラスター上で実行中の元となるアプリケーションのインスタンスです。各インスタンスには、クラスター内でのユニーク名が必要です。各リソースは、リソース・グループ内に構成される必要があります。 RGM は、すべてのリソースを、オンラインおよびオフラインで同じノードの 1 つのグループにまとめます。 RGM がリソース・グループをオンラインまたはオフラインにする時には、 RGM はそのグループの個々のリソース上でコールバック・メソッドを呼び出します。

リソース・グループが現行でオンラインになっているノードのことを、そのリソース・グループの 1 次ノードまたはその 1 次と呼びます。リソース・グループは、それぞれの 1 次により管理されます。各リソース・グループは、関連した Nodelist プロパティーがあり、それは、クラスター管理により設定され、すべてのリソース・グループの 1 次またはマスターである可能性があるものを識別します。

Sun Cluster 3.0 における高度に使用可能な IBM® DB2 Universal Database 環境のインプリメンテーションと設計の詳細情報については、「DB2 UDB and DB2 Connect™ Support」Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できる、「DB2 and High Availability on Sun Cluster 3.0」というタイトルの白書を参照してください。

関連概念:

- 255 ページの『Solaris オペレーティング環境でのクラスター・サポート』
- 261 ページの『VERITAS Cluster Server のサポート』

VERITAS Cluster Server のサポート

VERITAS Cluster Server は、計画された稼働休止時間および臨時の稼働休止時間の両方を削減するために使用されます。それにより、異機種混合環境におけるサーバー統合および広範囲にわたるアプリケーションの効果的な管理を容易にできます。VERITAS Cluster Server は、ストレージ・エリア・ネットワーク (SAN) および従来型のクライアント/サーバー環境の両方において、最大 32 のノード・クラスターをサポートします。VERITAS Cluster Server はネットワーク・ストレージ環境において、単一の重要なデータベース・インスタンスから非常に大規模なマルチ・アプリケーション・クラスターまで、すべてを保護できます。このセクションでは、VERITAS Cluster Server の機能の簡単なサマリーをしています。

ハードウェア要件

以下のものは、現行で VERITAS Cluster Server がサポートするハードウェアのリストです。

- サーバー・ノード:
  - Solaris™ 2.6 以降を実行する Sun™ Microsystems の SPARC/Solaris サーバー (最低 128MB の RAM)。
- ディスク装置:
  - EMC Symmetrix、IBM® Enterprise Storage Server®、HDS 7700 および 9xxx、Sun T3、 Sun A5000、Sun A1000、Sun D1000 および VCS 2.0 以降がサポートするその他の任意のディスク装置。 VERITAS 担当者はどのディスク・サブシステムがサポートされているかを確認できますし、VCS 資料を参照することもできます。
  - 典型的な環境では、 DB2® UDB バイナリー用にミラーリングされた専用ディスクが (各クラスター・ノードに) 必要であり、 DB2 UDB データ用にノード間の共用ディスクが必要です。
- ネットワーク相互接続:
  - 公衆ネットワーク接続用に、IP ベースのアドレッシングをサポートする任意のネットワーク接続が必要です。
  - ハートビート接続 (クラスターへの内部接続) 用に、重複ハートビート接続が必要です。この要件は、サーバーごとに 2 つのイーサネット・コントローラーを追加して使用するか、サーバーごとに 1 つのイーサネット・コントローラーを追加し、クラスターごとに 1 つの共用 GABdisk を使用することにより満たすことができます。

ソフトウェア要件

以下の VERITAS ソフトウェア・コンポーネントは、条件付きの構成です。

- VERITAS Volume Manager 3.2 以降、VERITAS File System 3.4 以降、VERITAS Cluster Server 2.0 以降。
- DB Edition for DB2 for Solaris 1.0 以降。

VERITAS Cluster Server にはボリューム・マネージャーは必要ないため、容易にインストール、構成、および管理をするために VERITAS Volume Manager を使用することを強くお勧めします。

フェイルオーバー

VERITAS Cluster Server は、アプリケーション・フェイルオーバーを使用可能にすることによりアプリケーション・サービス (DB2 UDB など) の可用性を管理する、可用性クラスタリング・ソリューションです。それぞれの個々のクラスター・ノードおよびそれに関連したソフトウェア・サービスの状態は、定期的にモニターされます。アプリケーション・サービス (この場合では、DB2 UDB サービス) を中断してしまうような障害が発生した時、 VERITAS Cluster Server および/または VCS HA-DB2 Agent は障害を検出し、自動的にサービスをリストアするためのステップに入ります。このことには、同じノード上で DB2 UDB を再始動したり、そのクラスターの別のノードに DB2 UDB を移動し、そのノードで再始動することが含まれるかもしれません。アプリケーションを新規のノードに移行する必要がある場合、VERITAS Cluster Server は、アプリケーションに関連したすべてのもの (すなわち、ネットワーク IP アドレス、元となるストレージの所有権) を新規ノードに移動しますので、サービスが実際には別のノードで実行されていることをユーザーは気付かないでしょう。ユーザーは、以後も同一の IP アドレスを使用してサービスにアクセスできますが、それらのアドレスは、異なるクラスター・ノードを指すようになります。

VERITAS Cluster Server でフェイルオーバーが発生すると、ユーザーはサービスの中断に気付くかもしれませんし、気付かないかもしれません。これは、そのアプリケーション・サービスにおいてクライアントが使用している接続のタイプ (ステートフルまたはステートレス) によります。 (DB2 UDB のような) ステートフル接続でのアプリケーション環境では、ユーザーはサービスの短い中断に気付き、フェイルオーバーが完了した後に再接続する必要があるかもしれません。 (NFS のような) ステートレス接続でのアプリケーション環境では、ユーザーはサービスの短い遅延に気付くかもしれませんが、一般的には中断には気付かず、またログオンする必要もないでしょう。

クラスター・ノード間で自動的に移行できるサービスとしてアプリケーションをサポートすることにより、 VERITAS Cluster Server は、稼働休止時間を短縮できるだけでなく、計画された稼働休止時間 (すなわち、保守やアップグレードのための) に関連した停止の継続時間を短縮することもできます。フェイルオーバーは手動で開始することもできます。ハードウェアまたはオペレーティング・システムのアップグレードを特定のノードで実施する必要がある場合、 DB2 UDB をそのクラスターの別のノードに移行し、アップグレードを行い、その後 DB2 UDB を元のノードに移行できます。

このタイプのクラスタリング環境で使用するために推奨されているアプリケーションは、破損に耐えられるものでなければなりません。破損に耐えられるアプリケーションは、コミット済みデータの整合性を維持しながら予期しない破損からリカバリーすることができます。破損に耐えられるアプリケーションはしばしば、クラスター・フレンドリーな アプリケーションと言われます。 DB2 UDB は、破損に耐えられるアプリケーションです。

VERITAS CFS、CVM、および VCS を使用してフェイルオーバーを実行するときにかかる時間を減らす方法についての情報は、「DB2 UDB Version 8 and VERITAS Database Edition: Accelerating Failover Times in DB2 UDB Database Environments」というタイトルの白書を参照してください。これは「DB2 UDB and DB2 Connect™ Online Support」Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できます。

共用ストレージ

VCS HA-DB2 Agent とともに使用した場合、Veritas Cluster Server は、共用ストレージが必要です。共用ストレージとは、クラスター内の複数のノードへの物理接続を持つストレージです。共用ストレージに常駐のディスク装置は、障害があっても、ディスク装置への物理パスが 1 つ以上の代替クラスター・ノードを経由して存在するため、ノード障害に耐えることができます。

VERITAS Cluster Server の制御を通して、クラスター・ノードは、「ディスク・グループ」と呼ばれる論理構成を通して共用ストレージにアクセスできます。ディスク・グループは、論理的に定義されたストレージ・デバイス (その所有権はクラスターのノード間で自動的に移行される) の集合を表します。ディスク・グループは、任意の特定の時に、単一のノードにのみインポートすることができます。たとえば、Disk Group A が Node 1 にインポートされた後、Node 1 に障害が起きた場合、 Disk Group A を障害が起きたノードからエクスポートし、そのクラスターの新規ノードにインポートすることができます。 VERITAS Cluster Server は、単一クラスター内の複数のディスク・グループを同時に制御することができます。

ディスク・グループ定義を許可することに加え、ボリューム・マネージャーは、ミラーリングまたは RAID 5 を使用して、共用ストレージに冗長データ構成を提供することができます。 VERITAS Cluster Server は、 VERITAS Volume Manager および Solstice DiskSuite を論理ボリューム・マネージャーとしてサポートします。共用ストレージをディスク・ミラーおよびストライピングと結合させるなら、ノード障害および個々のディスクまたはコントローラー障害の両方を保護できます。

VERITAS Cluster Server Global Atomic Broadcast (GAB) および省待機時間トランスポート (LLT)

ノードがハードウェアおよびソフトウェア状況に関する情報を交換したり、クラスターのメンバーシップを追跡したり、この情報をすべてのクラスター・ノードにわたり同期できるようにするために、ノード間通信メカニズムが、クラスター構成内になければなりません。 Low Latency Transport (LLT) 全体にわたり実行される Global Atomic Broadcast (GAB) 機能は、 VERITAS Cluster Server がこのことを行うために用いられる高速で待ち時間が少ないメカニズムを備えています。 GAB は、各クラスター・ノード上にカーネル・モジュールとしてロードされ、確実にすべてのノードが同時に状況更新情報を入手するようにする Atomic Broadcast のメカニズムを提供します。

カーネル間通信能力の効力により、LLT はクラスター・ノード間で交換および同期する必要があるすべての情報に対して、高速で、待ち時間が少ないトランスポートを提供します。 GAB は、LLT の上で実行されます。 VERITAS Cluster Server は、IP をハートビート機構として使用しませんが、さらに他の 2 つの信頼性のあるオプションを備えています。 LLT とともに使用する GAB は、ハートビート機

構として作動するように構成できますし、 GABdisk はディスク・ベースのハートビートとして構成できます。ハートビートは重複接続を使用して実行する必要があります。それらの接続は、2 本のクラスター・ノード間のプライベート・イーサネット接続か、 1 本のプライベート・イーサネット接続と 1 本の GABdisk 接続のいずれかになります。 2 本の GABdisks はサポートされておりません。それはノード間のクラスター状況の交換には、プライベート・イーサネット接続が必要だからです。

GAB または LLT についての詳細、またはそれらを VERITAS Cluster Server 構成で構成する方法については、「VERITAS Cluster Server 2.0 User's Guide for Solaris」をご覧ください。

バンドルおよびエンタープライズ・エージェント

エージェントとは、特定のリソースまたはアプリケーションの可用性を管理するために設計されたプログラムです。エージェントが開始されると、それは VCS から必要な構成情報を取得し、その後周期的にリソースまたはアプリケーションをモニターし、VCS を状況とともに更新します。一般にはエージェントは、リソースをオンラインにしたり、リソースをオフラインにしたり、あるいはリソースをモニターし、4 種類のサービス (開始、停止、モニターおよびクリーン) を提供するために使用されます。開始および停止は、リソースをオンラインまたはオフラインにするために使用され、モニターは、特定のリソースまたはアプリケーションの状況を見るためにそれをテストするために使用され、クリーンはリカバリー処理で使用されます。

VERITAS Cluster Server の一部として様々なバンドル・エージェントが組み込まれており、 VERITAS Cluster Server のインストール時にインストールされます。バンドル・エージェントは、通常クラスター構成にある定義済みリソース・タイプ (すなわち、IP、マウント、プロセス、および共用) を管理する VCS プロセスで、それらはクラスターのインストールおよび構成をかなりの程度単純化する上で役立ちます。 VERITAS Cluster Server には、20 以上のバンドル・エージェントがあります。

エンタープライズ・エージェントは、どちらかというと DB2 UDB のような特定のアプリケーションに焦点が向けられています。 VCS HA-DB2 Agent は、エンタープライズ・エージェントと考えることができ、それは、VCS とのインターフェースを VCS Agent フレームワークを通して行います。

VCS リソース、リソース・タイプ、およびリソース・グループ

リソース・タイプとは、モニターされる VCS クラスター内部のリソースを定義するために使用されるオブジェクト定義のことをいいます。リソース・タイプには、リソース・タイプ名およびそのリソースに関連していて高可用性の観点からみて顕著なプロパティーのセットが含まれます。リソースは、プロパティーおよびそのリソース・タイプの値を継承し、リソース名は、クラスター全体にわたりユニークなものでなければなりません。

リソースには、2 つのタイプがあります。永続および標準 (非永続) です。永続リソースとは、モニターされてはいても VCS によりオンラインまたはオフラインにならない、ネットワーク・インターフェース・コントローラー (NIC) のようなリソー

スのことをいいます。標準リソースとは、オンラインおよびオフラインの状況が VCS により制御されるもののことをいいます。

モニターされる最低レベルのオブジェクトはリソースであり、様々なリソース・タイプがあります (すなわち、共用、マウント)。各リソースは、リソース・グループに構成される必要があり、 VCS は特定のリソース・グループのすべてのリソースをまとめてオンラインおよびオフラインにします。リソース・グループをオンラインまたはオフラインにするために、 VCS はそのグループの各リソースのために開始または停止メソッドを呼び出します。リソース・グループには、2 つのタイプがあります。フェイルオーバーおよび並列です。可用性の高い DB2 UDB 構成では、それがパーティション化されていてもいなくても、フェイルオーバー・リソース・グループを使用します。

「1 次」または「マスター」ノードとは、リソースをホストする可能性があるノードのことをいいます。 `systemlist` と呼ばれるリソース・グループ属性は、クラスター内のどのノードが特定のリソース・グループの 1 次になれるかを指定するために使用されます。 2 つのノードがあるクラスターでは、たいてい両方のノードは `systemlist` に含まれますが、いくつもの高度に使用可能なアプリケーションを管理するようなより大規模なマルチノード・クラスターでは、 (最低のレベルのリソースにより定義される) 特定のアプリケーション・サービスが特定のノードに決してフェイルオーバーされないようにすることを保証するための要件があるかもしれません。

リソース・グループ間に従属関係を定義することができ、VERITAS Cluster Server は、さまざまなリソース障害の影響を評価する上で、さらにリカバリーを管理する上でこのリソース・グループ従属関係階層に依存しています。たとえば、リソース・グループ DB2 がすでに正常に開始済みでないかぎりリソース・グループ ClientApp1 がオンラインにならない場合には、リソース・グループ ClientApp1 は、リソース・グループ DB2 に依存していると考えられます。

VERITAS Cluster Server による高度に使用可能な IBM DB2 Universal Database 環境のインプリメンテーションと設計についての詳細情報は、「DB2 UDB and High Availability with VERITAS Cluster Server」というタイトルのテクニカル・ノートを参照してください。それは、http://www.ibm.com/support でキーワード「1045033」を検索することにより参照できます。

関連概念:

- 255 ページの『Solaris オペレーティング環境でのクラスター・サポート』
- 258 ページの『Sun Cluster 3.0 のサポート』

第 3 部 付録

付録 A. 構文図の読み方

構文図では、オペレーティング・システムが入力内容を正しく判別できるようにコマンドを指定する方法を示します。

構文図は、水平線 (メインパス) に沿って左から右、上から下へ読みます。線が矢印で終わっている場合、コマンド構文は継続しており、次の線は矢印で始まります。垂直線は、コマンド構文の終わりを示します。

構文図からの情報を入力する時は、引用符や等号などの記号類を必ず含めてください。

パラメーターは、キーワードと変数に分類されます。

- キーワードは定数を表し、英大文字です。しかし、コマンド・プロンプトでは、大文字でも、小文字でも、大文字小文字の混合でも構いません。コマンド名はキーワードの一例です。
- 変数はユーザーが提供する名前や値を表し、英小文字で示されます。しかし、コマンド・プロンプトでは、文字の種類がはっきり指定されている場合以外は、大文字、小文字、大文字小文字の混合のどれで入力しても構いません。ファイル名は変数の一例です。

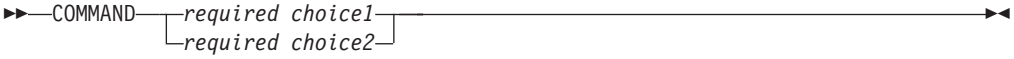
1 つのパラメーターがキーワードと変数の組み合わせである場合があります。

必要パラメーターは、メインパスと同じ高さに示されます。

オプション・パラメーターは、メインパスの下側に示されます。

パラメーターのデフォルト値は、線の上側に示されます。

パラメーターのスタックで、最初のパラメーターがメインパスと同じ高さに示されている場合は、パラメーターの 1 つを必ず選択しなければならないことを示します。



パラメーターのスタックで、最初のパラメーターがメインパスの下側に示されている場合は、パラメーターの 1 つを選択可能であることを示します。

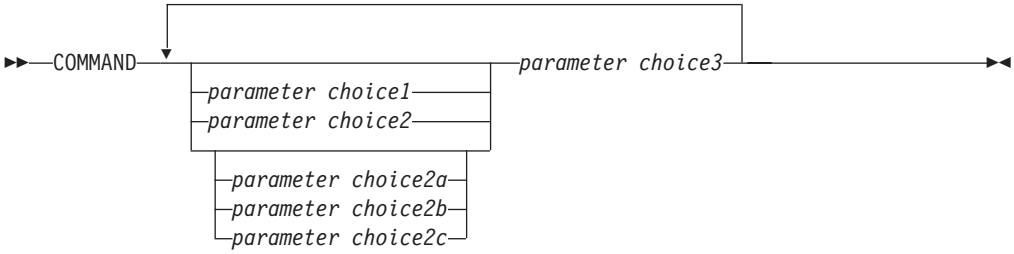
線の上側で左に戻る矢印がある場合は、項目が以下の規則に従って繰り返し可能であることを示します。

- 矢印が中断されていない場合、項目はブランク・スペースによって区切られたリストの形式で繰り返すことができます。

- 矢印にコンマが含まれている場合、項目はコンマによって区切られたリストの形式で繰り返すことができます。

パラメーター・スタックの項目は、前述の必須およびオプション・パラメーターのスタック規則に従って繰り返すことができます。

一部の構文図では、パラメーター・スタックが他のパラメーター・スタックの中に含まれていることがあります。スタックの項目を繰り返すには、前述の規則に従わなければなりません。つまり、内側のスタックの上に繰り返し矢印がなく、外側のスタックの上にはある場合には、内側のスタックから 1 つのパラメーターのみを選択し、外側のスタックからの任意のパラメーターと組み合わせて、その組み合わせを繰り返すことができます。たとえば、次の図では、パラメーター *choice2a* をパラメーター *choice2* と組み合わせて、その組み合わせ (*choice2* と *choice2a*) を繰り返すことができます。



一部のコマンドの前には、オプションのパス・パラメーターが付けられています。

このパラメーターを指定しない場合、システムはコマンドを現行ディレクトリーから検索します。コマンドが見つからない場合、システムは .profile にリストされているパスにあるすべてのディレクトリーでコマンドの検索を続行します。

一部のコマンドには、機能的に同等な構文上の変形があります。

付録 B. 警告、エラー、および完了メッセージ

SQL メッセージの中には、さまざまなユーティリティーによって生成されるものが含まれます。それらのメッセージは、警告またはエラー条件が検出された場合にデータベース・マネージャーによって生成されます。各メッセージには、接頭部 (SQL) と 4 桁または 5 桁のメッセージ番号から構成されるメッセージ ID があります。メッセージ・タイプには、通知、警告、および重大の 3 種類があります。 N で終わるメッセージ ID は、エラー・メッセージです。 W で終わるメッセージ ID は、警告または通知メッセージです。 C で終わるメッセージ ID は、重大なシステム・エラーです。

メッセージ番号は *SQLCODE* とも呼ばれます。 SQLCODE は、そのメッセージ・タイプ (N、W、または C) に従って、正数または負数としてアプリケーションに渡されます。 N および C の場合は負の値、W の場合は正の値になります。 DB2 は SQLCODE をアプリケーションに戻し、アプリケーションでは SQLCODE に関連するメッセージを入手することができます。さらに DB2 は、SQL ステートメントの結果として発生することのある条件を表す *SQLSTATE* 値を戻します。一部の SQLCODE 値には、それに関連する SQLSTATE 値があります。

この資料に記載されている情報を使用すると、エラーまたは問題を識別し、適切なリカバリー処置を使用することによって問題を解決することができます。また、この情報はメッセージが生成および記録された場所を理解するためにも使用できます。

SQL メッセージ、および SQLSTATE 値に関連するメッセージ・テキストについては、オペレーティング・システムのコマンド行で調べることもできます。これらのエラー・メッセージのヘルプを表示するには、オペレーティング・システムのコマンド・プロンプトで次のように入力します。

```
db2 ? SQLnnnnn
```

nnnnn はメッセージ番号です。 UNIX ベースのシステムでは、二重引用符の区切り文字を使用することが推奨されています。これにより、ディレクトリー内に単一文字のファイル名がある場合の問題を避けることができます。

```
db2 "? SQLnnnnn "
```

db2 コマンドのパラメーターとして受け入れられるメッセージ ID には大文字小文字の区別がなく、最後の文字は省略可能です。したがって、以下のコマンドはどれも同じ結果になります。

```
db2 ? SQL0000N
db2 ? sql0000
db2 ? SQL0000n
```

メッセージ・テキストが長すぎて画面に入らない場合は、次のコマンドを使用してください (UNIX ベースのオペレーティング・システム、および ″more″ パイプをサポートするその他のオペレーティング・システムの場合)。

```
db2 ? SQLnnnnn | more
```

出力をファイルにリダイレクトし、後でそれを見ることもできます。

ヘルプは、対話式入力モードから呼び出すこともできます。このモードにアクセスするには、オペレーティング・システムのコマンド・プロンプトで次のように入力します。

```
db2
```

このモードで DB2 メッセージ・ヘルプを表示するには、コマンド・プロンプト (db2 =>) で次のように入力します。

```
? SQLnnnnn
```

SQLSTATE に関連するメッセージ・テキストは、次のコマンドを実行することによって検索できます。

```
db2 ? nnnnn
 または
db2 ? nn
```

nnnnn は 5 文字の SQLSTATE 値 (英数字)、 *nn* は 2 桁の SQLSTATE クラス・コード (SQLSTATE 値の最初の 2 桁) です。

付録 C. 追加の DB2 コマンド

この付録では、このマニュアルでは詳しく説明されていない、リカバリー関連のシステム・コマンドおよび CLP コマンドを紹介します。

システム・コマンド

db2adutl - TSM 内の DB2 オブジェクトの管理

Tivoli Storage Manager を使用して保管した、バックアップ・イメージ、ログ、およびロード・コピー・イメージの、照会、抽出、検査、および削除をユーザーに許可します。また、ユーザーが TSM サーバー上のオブジェクトへのアクセスを付与したり取り消したりできるようにします。

UNIX ベースのオペレーティング・システムでは、このユーティリティーは sqllib/adsm ディレクトリーにあります。 Windows では、これは sqllib¥bin にあります。

権限:

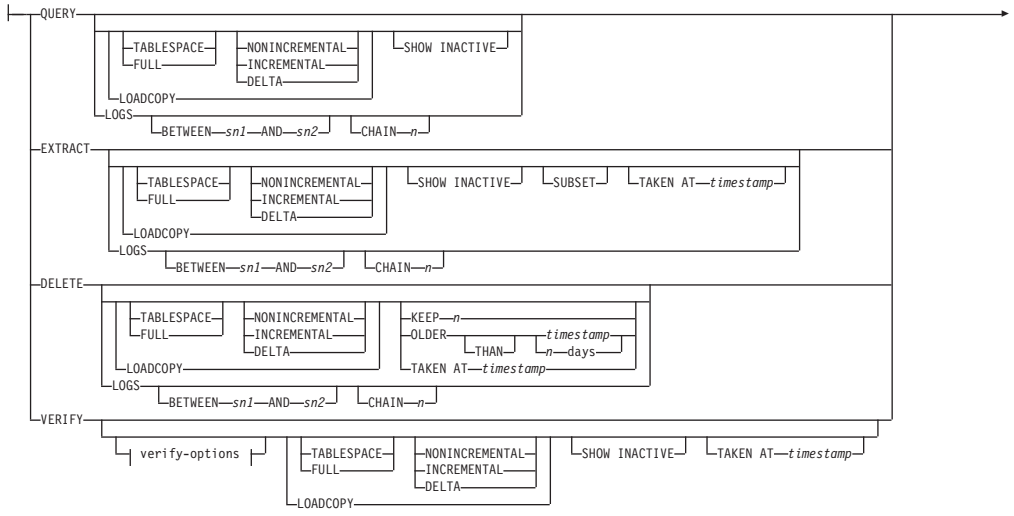
なし

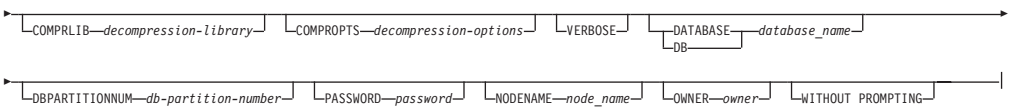
必要な接続:

なし

コマンド構文:

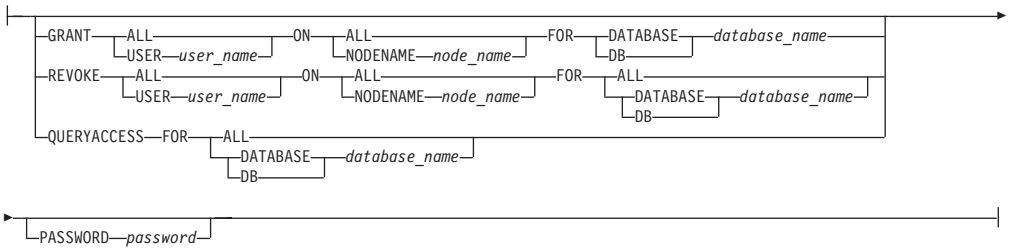
db2-object-options:





verify-options:

access-control-options:



コマンド・パラメーター:

QUERY
TSM サーバーで DB2 オブジェクトを照会します。

EXTRACT
TSM サーバーからの DB2 オブジェクトを、ローカル・マシンにある現行ディレクトリーにコピーします。

DELETE
バックアップ・オブジェクトを非活動化するか、または TSM サーバーにあるログ・アーカイブを削除します。

VERIFY
サーバー上のバックアップ・コピーに対して整合性検査を実行します。

注: このパラメーターを指定すると、バックアップ・イメージ全体がネットワークを介して転送されます。

ALL 使用可能なすべての情報を表示します。

CHECK
チェックビットおよびチェックサムの結果を表示します。

DMS DMS 表スペース・データ・ページのヘッダーからの情報を表示します。

HEADER
メディア・ヘッダー情報を表示します。

HEADERONLY
HEADER と同じ情報を表示します。ただし、イメージの先頭から 4 K メディア・ヘッダー情報のみを読み取ります。イメージの妥当性検査は実行しません。

LFH ログ・ファイル・ヘッダー (LFH) データを表示します。

OBJECT
オブジェクト・ヘッダーからの詳細情報を表示します。

PAGECOUNT
イメージの中にある各オブジェクト・タイプのページ数を表示します。

TABLESPACES
コンテナー情報など、イメージ中の表スペースに関する詳細情報を表示します。

TABLESPACESONLY
TABLESPACES と同じ情報を表示しますが、イメージの検証は実行しません。

TABLESPACE
表スペース・バックアップ・イメージだけを組み込みます。

FULL フル・データベース・バックアップ・イメージだけを組み込みます。

NONINCREMENTAL
非増分バックアップ・イメージだけを組み込みます。

INCREMENTAL
増分バックアップ・イメージだけを組み込みます。

DELTA
増分差分バックアップ・イメージだけを組み込みます。

LOADCOPY
ロード・コピー・イメージだけを組み込みます。

LOGS ログ・アーカイブ・イメージだけを組み込みます。

BETWEEN *sn1* **AND** *sn2*
ログ・シーケンス番号 1 とログ・シーケンス番号 2 の間のログの使用を指定します。

CHAIN *n*
使用するログのチェーン ID。

SHOW INACTIVE
非活動化されているバックアップ・オブジェクトを組み込みます。

SUBSET
イメージからファイルにページを抽出します。ページを抽出するには、入力ファイルと出力ファイルが必要です。デフォルトの入力ファイルは extractPage.in という名前です。デフォルトの入力ファイル名は、DB2LISTFILE 環境変数に絶対パスを設定することによりオーバーライドできます。入力ファイルの形式は、次のとおりです。

SMS 表スペースの場合:

```
S <tbspID> <objID> <objType> <startPage> <numPages>
```

DMS 表スペースの場合:

```
D <tbspID> <objType> <startPage> <numPages>
```

注: <objType> が必要なのは、DMS ロード・コピー・イメージを検証する場合だけです。

ログ・ファイルの場合:

```
L <log num> <startPos> <numPages>
```

その他のデータ (初期データなど) の場合:

```
O <objType> <startPos> <numBytes>
```

デフォルトの出力ファイルは extractPage.out です。デフォルトの出力ファイル名は、 DB2EXTRACTFILE 環境変数に絶対パスを設定することによりオーバーライドできます。

TAKEN AT *timestamp*
タイム・スタンプを基準としてバックアップ・イメージを指定します。

KEEP *n*
タイム・スタンプで最新の *n* 個を除き、指定したタイプのすべてのオブジェクトを非活動化します。

OLDER THAN *timestamp* **または** *n* **days**
timestamp または *n* 日より前のタイム・スタンプが付けられているオブジェクトを非活動化することを指定します。

COMPRLIB *decompression-library*
解凍を実行するために使用するライブラリーの名前。この名前は、サーバー上の 1 個のファイルを参照する完全修飾パスでなければなりません。このパラメーターを指定しない場合、 DB2 はイメージ内に格納されているライブラリーの使用を試みます。バックアップが圧縮されていなかった場合、このパラメーターの値は無視されます。指定されたライブラリーをロードできない場合、操作は失敗します。

COMPROPTS *decompression-options*
バイナリー・データのうち、解凍ライブラリーの初期設定ルーチンに渡すブロックを記述します。 DB2 はこのストリングをクライアントからサーバーに直接渡すため、バイト反転やコード・ページ変換の問題がある場合は解凍ライブラリーで処理する必要があります。データ・ブロックの最初の文字が '@' なら、データの残りの部分は、サーバー上に存在するファイルの名前を指定するものとして解釈されます。その場合 DB2 は、データ・ブロックの内容をそのファイルの内容で置き換え、そのようにして得られる新しい値を初期設定ルーチンに渡します。このストリングの最大長は 1024 バイトです。

DATABASE *database_name*
指定したデータベース名に関連したオブジェクトだけを対象にします。

DBPARTITIONNUM *db-partition-number*
指定したデータベース・パーティション番号で作成されたオブジェクトだけを対象にします。

PASSWORD *password*
このノードの TSM クライアント・パスワードを指定します (必要な場合)。データベースが指定されたもののパスワードが提供されない場合には、 *tsm_password* データベース構成パラメーターに指定した値が TSM に渡されます。渡されない場合には、パスワードは使用されません。

NODENAME *node_name*
特定の TSM ノード名に関連したイメージだけを対象にします。

OWNER *owner*
指定した所有者により作成されたオブジェクトだけを対象にします。

WITHOUT PROMPTING
オブジェクトの削除の前に、確認を求めるプロンプトが出ないようにします。

VERBOSE
付加的なファイル情報を表示します。

GRANT ALL / USER *user_name*
現在の TSM ノード上の TSM ファイルに対するアクセス権を、すべてのユーザーまたは指定したユーザーに付与します。アクセス権をユーザーに付与すると、指定されたデータベースに関連する現在のファイルと将来のファイルのすべてのアクセス権を付与することになります。

REVOKE ALL / USER *user_name*
現在の TSM ノード上の TSM ファイルに対するアクセス権を、すべてのユーザーまたは指定したユーザーから削除します。

QUERYACCESS
現在のアクセス・リストを取り出します。ユーザーと TSM ノードのリストが表示されます。

ON ALL / NODENAME *node_name*
アクセス権を変更する TSM ノード。

FOR ALL / DATABASE *database_name*
対象となるデータベース。

例:

1. 以下に示すのは、`db2 backup database rawsampl use tsm` コマンドの出力例です。

```
Backup successful. The timestamp for this backup is : 20031209184503
```

以下に示すのは、バックアップ操作の後で発行された `db2adutl query` コマンドの出力例です。

```
Query for database RAWSAMPL

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20031209184403, Oldest log: S0000050.LOG, Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for RAWSAMPL

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for RAWSAMPL
```

```
Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for RAWSAMPL

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for RAWSAMPL

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for RAWSAMPL

Retrieving LOCAL COPY information.
  No LOCAL COPY images found for RAWSAMPL

Retrieving log archive information.
   Log file: S0000050.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.46.13
   Log file: S0000051.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.46.43
   Log file: S0000052.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.47.12
   Log file: S0000053.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.50.14
   Log file: S0000054.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.50.56
   Log file: S0000055.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.52.39
```

2. 以下に示すのは、 db2adutl delete full taken at 20031209184503 db rawsampl コマンドの出力例です。

```
Query for database RAWSAMPL

Retrieving FULL DATABASE BACKUP information.
  Taken at: 20031209184503  DB Partition Number: 0    Sessions: 1

  Do you want to delete this file (Y/N)? y

    Are you sure (Y/N)? y


Retrieving INCREMENTAL DATABASE  BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for RAWSAMPL

Retrieving DELTA DATABASE  BACKUP information.
  No DELTA DATABASE BACKUP images found for RAWSAMPL
```

以下に示すのは、全バックアップ・イメージを削除した操作の後で発行された db2adutl query コマンドの出力例です。バックアップ・イメージのタイム・スタンプに注意してください。

```
Query for database RAWSAMPL

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20031209184403, Oldest log: S0000050.LOG, Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for RAWSAMPL

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for RAWSAMPL

Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for RAWSAMPL

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for RAWSAMPL
```

```
Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for RAWSAMPL

Retrieving LOCAL COPY information.
  No LOCAL COPY images found for RAWSAMPL

Retrieving log archive information.
   Log file: S0000050.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.46.13
   Log file: S0000051.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.46.43
   Log file: S0000052.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.47.12
   Log file: S0000053.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.50.14
   Log file: S0000054.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.50.56
   Log file: S0000055.LOG, Chain Num: 0, DB Partition Number: 0,
    Taken at 2003-12-09-18.52.39
```

3. 以下に示すのは、db2adutl queryaccess for all コマンドの出力例です。

```
Node                 User                 Database Name      type
------------------------------------------------------------------
bar2                 jchisan              sample             B
<all>                <all>                test               B
------------------------------------------------------------------
Access Types: B - Backup images  L - Logs  A - both
```

使用上の注意:

以下の各グループから 1 つのパラメーターを使用して、何のバックアップ・イメージ・タイプを操作に組み込むかを制限できます。

細分:

- FULL - データベース・バックアップ・イメージだけを組み込みます。
- TABLESPACE - 表スペースのバックアップ・イメージだけを組み込みます。

累積性:

- NONINCREMENTAL - 非増分バックアップ・イメージだけを組み込みます。
- INCREMENTAL - 増分バックアップ・イメージだけを組み込みます。
- DELTA - 増分差分バックアップ・イメージだけを組み込みます。

互換性:

バージョン 8 より前のバージョンとの互換性:

- キーワード DBPARTITIONNUM の代わりに NODE を使用できます。

db2ckbkp - バックアップの検査

このユーティリティーを使用すると、バックアップ・イメージの保全性をテストして、イメージがリストア可能かどうかを判別することができます。また、バックアップ・ヘッダーに保管されているメタデータを表示するために使用することもできます。

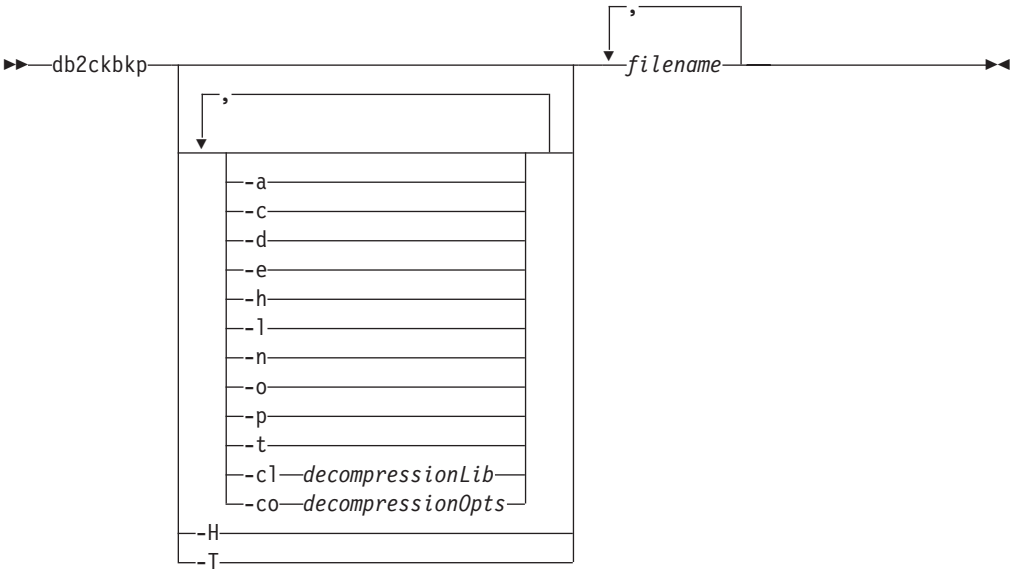
権限:

このユーティリティーにはすべてのユーザーがアクセス可能ですが、イメージ・バックアップに対してこのユーティリティーを実行するには、それらの読み取り許可がなければなりません。

必要な接続:

なし

コマンド構文:



コマンド・パラメーター:

-a 使用可能なすべての情報を表示します。

-c チェックビットおよびチェックサムの結果を表示します。

-cl *decompressionLib*
解凍を実行するために使用するライブラリーの名前。この名前は、サーバー上の 1 個のファイルを参照する完全修飾パスでなければなりません。このパラメーターを指定しない場合、 DB2 はイメージ内に格納されているライブラリーの使用を試みます。バックアップが圧縮されていなかった場合、このパラメーターの値は無視されます。指定されたライブラリーをロードできない場合、操作は失敗します。

-co *decompressionOpts*
バイナリー・データのうち、解凍ライブラリーの初期設定ルーチンに渡すブロックを記述します。 DB2 はこのストリングをクライアントからサーバーに直接渡すため、バイト反転やコード・ページ変換の問題がある場合は解凍ライブラリーで処理する必要があります。データ・ブロックの最初の文字が '@' なら、データの残りの部分は、サーバー上に存在するファイルの名前を指定するものとして解釈されます。その場合 DB2 は、 *string* の内容をそのファイルの内容で置き換え、そのようにして得られる新しい値を初期設定ルーチンに渡します。 string の最大長は 1024 バイトです。

-d DMS 表スペース・データ・ページのヘッダーからの情報を表示します。

-e イメージからファイルにページを抽出します。ページを抽出するには、入力

ファイルと出力ファイルが必要です。デフォルトの入力ファイルは extractPage.in という名前です。デフォルトの入力ファイル名は、DB2LISTFILE 環境変数に絶対パスを設定することによりオーバーライドできます。入力ファイルの形式は、次のとおりです。

SMS 表スペースの場合:

```
S <tbspID> <objID> <objType> <startPage> <numPages>
```

DMS 表スペースの場合:

```
D <tbspID> <objType> <startPage> <numPages>
```

注: <objType> が必要なのは、DMS ロード・コピー・イメージを検証する場合だけです。

ログ・ファイルの場合:

```
L <log num> <startPos> <numPages>
```

その他のデータ (初期データなど) の場合:

```
O <objType> <startPos> <numBytes>
```

デフォルトの出力ファイルは extractPage.out です。デフォルトの出力ファイル名は、 DB2EXTRACTFILE 環境変数に絶対パスを設定することによりオーバーライドできます。

-h メディア・ヘッダー情報を表示します。これには、リストア・ユーティリティーで要求されるイメージの名前およびパスも含みます。

-H -h と同じ情報を表示します。ただし、イメージの先頭から 4K メディア・ヘッダー情報のみを読み取ります。イメージの妥当性検査は実行しません。

注: このオプションは他のオプションと併用できません。

-l ログ・ファイル・ヘッダー (LFH) およびミラー・ログ・ファイル・ヘッダー (MFH) データを表示します。

-n テープ・マウントのプロンプトを出します。 1 つの装置につき 1 つのテープが前提となります。

-o オブジェクト・ヘッダーからの詳細情報を表示します。

-p 各オブジェクト・タイプのページ数を表示します。

-t コンテナー情報など、イメージ中の表スペースに関する詳細情報を表示します。

-T -t と同じ情報を表示しますが、イメージの検証は実行しません。

注: このオプションは他のオプションと併用できません。

filename

バックアップ・イメージ・ファイルの名前。 1 つ以上のファイルを一度に検査できます。

注:

1. 完全バックアップが複数のオブジェクトで構成されている場合には、同時にすべてのオブジェクトを **db2ckbkp** を使用して妥当性検査する場合にのみ、妥当性検査は正常に実行できます。

2. イメージの複数の部分を検査する場合には、最初のバックアップ・イメージ・オブジェクト (.001) を最初に指定しなければなりません。

例:

例 1 (UNIX プラットフォーム)

```
db2ckbkp SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.001
SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.002
SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.003

[1] Buffers processed:  ##
[2] Buffers processed:  ##
[3] Buffers processed:  ##
Image Verification Complete - successful.
```

例 2 (Windows プラットフォーム)

```
db2ckbkp SAMPLE.0¥krodger¥NODE0000¥CATN0000¥19990817¥150714.001
SAMPLE.0¥krodger¥NODE0000¥CATN0000¥19990817¥150714.002
SAMPLE.0¥krodger¥NODE0000¥CATN0000¥19990817¥150714.003

[1] Buffers processed:  ##
[2] Buffers processed:  ##
[3] Buffers processed:  ##
Image Verification Complete - successful.
```

例 3

```
db2ckbkp -h SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001

=====================
MEDIA HEADER REACHED:
=====================
        Server Database Name           -- SAMPLE2
        Server Database Alias          -- SAMPLE2
        Client Database Alias          -- SAMPLE2
        Timestamp                      -- 19990818122909
        Database Partition Number      -- 0
        Instance                       -- krodger
        Sequence Number                -- 1
        Release ID                     -- 900
        Database Seed                  -- 65E0B395
        DB Comment's Codepage (Volume) -- 0
        DB Comment (Volume)            --
        DB Comment's Codepage (System) -- 0
        DB Comment (System)            --
        Authentication Value           -- 255
        Backup Mode                    -- 0
        Include Logs                   -- 0
        Compression                    -- 0
        Backup Type                    -- 0
        Backup Gran.                   -- 0
        Status Flags                   -- 11
        System Cats inc                -- 1
        Catalog Database Partition No. -- 0
        DB Codeset                     -- ISO8859-1
        DB Territory                   --
        LogID                          -- 1074717952
        LogPath                        -- /home/krodger/krodger/NODE0000/
                                          SQL00001/SQLOGDIR
        Backup Buffer Size             -- 4194304
        Number of Sessions             -- 1
        Platform                       -- 0

The proper image file name would be:
```

```
SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001

[1] Buffers processed:  ####
Image Verification Complete - successful.
```

使用上の注意:

1. 複数のセッションを使用してバックアップ・イメージを作成した場合には、**db2ckbkp** は同時にすべてのファイルを検査できます。シーケンス番号 001 のセッションが、最初に指定されるファイルであることを確認してください。
2. このユーティリティーは、テープに保管されているバックアップ・イメージ (変数ブロック・サイズを指定して作成されたイメージは除く) も検査できます。これは、リストア操作の場合のようにテープを準備し、テープ装置名を指定してユーティリティーを起動することにより行えます。たとえば、UNIX ベースのシステムでは以下のようにします。

   ```
   db2ckbkp -h /dev/rmt0
   ```

   Windows では以下のようにします。

   ```
   db2ckbkp -d ¥¥.¥tape1
   ```

3. イメージがテープ装置上にある場合、テープ装置パスを指定します。オプション '-n' を指定しない場合、マウント確認のプロンプトが出されます。テープが複数存在する場合、最初のテープを指定された最初の装置パスにマウントしなければなりません (これは、ヘッダー内の順序 001 のテープです)。

   デフォルトでは、テープ装置が検出されるとテープのマウントを促すプロンプトが出されます。ユーザーは、プロンプトで選択します。以下は、プロンプトとオプションです。 (指定された装置 I は、装置パス /dev/rmt0 上にあります。)

   ```
   Please mount the source media on device /dev/rmt0.
   Continue(c), terminate only this device(d), or abort this tool(t)?
   (c/d/t)
   ```

   指定した装置ごとに、テープの終了時にプロンプトが出されます。

関連資料:

• 275 ページの『db2adutl - TSM 内の DB2 オブジェクトの管理』

db2ckrst - 増分リストア・イメージ順序の検査

データベース履歴を照会して、増分リストアに必要なバックアップ・イメージのタイム・スタンプのリストを生成します。手操作の増分リストアに使用する、単純化されたリストア構文も生成されます。

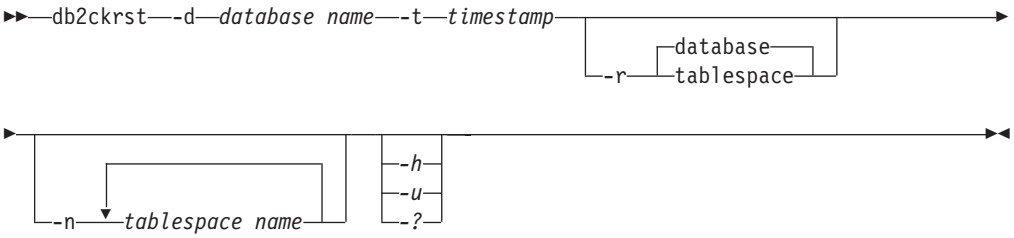
権限:

なし

必要な接続:

なし

コマンド構文:



コマンド・パラメーター:

-d *database name*
リストアされるデータベースの別名を指定します。

-t *timestamp*
増分をリストアするバックアップ・イメージのタイム・スタンプを指定します。

-r 実行するリストアのタイプを指定します。デフォルトはデータベースです。

注: tablespace を選択していながら表スペース名を指定しなかった場合、ユーティリティーは指定のイメージの履歴項目内を探索して、リストアを実行するためにリストされた表スペース名を使用します。

-n *tablespace name*
リストアされる 1 つ以上の表スペースの名前を指定します。

注: データベース・リストア・タイプを選択して、表スペース名のリストを指定した場合、ユーティリティーは指定の表スペース名を使用して tablespace restore を続行します。

-h/-u/-?
ヘルプ情報を表示します。このオプションを指定すると、他のすべてのオプションは無視され、ヘルプ情報だけが表示されます。

例:

```
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
==================================================================
```

使用上の注意:

このユーティリティーを使用するためには、データベース履歴が存在していなければなりません。データベース履歴が存在しない場合は、このユーティリティーを使用する前に、 RESTORE コマンドで HISTORY FILE オプションを指定してください。

PRUNE HISTORY コマンドの FORCE オプションが使用されている場合、データベースの自動増分リストアに必要な項目を削除してしまう可能性があります。手動リストアは正常に動作します。また、このコマンドを使用すると、 dbckrst ユーティリティーが、必要なバックアップ・イメージの完全なチェーンを正しく分析できなくなることもあります。 PRUNE HISTORY コマンドのデフォルトの操作では、必要な項目を削除しないようになっています。 PRUNE HISTORY コマンドの FORCE オプションは使用しないことをお勧めします。

このユーティリティーは、バックアップを記録するための代替手段として使用してはなりません。

関連タスク:

- 31 ページの『増分バックアップ・イメージからのリストア』

関連資料:

- 102 ページの『RESTORE DATABASE』
- 301 ページの『PRUNE HISTORY/LOGFILE』

db2flsn - ログ・シーケンス番号の検出

指定されたログ・シーケンス番号 (LSN) で識別されるログ・レコードを含むファイルの名前を戻します。

権限:

なし

コマンド構文:

コマンド・パラメーター:

-q ログ・ファイル名だけが印刷されます。エラー・メッセージや警告メッセージは印刷されず、状況は戻りコードを介してのみ判別できます。有効なエラー・コードは以下のとおりです。

- -100 無効な入力
- -101 LFH ファイルをオープンできない

- -102 LFH ファイルの読み取りに失敗した
- -103 無効な LFH
- -104 データベースがリカバリー可能でない
- -105 LSN が大きすぎる
- -500 論理エラー

他の有効な戻りコードは以下のとおりです。

- 0 正常な実行
- 99 警告: 結果は、分かっている最後のログ・ファイル・サイズに基づいている

input_LSN
ストリング付きの内部 (6 バイト) 16 進数値を表す 12 文字のストリング。

例:

```
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
```

使用上の注意:

ログ・ヘッダー制御ファイル SQLOGCTL.LFH が現行ディレクトリーになければなりません。このファイルはデータベース・ディレクトリーにあるので、データベース・ディレクトリーからこのツールを実行するか、このツールが実行されるディレクトリーに制御ファイルをコピーすることができます。

このツールは、*logfilsiz* データベース構成パラメーターを使用します。 DB2 は、このパラメーターの最新の 3 つの値と、各 *logfilsiz* 値によって作成された最初のログ・ファイルを記録します。このため、*logfilsiz* が変更されても、ツールは正しく動作することができます。指定された LSN の日付が、記録されている *logfilsiz* の最も古い値の日付より前の場合、ツールはこの値を使用し、警告を戻します。このツールは、UDB バージョン 5.2 より前のデータベース・マネージャーでも使用できます。その場合、正しい結果 (*logfilsiz* の値が変更されない場合に得られる) についても警告が戻されます。

このツールは、リカバリー可能データベースでのみ使用することができます。データベースがリカバリー可能なのは、そのデータベースの構成パラメーター *logarchmeth1* または *logarchmeth2* が OFF 以外の値に設定されている場合です。

db2inidb - ミラー・データベースの初期化

ミラー環境のミラー・データベースを初期化します。ミラー・データベースは、ロールフォワード・ペンディング状態にある 1 次データベースの複製として初期化したり、1 次データベースをリストアするためのバックアップ・イメージとして使用できます。このコマンドはスプリット・ミラー・データベースに対してのみ実行であり、スプリット・ミラーを使用するには、その前にこのコマンドを実行しておく必要があります。

権限:

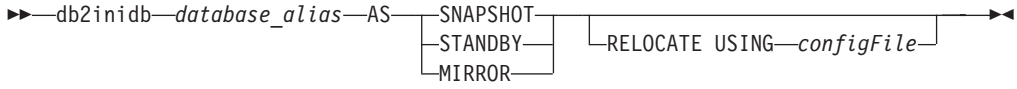
以下のいずれかが必要です。

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

必要な接続:

なし

コマンド構文:



コマンド・パラメーター:

database_alias
初期設定するデータベースの別名を指定します。

SNAPSHOT
ミラー・データベースは、1 次データベースの複製として初期化されることを指定します。

STANDBY
データベースをロールフォワード・ペンディング状態にすることを指定します。

注: 1 次データベースからの新しいログは、フェッチおよびスタンバイ・データベースに適用することが可能です。スタンバイ・データベースは、1 次データベースがダウンした場合に、その代わりに使用できます。

MIRROR
ミラー･データベースを、1 次データベースをリストアするために使用できるバックアップ・イメージとして使用することを指定します。

RELOCATE USING *configFile*
データベースをスナップショット、スタンバイ、またはミラーとして初期化する前に、指定された *configFile* の中でリストされている情報に基づいて、データベース・ファイルを再配置することを指定します。 *configFile* の形式については、 db2relocatedb - データベースの再配置コマンドを参照してください。

使用上の注意:

パーティション・データベース環境では、**db2inidb** は各パーティション上でそれぞれ実行する必要があります。それから、すべてのパーティションのスプリット・ミラーを使用することができます。 **db2inidb** は、**db2_all** コマンドを使用してすべてのパーティションで同時に実行することができます。

しかし、RELOCATE USING オプションを使用する場合は、 **db2_all** コマンドを使用して全パーティションに対して同時に **db2inidb** を実行することはできません。パーティションごとにそれぞれ別個の構成ファイル (変更するパーティションの NODENUM 値が含まれる) を用意する必要があります。たとえば、データベースの名前を変更する場合は、すべてのパーティションが影響を受けることになり、各パーティションごとに別個の構成ファイルを用意して **db2relocatedb** コマンドを実行する必要があります。単一データベース・パーティションに属するコンテナーを移動する場合は、そのパーティション上で一度だけ **db2relocatedb** コマンドを実行することが必要です。

RELOCATE USING *configFile* パラメーターが指定されており、データベースの再配置が正常に実行されたなら、指定された *configFile* はデータベース・ディレクトリーにコピーされ、その名前が db2path.cfg に変更されます。それ以降のクラッシュ・リカバリーまたはロールフォワード・リカバリーにおいて、このファイルは、ログ・ファイルの処理時にコンテナー・パスの名前を変更するために使用されます。

クローン・データベースを初期化している場合、指定された *configFile* は、クラッシュ・リカバリー完了後にデータベース・ディレクトリーから自動的に除去されます。

スタンバイ・データベースまたはミラー･データベースを初期化している場合、指定された *configFile* は、ロールフォワード・リカバリーの完了後またはキャンセル後に、データベース・ディレクトリーから自動的に除去されます。 **db2inidb** 実行後には、新しいコンテナー・パスを db2path.cfg ファイルに追加できます。元のデータベースに対して CREATE 操作または ALTER TABLESPACE 操作を実行し、スタンバイ・データベース上で異なるパスを使用しなければならない場合には、このことが必要になります。

関連タスク:

- 191 ページの『スプリット・ミラーを使用したデータベースのクローン作成』
- 192 ページの『スプリット・ミラーをスタンバイ・データベースとして使用する』
- 194 ページの『スプリット・ミラーをバックアップ・イメージとして使用する』

関連資料:

- 「コマンド・リファレンス」の『db2relocatedb - データベースの再配置コマンド』
- 「*管理ガイド: インプリメンテーション*」の『rah および db2_all コマンドの説明』

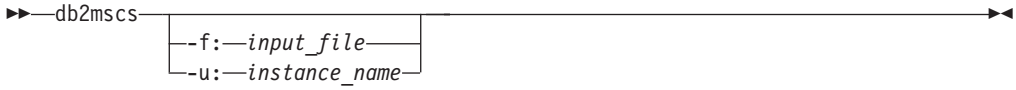
db2mscs - Windows フェイルオーバー・ユーティリティーのセットアップ

Microsoft Cluster Server (MSCS) を使用する Windows で DB2 フェイルオーバーをサポートするためインフラストラクチャーを作成します。このユーティリティーを使用すると、単一パーティション環境とパーティション・データベース環境の両方でフェイルオーバーが可能になります。

権限:

ユーザーは、MSCS クラスター内の各マシンの管理者グループに属するドメイン・ユーザー・アカウントにログオンする必要があります。

コマンド構文:



コマンド・パラメーター:

-f:input_file
MSCS ユーティリティーによって使用される DB2MSCS.CFG 入力ファイルを指定します。このパラメーターが指定されない場合、DB2MSCS ユーティリティーは、現行のディレクトリーにある DB2MSCS.CFG ファイルを読み取ります。

-u:instance_name
このオプションを使用すると、db2mscs 操作を取り消し、インスタンスを instance_name で指定された非 MSCS インスタンスに復帰させることができます。

使用上の注意:

DB2MSCS ユーティリティーは、非 MSCS インスタンスを MSCS インスタンスにトランスフォームするのに使用できる、スタンドアロン型のコマンド行ユーティリティーです。このユーティリティーは、すべての MSCS グループ、リソース、およびリソース依存関係を作成します。また、このユーティリティーは、 Windows レジストリーに保管されているすべての DB2 情報をレジストリーのクラスター部分にコピーし、インスタンス・ディレクトリーを共有クラスター・ディスクに移動します。 DB2MSCS ユーティリティーは、ユーザーから渡される構成ファイルを、クラスターのセットアップ方法を指定する入力として受け取ります。 DB2MSCS.CFG ファイルは、ASCII テキスト・ファイルで、 DB2MSCS ユーティリティーが読み取るパラメーターが含まれています。各入力パラメーターは、それぞれ別々の行に PARAMETER_KEYWORD=parameter_value というフォーマットで指定します。たとえば、次のようにします。

```
CLUSTER_NAME=FINANCE
GROUP_NAME=DB2 Group
IP_ADDRESS=9.21.22.89
```

DB2 インストール・ディレクトリーの CFG サブディレクトリーには、2 つのサンプル構成ファイルがあります。 1 つは DB2MSCS.EE というファイルで、これは単一

パーティション・データベース環境の例になっています。もう 1 つは DB2MSCS.EEE で、これは、パーティション・データベース環境の例です。

DB2MSCS.CFG ファイルのパラメーターは次のようになっています。

DB2_INSTANCE
DB2 インスタンスの名前。このパラメーターは、グローバルな有効範囲を持っているため、 DB2MSCS.CFG ファイル内で一度だけ指定します。

DAS_INSTANCE
DB2 Administration Server インスタンスの名前。このパラメーターは、MSCS 環境で稼働するように DB2 Administration Server を移行する場合に指定します。このパラメーターは、グローバルな有効範囲を持っているため、 DB2MSCS.CFG ファイル内で一度だけ指定します。

CLUSTER_NAME
MSCS クラスターの名前。この行より後に指定されるすべてのリソースは、別の CLUSTER_NAME パラメーターが指定されるまでこのクラスターに作成されます。

DB2_LOGON_USERNAME
DB2 サービス用ドメイン・アカウントのユーザー名 (*domain¥user* のように指定)。このパラメーターは、グローバルな有効範囲を持っているため、DB2MSCS.CFG ファイル内で一度だけ指定します。

DB2_LOGON_PASSWORD
DB2 サービス用ドメイン・アカウントのパスワード。このパラメーターは、グローバルな有効範囲を持っているため、 DB2MSCS.CFG ファイル内で一度だけ指定します。

GROUP_NAME
MSCS グループの名前。このパラメーターが指定されたときに、指定された名前の MSCS グループが存在していない場合は、そのグループが新しく作成されます。むろん、グループがすでに存在している場合は、そのグループがターゲット・グループになります。このパラメーターより後に指定された MSCS リソースは、別の GROUP_NAME パラメーターが指定されるまで、このグループに作成または移動されます。このパラメーターは、各グループにつき 1 つ指定してください。

DB2_NODE
現行の MSCS グループに組み込むデータベース・パーティション・サーバー (またはデータベース・パーティション) のパーティション番号。同じマシン上に複数の論理データベース・パーティションが存在する場合は、データベース・パーティションごとに別々の DB2_NODE パラメーターが必要です。 DB2 リソースが正しい MSCS グループに作成されるよう、このパラメーターは GROUP_NAME パラメーターの後に指定してください。このパラメーターは、複数パーティション・データベース・システムに必要です。

IP_NAME
IP アドレス・リソースの名前。 IP_NAME の値は任意ですが、クラスター内でユニークな値でなければなりません。このパラメーターが指定されると、IP アドレス・タイプの MSCS リソースが作成されます。このパラメー

ターは、リモート TCP/IP 接続で必要です。単一パーティション環境の場合、このパラメーターはオプションです。推奨されている名前は、その IP アドレスに対応するホスト名です。

IP_ADDRESS
前述の IP_NAME パラメーターで指定した IP リソースの TCP/IP アドレス。 IP_NAME パラメーターを指定するときはこのパラメーターが必要です。新しい、ネットワーク内のいかなるマシンでも使用されていない IP アドレスが使用されます。

IP_SUBNET
前述の IP_NAME パラメーターで指定した IP リソースの TCP/IP サブネット・マスク。 IP_NAME パラメーターを指定するときはこのパラメーターが必要です。

IP_NETWORK
前述の IP アドレス・リソースが属している MSCS ネットワークの名前。このパラメーターはオプションです。このパラメーターが指定されない場合は、システムが最初に検出した MSCS ネットワークが使用されます。MSCS ネットワークの名前は、「クラスター管理 (Cluster Administrator)」の Networks の分岐の下に示されている通りに、正確に入力してください。

注: 前述の 4 つの IP キーワードは、IP アドレス・リソースの作成に使用されます。

NETNAME_NAME
ネットワーク名リソースの名前。このパラメーターは、ネットワーク名リソースを作成する場合に指定してください。単一パーティション・データベース環境では、このパラメーターはオプションです。しかし、パーティション・データベース環境でマシンを所有するインスタンスには、必ずこのパラメーターを指定する必要があります。

NETNAME_VALUE
ネットワーク名リソースの値。 NETNAME_NAME パラメーターを指定する場合には、このパラメーターの指定が必要です。

NETNAME_DEPENDENCY
ネットワーク名リソースが依存する IP リソースの名前。各ネットワーク名リソースには、必ず IP アドレス・リソースへの依存関係が必要です。このパラメーターはオプションです。このパラメーターが指定されない場合、ネットワーク名リソースは、グループ内の最初の IP リソースに依存するようになります。

SERVICE_DISPLAY_NAME
汎用サービス・リソースの表示名。このパラメーターは、汎用サービス・リソースを作成する場合に指定します。

SERVICE_NAME
汎用サービス・リソースのサービス名。 SERVICE_DISPLAY_NAME パラメーターを指定する場合には、このパラメーターの指定が必要です。

SERVICE_STARTUP
汎用サービス・リソース用のオプショナル始動パラメーター。

DISK_NAME

現行グループに移動させる物理ディスク・リソースの名前。必要な分だけのディスク・リソースを指定してください。ディスク・リソースは、あらかじめ存在するものでなければなりません。 DB2MSCS ユーティリティーがフェイルオーバー・サポート用に DB2 インスタンスを構成する場合は、グループ内の最初の MSCS ディスクにインスタンス・ディレクトリーがコピーされます。インスタンス・ディレクトリーに別の MSCS ディスクを指定する場合は、INSTPROF_DISK パラメーターを使用してください。なお、ディスク名は、「クラスター管理 (Cluster Administrator)」で示されている通りに、正確に入力してください。

INSTPROF_DISK

DB2 インスタンス・ディレクトリーを入れる MSCS ディスクを指定するための、オプショナル・パラメーター。このパラメーターが指定されない場合、DB2MSCS ユーティリティーは、同じグループに属する最初のディスクを使用します。

INSTPROF_PATH

インスタンス・ディレクトリーのコピー先の正確なパスを指定するための、オプショナル・パラメーター。 IPSHAdisks、つまり ServerRAID Netfinity ディスク・リソース (例、INSTPROF_PATH=p:¥db2profs) を使用する場合には、必ずこのパラメーターを指定する必要があります。なお、INSTPROF_PATH と INSTPROF_DISK の両方が指定されている場合は、INSTPROF_PATH の方が優先順位が上です。

TARGET_DRVMAP_DISK

複数パーティション・データベース・システムのためのデータベース・ドライブ・マッピングのターゲット MSCS ディスクを指定する、オプショナル・パラメーター。このパラメーターは、データベースの作成コマンドで指定されたドライブからディスクをマップすることにより、データベースが作成されるディスクを指定します。このパラメーターを指定しない場合は、DB2DRVMP ユーティリティーを使用して手動でデータベース・ドライブ・マッピングを登録する必要があります。

DB2_FALLBACK

DB2 リソースがオフラインにされたときにアプリケーションを強制的にオフにするかどうかを制御する、オプショナル・パラメーター。このパラメーターが指定されなければ、DB2_FALLBACK の設定は YES になります。アプリケーションを強制的にオフにしない場合は、DB2_FALLBACK を NO に設定してください。

db2rfpen - ロールフォワード・ペンディング状態のリセット

データベースをロールフォワード・ペンディング状態にします。高可用性災害時リカバリー (HADR) を使用している場合、データベースは標準データベースにリセットされます。

権限:

なし

必要な接続:

なし

コマンド構文:

コマンド・パラメーター:

database_alias
ロールフォワード・ペンディング状態にするデータベースの名前を指定します。高可用性災害時リカバリー (HADR) を使用している場合、データベースは標準データベースにリセットされます。

-log logfile_path
ログ・ファイル・パスを指定します。

関連概念:

- 201 ページの『高可用性災害時リカバリーの概要』

CLP コマンド

ARCHIVE LOG

リカバリー可能データベースのアクティブ・ログ・ファイルをクローズし、切り捨てます。

権限:

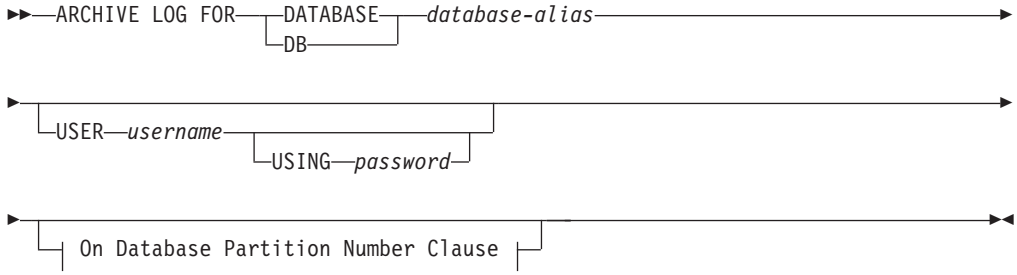
以下のいずれかが必要です。

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

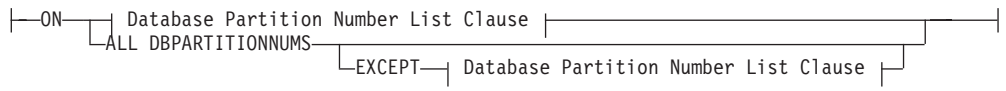
必要な接続:

なし。このコマンドは、コマンドの持続期間の間、データベース接続を確立します。

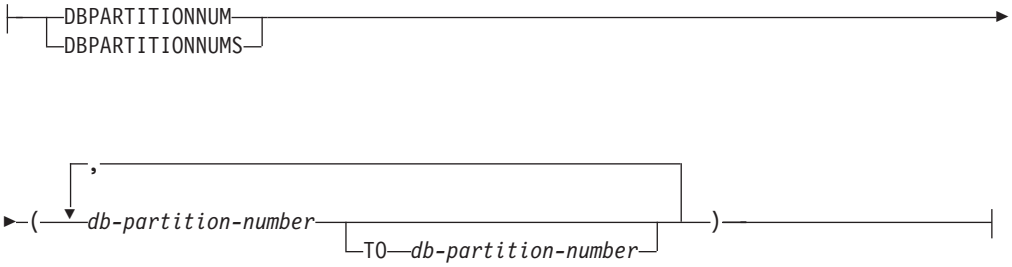
コマンド構文:



On Database Partition Number Clause:



Database Partition Number List Clause:



コマンド・パラメーター:

DATABASE database-alias
アーカイブするアクティブ・ログを持つデータベースの別名を指定します。

USER username
接続を試みるユーザー名を識別します。

USING password
ユーザー名を認証するためのパスワードを指定します。

ON ALL DBPARTITIONNUMS
コマンドを db2nodes.cfg ファイルにあるすべてのデータベース・パーティションで発行することを指定します。データベース・パーティション番号文節が指定されていない場合、これがデフォルトです。

EXCEPT
コマンドを、データベース・パーティション番号リストに指定されたデータベース・パーティションを除く、 db2nodes.cfg ファイルにあるすべてのデータベース・パーティションで発行することを指定します。

ON DBPARTITIONNUM/ON DBPARTITIONNUMS
指定されたデータベースのログをデータベース・パーティションのセットでアーカイブすることを指定します。

db-partition-number
データベース・パーティション番号リスト内のデータベース・パーティション番号を指定します。

TO db-partition-number
ログをアーカイブするデータベース・パーティションの範囲を指定するときに使用されます。指定された最初のデータベース・パーティション番号から 2 番目のデータベース・パーティション番号までのすべてのデータベース・パーティションがデータベース・パーティション・リストに含まれます。

使用上の注意:

このコマンドは、ある時点までのログ・ファイルの完全なセットを収集するために使用できます。次に、そのログ・ファイルを使用してスタンバイ・データベースを更新することができます。

このコマンドは、起動側アプリケーションまたはシェルに、指定されたデータベースへのデータベース接続がないときにしか実行できません。これにより、コミットされていないトランザクションでユーザーがコマンドを実行するのを防ぎます。実際に、ARCHIVE LOG コマンドは、ユーザーの未完了のトランザクションをコミッ

トしません。起動側アプリケーションまたはシェルに、指定されたデータベースへのデータベース接続がすでに存在している場合は、コマンドは終了してエラーを戻します。このコマンドを実行したときに、指定されたデータベースで進行中のトランザクションが別のアプリケーションにあった場合には、コマンドがログ・バッファーをディスクにフラッシュするため、パフォーマンスがやや低下する可能性があります。ログ・レコードをバッファーに書き込む別のトランザクションは、フラッシュが完了するまで待機しなければなりません。

パーティション・データベース環境で使用する場合は、データベース・パーティション番号文節を使用してデータベース・パーティションのサブセットを指定できます。データベース・パーティション番号文節が指定されていない場合、このコマンドのデフォルトの動作は、クローズしてすべてのデータベース・パーティションのアクティブ・ログをアーカイブすることです。

このコマンドを使用すると、アクティブ・ログ・ファイルの切り捨てのために、アーカイブ・ログ・スペースの部分を使い果たします。アクティブ・ログ・スペースは、切り捨てられたログが非アクティブになると前のサイズを再開します。このコマンドを頻繁に使用すると、トランザクションで使用できるアクティブ・ログ・スペースの量が劇的に削減される場合があります。

互換性:

バージョン 8 より前のバージョンとの互換性:

- キーワード DBPARTITIONNUM の代わりに NODE を使用できます。
- キーワード DBPARTITIONNUMS の代わりに NODES を使用できます。

INITIALIZE TAPE

Windows NT ベースのオペレーティング・システムで実行する場合、 DB2 は、ストリーミング磁気テープ装置へのバックアップおよびリストア操作をサポートしています。このコマンドは、テープを初期化するのに使います。

権限:

なし

必要な接続:

なし

コマンド構文:

コマンド・パラメーター:

ON device
有効なテープ装置名を指定します。デフォルトは ¥¥.¥TAPE0 です。

USING blksize

装置のブロック・サイズを指定します (バイト単位)。値が装置のブロック・サイズとしてサポートされている範囲内にあれば、装置は指定されたそのブロック・サイズで初期化されます。

注: BACKUP DATABASE コマンドおよび RESTORE DATABASE コマンドで指定されるバッファー・サイズは、ここで指定されるブロック・サイズで割り切れなければなりません。

このパラメーターに値を指定しなかった場合、装置はデフォルトのブロック・サイズで初期化されます。値ゼロを指定した場合は、装置は可変長のブロック・サイズで初期化されます。装置が可変長のブロック・モードをサポートしていない場合は、エラーが戻されます。

関連資料:

- 77 ページの『BACKUP DATABASE』
- 102 ページの『RESTORE DATABASE』
- 302 ページの『REWIND TAPE』
- 303 ページの『SET TAPE POSITION』

LIST HISTORY

履歴ファイルの中の項目のリストを表示します。履歴ファイルには、リカバリーと管理のさまざまなイベントの記録が含まれています。リカバリー・イベントには、データベース・レベルおよび表スペース・レベルのフル・バックアップ、増分バックアップ、リストア、およびロールフォワード操作が含まれます。さらにログ記録されるイベントには、表スペースの作成、変更、ドロップ、または名前変更、統計実行、表の再編成、表のドロップ、およびロードが含まれます。

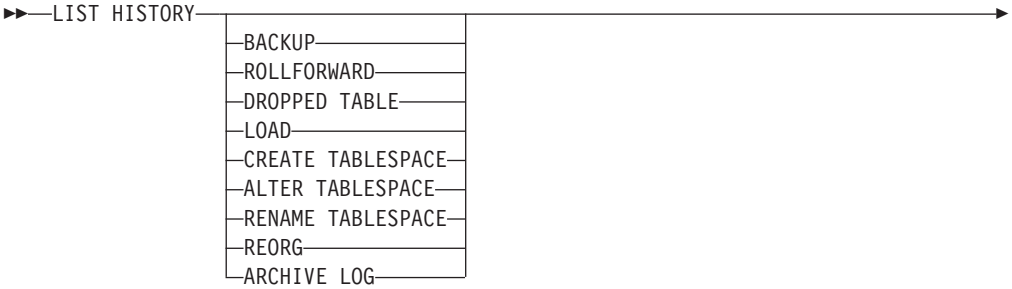
権限:

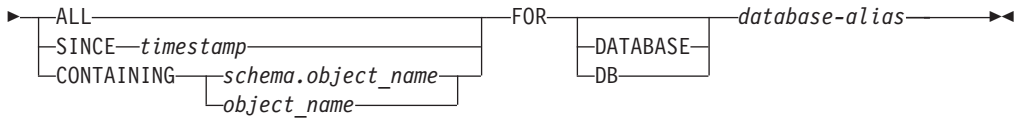
なし

必要な接続:

インスタンス。これに対してこのコマンドを実行するためには、いずれかのリモート・データベースにアタッチしていなければなりません。ローカル・データベースの場合、明示的なアタッチは必要ありません。

コマンド構文:





コマンド・パラメーター:

HISTORY
現在履歴ファイルの中に記録されているイベントのすべてのリストを表示します。

BACKUP
バックアップ操作およびリストア操作をリストします。

ROLLFORWARD
ロールフォワード操作をリストします。

DROPPED TABLE
ドロップした表レコードをリストします。ドロップした表レコードが作成されるのは、表がドロップされ、それを含む表スペースについて DROPPED TABLE RECOVERY オプションが有効になっている場合だけです。

LOAD ロード操作をリストします。

CREATE TABLESPACE
表スペースの作成およびドロップ操作をリストします。

RENAME TABLESPACE
表スペースの名前変更操作をリストします。

REORG
再編成操作のリストを表示します。

ALTER TABLESPACE
表スペースの変更操作をリストします。

ARCHIVE LOG
アーカイブ・ログ操作と、それによってアーカイブされるログのリストをリストします。

ALL 履歴ファイルのうち、指定したタイプのすべての項目のリストを表示します。

SINCE timestamp
完全なタイム・スタンプ (形式は *yyyymmddhhmmss*)、または先頭の接頭部 (最小値は *yyyy*) を指定できます。指定したタイム・スタンプ以降のタイム・スタンプの項目のリストを表示します。

CONTAINING schema.object_name
この修飾名は表を固有に識別します。

CONTAINING object_name
この非修飾名は表スペースを固有に識別します。

FOR DATABASE database-alias
リカバリー履歴ファイルをリスト表示するデータベースを指定します。

例:

```
db2 list history since 19980201 for sample
db2 list history backup containing userspace1 for sample
db2 list history dropped table all for db sample
```

使用上の注意:

このコマンドによって生成されるレポートには、以下の記号が含まれます。

操作
- A - 表スペースの作成
- B - バックアップ
- C - ロードのコピー・ファイル取得
- D - ドロップされた表
- F - ロールフォワード
- G - 表の再編成
- L - ロード
- N - 表スペースの名前変更
- O - 表スペースのドロップ
- Q - 静止
- R - リストア
- T - 表スペースの変更
- U - アンロード
- X - アーカイブ・ログ

タイプ

アーカイブ・ログ・タイプ:
- P - 1 次ログ・パス
- M - 2 次 (ミラー) ログ・パス
- F - フェイルオーバー・アーカイブ・パス
- 1 - 1 次ログ・アーカイブ・メソッド
- 2 - 2 次ログ・アーカイブ・メソッド

バックアップ・タイプ:

- F - オフライン
- N - オンライン
- I - 増分オフライン
- O - 増分オンライン
- D - デルタ・オフライン
- E - デルタ・オンライン

ロールフォワード・タイプ:

- E - ログの最後
- P - ポイント・イン・タイム

ロード・タイプ:

- I - 挿入
- R - 置換

表スペースの変更タイプ

- C - コンテナーの追加
- R - 再調整

静止タイプ:

- S - 静止共有
- U - 静止更新
- X - 静止排他
- Z - 静止リセット

PRUNE HISTORY/LOGFILE

リカバリー履歴ファイルから項目を削除したり、アクティブ・ログ・ファイル・パスからログ・ファイルを削除したりするのに使用します。リカバリー履歴ファイルからの項目の削除は、ファイルが非常に大きくなったり保存期間が長くなっている場合に必要になることがあります。

権限:

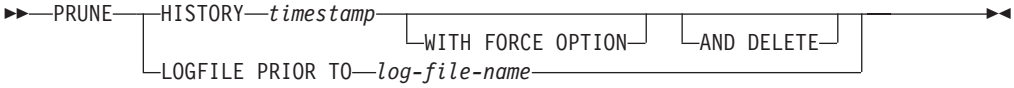
以下のいずれかが必要です。

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

必要な接続:

データベース

コマンド構文:



コマンド・パラメーター:

HISTORY timestamp
削除される、リカバリー履歴ファイルにある項目範囲を識別します。完全なタイム・スタンプ (書式 *yyyymmddhhmmss*)、または最初の接頭部 (最小値 *yyyy*) を指定できます。提供されているそのタイム・スタンプ以下のタイム・スタンプ付きのすべての項目は、リカバリー履歴ファイルから削除されます。

WITH FORCE OPTION
最新のリストア・セットのいくつかの項目がファイルから削除されるとしても、指定したタイム・スタンプに従って項目を整理することを指定します。リストア・セットは、バックアップ・イメージのすべてのリストアを含む、最新の全データベース・バックアップです。このパラメーターを指定しない場合、バックアップ・イメージ転送からのすべての項目は履歴の中で保守されます。

AND DELETE
履歴ファイルの項目を削除する際に、関連するログ・アーカイブを (ロケーション情報に基づいて) 物理的に削除することを指定します。このオプションは、ログ・アーカイブが不要になった場合に、アーカイブ・ストレージ・スペースが確実にリカバリーされるようにする上で、特に有用です。

注: ユーザー出口プログラムによりログをアーカイブしている場合は、このオプションを使用してそれらのログを削除することはできません。

LOGFILE PRIOR TO log-file-name
ログ・ファイル名を表すストリング (例: S0000100.LOG) を指定します。指定したログ・ファイルより前のすべてのログ・ファイルは削除されます。指

定したログ・ファイルそのものは削除されません。 LOGRETAIN データベース構成パラメーターは、 RECOVERY または CAPTURE に設定する必要があります。

例:

前に行われた、すべてのリストア、ロード、表スペース、バックアップ、および全部のデータベース・バックアップのための項目を除去するには、リカバリー履歴ファイルから 1994 12.1 を含んで、次のように入力してください。

```
db2 prune history 199412
```

注: 199412 は 19941201000000 と解釈されます。

使用上の注意:

FORCE オプションが使用されている場合、データベースの自動増分リストアに必要な項目を削除してしまう可能性があります。手動リストアは正常に動作します。また、このコマンドを使用すると、 **dbckrst** ユーティリティーが、必要なバックアップ・イメージの完全なチェーンを正しく分析できなくなることもあります。 PRUNE HISTORY コマンドを FORCE オプションなしで使用した場合、必要な項目が削除されることはありません。

履歴ファイルからバックアップ項目を整理すると、 DB2 Data Links Manager サーバー上にある関連ファイルのバックアップが削除されます。

REWIND TAPE

Windows NT ベースのオペレーティング・システムで実行する場合、 DB2 は、ストリーミング磁気テープ装置へのバックアップおよびリストア操作をサポートしています。このコマンドを使用してテープを巻き戻します。

権限:

なし

必要な接続:

なし

コマンド構文:

コマンド・パラメーター:

ON device
有効なテープ装置名を指定します。デフォルトは ¥¥.¥TAPE0 です。

関連資料:

- 297 ページの『INITIALIZE TAPE』
- 303 ページの『SET TAPE POSITION』

SET TAPE POSITION

Windows NT ベースのオペレーティング・システムで実行する場合、 DB2 は、ストリーミング磁気テープ装置へのバックアップおよびリストア操作をサポートしています。このコマンドを使用して、テープの位置決めを行います。

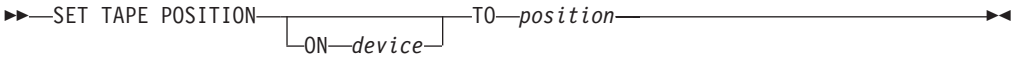
権限:

なし

必要な接続:

なし

コマンド構文:



コマンド・パラメーター:

ON device
有効なテープ装置名を指定します。デフォルトは ¥¥.¥TAPE0 です。

TO position
テープ位置のマークを指定します。 DB2 (Windows 版) は、バックアップ・イメージの度にテープ・マークを書き込みます。値 1 は 1 番目の位置、 2 は 2 番目の位置、以下同じ手順で指定します。テープがテープ・マーク 1 に位置している場合、たとえば、アーカイブ 2 がリストアされる位置に置かれます。

関連資料:

- 297 ページの『INITIALIZE TAPE』
- 302 ページの『REWIND TAPE』

UPDATE HISTORY FILE

履歴ファイル項目にあるロケーション、装置タイプ、またはコメントを更新します。

権限:

以下のいずれかが必要です。

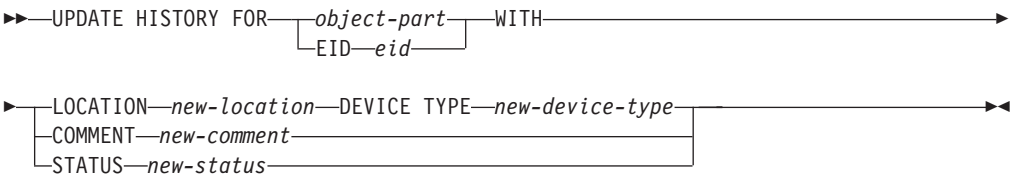
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

必要な接続:

データベース

コマンド構文:

UPDATE HISTORY FILE



コマンド・パラメーター:

EID *eid*
履歴項目 ID。

FOR *object-part*
イメージのバックアップまたはコピーの ID を指定します。この ID は、タイム・スタンプと 001 から 999 までのオプションのシーケンス番号で構成されます。

LOCATION *new-location*
バックアップ・イメージの新しい物理ロケーションを指定します。このパラメーターの解釈は装置タイプに依存します。

DEVICE TYPE *new-device-type*
バックアップ・イメージを保管する新しい装置タイプを指定します。有効な装置タイプは次のとおりです。

D ディスク

K ディスケット

T テープ

A TSM

U ユーザー出口

P パイプ

N Null 装置

X XBSA

Q SQL ステートメント

O その他

COMMENT *new-comment*
項目を記述する新しい注釈を指定します。

STATUS *new-status*
項目の新しい状況。有効な値は以下のとおりです。

A 項目をアクティブとしてマークします。

I 項目を非アクティブとしてマークします。

例:

1997 年 4 月 13 日午前 10 時 00 分に取った全データベース・バックアップの履歴ファイルを更新するには、次のように入力します。

```
db2 update history for 19970413100000001 with
   location /backup/dbbackup.1 device type d
```

使用上の注意:

データベース履歴ファイルの主な用途は情報を記録することですが、履歴に含まれるデータは、自動リストア操作で直接に使用されます。 AUTOMATIC オプションを指定したリストアにおいては、リストア・ユーティリティーによりバックアップ・イメージとそのロケーションの履歴が参照および使用されることにより、自動リストア要求が処理されます。自動リストア機能を使用する場合に、バックアップ・イメージが作成されて以来に再配置されているなら、現在のロケーションを反映するよう、データベース履歴レコードを更新することをお勧めします。データベース履歴の中のバックアップ・イメージのロケーションが更新されない場合、自動リストア処理においてはバックアップ・イメージを見つけることができなくなりますが、手動リストア・コマンドは正常に使用できます。

関連資料:

• 301 ページの『PRUNE HISTORY/LOGFILE』

付録 D. 追加の API および関連データ構造

この付録では、このマニュアルでは詳しく説明されていない、リカバリー関連の API およびそのデータ構造を紹介します。

db2ArchiveLog - アクティブ・ログのアーカイブ

リカバリー可能データベースのアクティブ・ログ・ファイルをクローズし、切り捨てます。ユーザー出口が使用可能な場合、アーカイブ要求を発行します。

権限:

以下のいずれかが必要です。

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

必要な接続:

この API を呼び出せば、指定したデータベースへの接続が自動的に確立されます。指定したデータベースへの接続がすでに存在している場合、API はエラーを戻します。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2ArchiveLog */
/* ... */
SQL_API_RC SQL_API_FN
   db2ArchiveLog (
  db2Uint32 version,
  void *pDB2ArchiveLogStruct,
  struct sqlca *pSqlca);

typedef struct
{
   char                *piDatabaseAlias;
  char                        *piUserName;
  char                        *piPassword;
   db2Uint16           iAllNodeFlag;
   db2Uint16           iNumNodes;
  SQL_PDB_NODE_TYPE           *piNodeList;
   db2Uint32           iOptions;
} db2ArchiveLogStruct
/* ... */
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2gArchiveLog */
/* ... */
```

```
SQL_API_RC SQL_API_FN
db2gArchiveLog (
  db2Uint32 version,
  void *pDB2ArchiveLogStruct,
  struct sqlca *pSqlca);

typedef struct
{
  db2Uint32                   iAliasLen;
  db2Uint32                   iUserNameLen;
  db2Uint32                   iPasswordLen;
   char                *piDatabaseAlias;
  char                        *piUserName;
  char                        *piPassword;
   db2Uint16           iAllNodeFlag;
   db2Uint16           iNumNodes;
  SQL_PDB_NODE_TYPE           *piNodeList;
   db2Uint32           iOptions;
} db2ArchiveLogStruct
/* ... */
```

API パラメーター:

version
入力。 2 番目のパラメーター *pDB2ArchiveLogStruct* として渡される変数のバージョンおよびリリース・レベルを指定します。

pDB2ArchiveLogStruct
入力。 *db2ArchiveLogStruct* 構造を指すポインター。

pSqlca
出力。 *sqlca* 構造を指すポインター。

iAliasLen
入力。データベースの別名の長さを示す 4 バイトの符号なし整数 (バイト単位)。

iUserNameLen
入力。ユーザー名の長さを示す 4 バイトの符号なし整数 (バイト単位) です。ユーザー名が使用されていない場合は、ゼロに設定してください。

iPasswordLen
入力。パスワードの長さを示す 4 バイトの符号なし整数 (バイト単位) です。パスワードが使用されていない場合は、ゼロに設定してください。

piDatabaseAlias
入力。アクティブ・ログをアーカイブする対象のデータベースのデータベース別名 (システム・データベース・ディレクトリーにカタログされている) を含むストリングです。

piUserName
入力。接続の試行時に使用されるユーザー名を含むストリングを指定します。

piPassword
入力。接続の試行時に使用されるパスワードを含むストリングです。

iAllNodeFlag

パーティション・データベース環境のみ。入力。操作を db2nodes.cfg ファイルでリストされているすべてのノードに適用するかどうかを示すフラグです。有効な値は以下のとおりです。

DB2ARCHIVELOG_NODE_LIST
piNodeList で渡されたノード・リスト内でノードに適用されます。

DB2ARCHIVELOG_ALL_NODES
すべてのノードに適用されます。*piNodeList* は NULL にしてください。これはデフォルト値です。

DB2ARCHIVELOG_ALL_EXCEPT
piNodeList で渡されたノード・リスト内で指定されたノードを除き、すべてのノードに適用されます。

iNumNodes

パーティション・データベース環境のみ。入力。 *piNodeList* 配列内のノードの数を指定します。

piNodeList

パーティション・データベース環境のみ。入力。アーカイブ・ログ操作を適用する対象のノード番号の配列を指すポインターです。

iOptions

入力。将来の利用のために予約されています。

関連資料:

• 295 ページの『ARCHIVE LOG』

db2HistoryCloseScan - 履歴ファイルのスキャンのクローズ

履歴ファイルのスキャンを終了し、スキャンに必要だった DB2 リソースを解放します。この API は、db2HistoryOpenScan の正常呼び出しの後でなければ使用できません。

権限:

なし

必要な接続:

インスタンス。この API を呼び出す前に、sqleatin を呼び出す必要はありません。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2HistoryCloseScan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2HistoryCloseScan (
```

```
    db2Uint32 version,
    void *piHandle,
    struct sqlca *pSqlca);
/* ... */
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryCloseScan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2GenHistoryCloseScan (
    db2Uint32 version,
    void *piHandle,
    struct sqlca *pSqlca);
/* ... */
```

API パラメーター:

version
入力。 2 番目のパラメーター *piHandle* のバージョンとリリースのレベルを指定します。

piHandle
入力。 db2HistoryOpenScan によって戻された、スキャン・アクセス用のハンドルを指すポインターを指定します。

pSqlca
出力。*sqlca* 構造へのポインター。

REXX API 構文:

```
CLOSE RECOVERY HISTORY FILE :scanid
```

REXX API パラメーター:

scanid
OPEN RECOVERY HISTORY FILE SCAN から戻されたスキャン ID を含むホスト変数。

使用上の注意:

履歴ファイル API の使用の詳細については、『db2HistoryOpenScan』を参照してください。

関連資料:

- 321 ページの『db2Prune - 履歴ファイルの整理』
- 318 ページの『db2HistoryUpdate - 履歴ファイルの更新』
- 313 ページの『db2HistoryOpenScan - 履歴ファイルのスキャンのオープン』
- 311 ページの『db2HistoryGetEntry - 履歴ファイルの次項目の入手』
- 「*管理 API リファレンス*」の『SQLCA』

関連サンプル:

- 『dbrecov.sqc -- How to recover a database (C)』
- 『dbrecov.sqC -- How to recover a database (C++)』

db2HistoryGetEntry - 履歴ファイルの次項目の入手

履歴ファイルの次項目を入手します。この API は、db2HistoryOpenScan の正常呼び出しの後でなければ使用できません。

権限:

なし

必要な接続:

インスタンス。この API を呼び出す前に、sqleatin を呼び出す必要はありません。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2HistoryGetEntry */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryGetEntry (
    db2Uint32 version,
    void *pDB2HistoryGetEntryStruct,
    struct sqlca *pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData *pioHistData
} db2HistoryGetEntryStruct;
/* ... */
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryGetEntry */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryGetEntry (
    db2Uint32 version,
    void *pDB2GenHistoryGetEntryStruct,
    struct sqlca *pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData *pioHistData
} db2GenHistoryGetEntryStruct;
/* ... */
```

API パラメーター:

version

入力。 2 番目のパラメーター *pDB2HistoryGetEntryStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2HistoryGetEntryStruct
入力。 *db2HistoryGetEntryStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

iHandle
入力。 db2HistoryOpenScan によって戻された、スキャン・アクセス用のハンドルが含まれます。

iCallerAction
入力。実行するアクションのタイプを指定します。有効な値 (db2ApiDf で定義) は、以下のとおりです。

DB2HISTORY_GET_ENTRY
次項目を入手しますが、コマンド・データはありません。

DB2HISTORY_GET_DDL
直前のフェッチからコマンド・データだけを入手します。

DB2HISTORY_GET_ALL
すべてのデータを含め、次項目を入手します。

pioHistData
入力。 *db2HistData* 構造を指すポインター。

REXX API 構文:

```
GET RECOVERY HISTORY FILE ENTRY :scanid [USING :value]
```

REXX API パラメーター:

scanid
OPEN RECOVERY HISTORY FILE SCAN から戻されたスキャン ID を含むホスト変数。

value 履歴ファイルの項目情報が戻されるコンパウンド REXX ホスト変数。以下の項目において、XXX はホスト変数名を表しています。

XXX.0	変数内の第 1 レベル・エレメントの数 (常に 15)
XXX.1	表スペース・エレメントの数
XXX.2	使用された表スペース・エレメントの数
XXX.3	OPERATION (実行された操作のタイプ)
XXX.4	OBJECT (操作の細分性)
XXX.5	OBJECT_PART (タイム・スタンプおよびシーケンス番号)
XXX.6	OPTYPE (操作の修飾子)
XXX.7	DEVICE_TYPE (使用された装置のタイプ)
XXX.8	FIRST_LOG (最初のログ ID)
XXX.9	LAST_LOG (現行のログ ID)
XXX.10	BACKUP_ID (バックアップ用の ID)
XXX.11	SCHEMA (表名の修飾子)
XXX.12	TABLE_NAME (ロードされた表の名前)

XXX.13.0	NUM_OF_TABLESPACES (バックアップまたはリストアに関係した表スペースの数)
XXX.13.1	最初にバックアップまたはリストアされた表スペースの名前
XXX.13.2	2 番目にバックアップまたはリストアされた表スペースの名前
XXX.13.3	以下同じ
XXX.14	LOCATION (バックアップまたはコピーが保管されている場所)
XXX.15	COMMENT (項目を記述するテキスト)

使用上の注意:

戻されるレコードは、db2HistoryOpenScan への呼び出しで指定した値を使用して選択されます。

履歴ファイル API の使用の詳細については、『db2HistoryOpenScan』を参照してください。

関連資料:

- 321 ページの『db2Prune - 履歴ファイルの整理』
- 318 ページの『db2HistoryUpdate - 履歴ファイルの更新』
- 313 ページの『db2HistoryOpenScan - 履歴ファイルのスキャンのオープン』
- 309 ページの『db2HistoryCloseScan - 履歴ファイルのスキャンのクローズ』
- 「*管理 API リファレンス*」の『SQLCA』
- 333 ページの『db2HistData』

関連サンプル:

- 『dbrecov.sqc -- How to recover a database (C)』
- 『dbrecov.sqC -- How to recover a database (C++)』

db2HistoryOpenScan - 履歴ファイルのスキャンのオープン

履歴ファイルのスキャンを開始します。

権限:

なし

必要な接続:

インスタンス。データベースがリモートとしてカタログされている場合には、この API を呼び出す前に sqleatin を呼び出してください。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2HistoryOpenScan */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryOpenScan (
    db2Uint32 version,
    void *pDB2HistoryOpenStruct,
    struct sqlca *pSqlca);

typedef struct
{
  char *piDatabaseAlias,
  char *piTimestamp,
  char *piObjectName,
  db2Uint32 oNumRows,
  db2Uint32 oMaxTbspaces,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2HistoryOpenStruct;
/* ... */
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryOpenScan */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryOpenScan (
    db2Uint32 version,
    void *pDB2GenHistoryOpenStruct,
    struct sqlca *pSqlca);

typedef struct
{
  char *piDatabaseAlias,
  char *piTimestamp,
  char *piObjectName,
  db2Uint32 oNumRows,
  db2Uint32 oMaxTbspaces,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2GenHistoryOpenStruct;
/* ... */
```

API パラメーター:

version
入力。 2 番目のパラメーター *pDB2HistoryOpenStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2HistoryOpenStruct
入力。 *db2HistoryOpenStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

piDatabaseAlias
入力。データベース別名を含むストリングを指すポインター。

piTimestamp
入力。レコードの選択に使用されるタイム・スタンプを指定するストリングを指すポインター。この値と同じタイム・スタンプまたはこの値より大きいタイム・スタンプを持つレコードが選択されます。このパラメーターを

NULL に設定するか、ゼロを指すようにすれば、タイム・スタンプを用いての項目のフィルターを実行しないようすることができます。

piObjectName
入力。レコードの選択に使用されるオブジェクト名を指定するストリングを指すポインター。オブジェクトとして表または表スペースを使用できます。オブジェクトが表の場合、表の完全修飾名を指定する必要があります。このパラメーターを NULL に設定するか、ゼロを指すようにすれば、オブジェクト名を用いての項目のフィルターを実行しないようにすることができます。

oNumRows
出力。 API からの戻り時に、このパラメーターには、一致した履歴ファイルの項目の数が入ります。

oMaxTbspaces
出力。任意の履歴項目で保管された表スペース名の最大数。

iCallerAction
入力。実行するアクションのタイプを指定します。有効な値 (db2ApiDf で定義) は、以下のとおりです。

DB2HISTORY_LIST_HISTORY
現在履歴ファイルの中に記録されているイベントのすべてのリストを表示します。

DB2HISTORY_LIST_BACKUP
バックアップ操作およびリストア操作をリストします。

DB2HISTORY_LIST_ROLLFORWARD
ロールフォワード操作をリストします。

DB2HISTORY_LIST_DROPPED_TABLE
ドロップした表レコードをリストします。項目と関連付けられた DDL フィールドは戻されません。項目の DDL 情報を検索するには、この項目が取り出された直後に、呼び出しアクション DB2HISTORY_GET_DDL を指定して db2HistoryGetEntry を呼び出す必要があります。

DB2HISTORY_LIST_LOAD
ロード操作をリストします。

DB2HISTORY_LIST_CRT_TABLESPACE
表スペースの作成およびドロップ操作をリストします。

DB2HISTORY_LIST_REN_TABLESPACE
表スペースの名前変更操作をリストします。

DB2HISTORY_LIST_ALT_TABLESPACE
表スペースの変更操作をリストします。項目と関連付けられた DDL フィールドは戻されません。項目の DDL 情報を検索するには、この項目が取り出された直後に、呼び出しアクション DB2HISTORY_GET_DDL を指定して db2HistoryGetEntry を呼び出す必要があります。

DB2HISTORY_LIST_REORG
REORGANIZE TABLE 操作をリストします。この値は、現在サポートされていません。

oHandle
出力。 API からの戻り時に、このパラメーターには、スキャン・アクセス用のハンドルが入れられます。このハンドルは、その後 db2HistoryGetEntry および db2HistoryCloseScan で使用されます。

REXX API 構文:

```
OPEN [BACKUP] RECOVERY HISTORY FILE FOR database_alias
[OBJECT objname] [TIMESTAMP :timestamp]
USING :value
```

REXX API パラメーター:

database_alias
履歴ファイルがリストされる、データベースの別名です。

objname
レコードの選択に使用されるオブジェクト名を指定します。オブジェクトとして表または表スペースを使用できます。オブジェクトが表の場合、表の完全修飾名を指定する必要があります。このパラメーターを NULL に設定すれば、 *objname* を使用しての項目のフィルターを実行しないようにすることができます。

timestamp
レコードの選択に使用されるタイム・スタンプを指定します。この値と同じタイム・スタンプまたはこの値より大きいタイム・スタンプを持つレコードが選択されます。このパラメーターを NULL に設定すれば、 *timestamp* を使用しての項目のフィルターを実行しないようにすることができます。

value 履歴ファイル情報が戻されるコンパウンド REXX ホスト変数です。以下の項目において、XXX はホスト変数名を表しています。

XXX.0 変数内のエレメント数 (常に 2)。

XXX.1 将来のスキャン・アクセスに使用される ID (ハンドル)。

XXX.2 一致した履歴ファイル項目の数

使用上の注意:

タイム・スタンプ、オブジェクト名、および呼び出し側アクションの組み合わせを使用して、レコードをフィルターにかけることもできます。指定したすべてのフィルターを通過するレコードだけが戻されます。

オブジェクト名のフィルター操作の結果は、指定した値によって異なります。

- 表を指定した場合、ロード操作に関するレコードだけが戻されます (これが履歴ファイル内の表に関する唯一の情報であるため)。
- 表スペースを指定した場合、その表スペースに関するバックアップ、リストア操作、およびロード操作に関するレコードが戻されます。

注: 表のレコードを戻すには、その表を *schema.tablename* として指定しなければなりません。 *tablename* を指定した場合は、表スペースのレコードしか戻されません。

1 つのプロセスで、最大 8 つの履歴ファイル・スキャンが許可されています。

履歴ファイル中のすべての項目をリストする場合、通常のアプリケーションであれば、以下のステップを実行します。

1. db2HistoryOpenScan を呼び出す *oNumRows* が戻されます。
2. *db2HistData* 構造に、 *n* 個の *oTablespace* フィールド用のスペースを割り振る。 *n* は任意の数値です。
3. *db2HistData* 構造の *iDB2NumTablespace* フィールドを *n* に設定する。
4. ループの中で、以下を実行してください。
   - db2HistoryGetEntry を呼び出して履歴ファイルから取り出しを行います。
   - db2HistoryGetEntry によって、 SQL_RC_OK という SQLCODE が戻されたら、 *db2HistData* 構造の *sqld* フィールドを使用して、戻された表スペース項目の数を判別します。
   - db2HistoryGetEntry によって、 SQLUH_SQLUHINFO_VARS_WARNING という SQLCODE が戻された場合は、 DB2 が戻そうとしている表スペースのために十分なスペースが割り振られていません。この場合は、スペースをいったん解放し、 *db2HistData* 構造に *oDB2UsedTablespace* の表スペース項目にとって十分なスペースを割り振り直し、 *iDB2NumTablespace* を *oDB2UsedTablespace* に設定してください。
   - db2HistoryGetEntry によって SQLE_RC_NOMORE という SQLCODE が戻された場合は、すべての履歴ファイル項目が検索されています。
   - 他の SQLCODE は、特定の問題が生じたことを示します。その指示に従ってください。
5. すべての情報の取り出しが終了したら、db2HistoryCloseScan を呼び出して、db2HistoryOpenScan の呼び出しに伴って割り振られたリソースを解放します。

db2HistData 構造の、 *n* 個の *oTablespace* フィールド用のスペースに必要とされるメモリーの量を判別しやすくするため、 (sqlutil で定義された) マクロ SQLUHINFOSIZE (*n*) が用意されています。

関連資料:

- 321 ページの『db2Prune - 履歴ファイルの整理』
- 318 ページの『db2HistoryUpdate - 履歴ファイルの更新』
- 311 ページの『db2HistoryGetEntry - 履歴ファイルの次項目の入手』
- 309 ページの『db2HistoryCloseScan - 履歴ファイルのスキャンのクローズ』
- 「*管理 API リファレンス*」の『SQLCA』

関連サンプル:

- 『dbrecov.sqc -- How to recover a database (C)』
- 『dbrecov.sqC -- How to recover a database (C++)』

db2HistoryUpdate - 履歴ファイルの更新

履歴ファイル項目にあるロケーション、装置タイプ、またはコメントを更新します。

権限:

以下のいずれかが必要です。

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

必要な接続:

データベース。デフォルトのデータベース以外のデータベースの履歴ファイル内にある項目を更新する場合は、この API を呼び出す前に、そのデータベースへの接続を確立しておく必要があります。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2HistoryUpdate */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryUpdate (
    db2Uint32 version,
    void *pDB2HistoryUpdateStruct,
    struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2HistoryUpdateStruct
{
   char                                 *piNewLocation;
   char                                 *piNewDeviceType;
   char                                 *piNewComment;
   char                                 *piNewStatus;
   db2HistoryEID                        iEID;
} db2HistoryUpdateStruct;

/* Structure db2HistoryEID */
typedef SQL_STRUCTURE db2HistoryEID
{
  SQL_PDB_NODE_TYPE ioNode;
   db2Uint32         ioHID;
} db2HistoryEID;

/* ... */
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2gHistoryUpdate */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryUpdate (
    db2Uint32 version,
    void *pDB2GenHistoryUpdateStruct,
    struct sqlca *pSqlca);
```

```
typedef SQL_STRUCTURE db2gHistoryUpdateStruct
{
   char                                *piNewLocation;
   char                                *piNewDeviceType;
   char                                *piNewComment;
   char                                *piNewStatus;
   db2Uint32                           iNewLocationLen;
   db2Uint32                           iNewDeviceLen;
   db2Uint32                           iNewCommentLen;
   db2Uint32                           iNewStatusLen;
   db2HistoryEID                       iEID;
} db2gHistoryUpdateStruct;
/* ... */
```

API パラメーター:

version
入力。 2 番目のパラメーター *pDB2HistoryUpdateStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2HistoryUpdateStruct
入力。 *db2HistoryUpdateStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

piNewLocation
入力。バックアップ、リストア、またはロード・コピー・イメージ用の新規ロケーションを指定するストリングを指すポインター。このパラメーターを NULL に設定するか、ゼロを指すようにすれば、値は変更されません。

piNewDeviceType
入力。バックアップ、リストア、またはロード・コピー・イメージを格納するための新規装置タイプを指定するストリングを指すポインター。このパラメーターを NULL に設定するか、ゼロを指すようにすれば、値は変更されません。

piNewComment
入力。項目について説明する新規のコメントを指定するストリングを指すポインター。このパラメーターを NULL に設定するか、ゼロを指すようにすれば、コメントは変更されません。

piNewStatus
入力。項目の新規の状況タイプを指定するストリングを指すポインター。このパラメーターを NULL に設定するか、ゼロを指すようにすれば、状況は変更されません。

iNewLocationLen
入力。 piNewLocationLen フィールドの長さ。

iNewDeviceLen
入力。 piNewDeviceLen フィールドの長さ。

iNewCommentLen
入力。 piNewCommentLen フィールドの長さ。

iNewStatusLen
入力。 piNewStatusLen フィールドの長さ。

iEID 入力。履歴ファイルの特定の項目を更新するときに使用できる、ユニークな ID です。

ioNode
このパラメーターは、入力パラメーターまたは出力パラメーターのどちらにでも使用できます。

ノード番号を示します。

ioHID このパラメーターは、入力パラメーターまたは出力パラメーターのどちらにでも使用できます。

ローカル履歴ファイルの項目 ID を示します。

REXX API 構文:

```
UPDATE RECOVERY HISTORY USING :value
```

REXX API パラメーター:

value 履歴ファイル項目の新規ロケーションに関する情報を含む、コンパウンド REXX ホスト変数です。以下の項目において、XXX はホスト変数名を表しています。

XXX.0 変数内のエレメント数 (必ず 1 ～ 4)

XXX.1 OBJECT_PART (タイム・スタンプとシーケンス番号 001 ～ 999)

XXX.2 バックアップまたはコピー・イメージの新規ロケーション (このパラメーターはオプションです)

XXX.3 バックアップまたはコピー・イメージの保管に使用される新規装置 (このパラメーターはオプションです)

XXX.4
新規コメント (このパラメーターはオプションです)

使用上の注意:

この API は更新関数であり、変更前の情報はすべて新しい情報に置き換えられ、再作成することができなくなります。これらの変更は記録されません。

データベース履歴ファイルの主な用途は情報を記録することですが、履歴に含まれるデータは、自動リストア操作で直接に使用されます。 AUTOMATIC オプションを指定したリストアにおいては、リストア・ユーティリティーによりバックアップ・イメージとそのロケーションの履歴が参照および使用されることにより、自動リストア要求が処理されます。自動リストア機能を使用する場合に、バックアップ・イメージが作成されて以来に再配置されているなら、現在のロケーションを反映するよう、データベース履歴レコードを更新することをお勧めします。データベース履歴の中のバックアップ・イメージのロケーションが更新されない場合、自動リストア処理においてはバックアップ・イメージを見つけることができなくなりますが、手動リストア・コマンドは正常に使用できます。

関連資料:

- 153 ページの『db2Rollforward - データベースのロールフォワード』
- 321 ページの『db2Prune - 履歴ファイルの整理』

- 313 ページの『db2HistoryOpenScan - 履歴ファイルのスキャンのオープン』
- 311 ページの『db2HistoryGetEntry - 履歴ファイルの次項目の入手』
- 309 ページの『db2HistoryCloseScan - 履歴ファイルのスキャンのクローズ』
- 「*管理 API リファレンス*」の『SQLCA』
- 303 ページの『UPDATE HISTORY FILE』
- 83 ページの『db2Backup - データベースのバックアップ』

関連サンプル:

- 『dbrecov.sqc -- How to recover a database (C)』
- 『dbrecov.sqC -- How to recover a database (C++)』

db2Prune - 履歴ファイルの整理

履歴ファイルから項目を削除するか、アクティブ・ログ・パスからログ・ファイルを削除します。

権限:

以下のいずれかが必要です。

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

必要な接続:

データベース。省略時データベース以外のデータベースの履歴ファイルから項目を削除する場合は、この API を呼び出す前に、そのデータベースへの接続を確立しておく必要があります。

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2Prune */
/* ... */
SQL_API_RC SQL_API_FN
  db2Prune (
    db2Uint32 version,
    void *pDB2PruneStruct,
    struct sqlca *pSqlca);

typedef struct
{
  char *piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2PruneStruct;
/* ... */
```

汎用 API 構文:

```
/* File: db2ApiDf.h */
/* API: db2GenPrune */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenPrune (
    db2Uint32 version,
    void *pDB2GenPruneStruct,
    struct sqlca *pSqlca);

typedef struct
{
  db2Uint32 iStringLen;
  char *piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2GenPruneStruct;
/* ... */
```

API パラメーター:

version
入力。 2 番目のパラメーター *pDB2PruneStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2PruneStruct
入力。 *db2PruneStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

iStringLen
入力。 *piString* の長さ (バイト単位) を指定します。

piString
入力。タイム・スタンプまたはログ・シーケンス番号 (LSN) を指定するストリングを指すポインターです。タイム・スタンプまたはその一部 (最小値は *yyyy*、つまり年) が、削除対象のレコードの選択に使用されます。タイム・スタンプと等しいかまたはタイム・スタンプよりも小さい、すべての項目が削除されます。必ず有効なタイム・スタンプを指定するようにしてください。 NULL パラメーターを指定しても省略時の動作はありません。

このパラメーターは、LSN を渡すときにも使用できるので、非アクティブ・ログの整理が可能です。

iEID 入力。履歴ファイルから単一の項目の整理をするときに使用できる、ユニークな ID を示します。

iCallerAction
入力。実行するアクションのタイプを指定します。有効な値 (db2ApiDf.h で定義) は、以下のとおりです。

DB2PRUNE_ACTION_HISTORY
履歴ファイルの項目を削除します。

DB2PRUNE_ACTION_LOG
アクティブ・ログ・パスからログ・ファイルを削除します。

iOptions
入力。有効な値 (db2ApiDf.h で定義) は、以下のとおりです。

DB2PRUNE_OPTION_FORCE
最後のバックアップの削除を強制します。

DB2PRUNE_OPTION_DELETE
履歴ファイルから整理されるログ・ファイルを削除します。

DB2PRUNE_OPTION_LSNSTRING
piString の値を LSN に指定します。これは、呼び出し側アクション DB2PRUNE_ACTION_LOG が指定されている場合に使用します。

REXX API 構文:

```
PRUNE RECOVERY HISTORY BEFORE :timestamp [WITH FORCE OPTION]
```

REXX API パラメーター:

timestamp
タイム・スタンプを含むホスト変数を示します。指定されたタイム・スタンプと等しいかまたは小さいタイム・スタンプを持つすべての項目が、履歴ファイルから削除されます。

WITH FORCE OPTION
指定した場合、最新のリストア・セット中の一部の項目がファイルから削除されることになる場合でも、指定されたタイム・スタンプに従って履歴ファイルの項目が削除されます。指定しない場合、入力時に指定したタイム・スタンプより小さいか等しい場合でも、最新のリストア・セットが保持されます。

使用上の注意:

履歴ファイルの項目を削除しても、実際のバックアップ、またはロード・ファイルは削除されません。それらのファイルを削除したい場合には、それを手作業で行って、それらのファイルが記憶媒体上で使用しているスペースを解放する必要があります。

注意:
最新の全データベース・バックアップを媒体から削除する (さらに、履歴ファイルから項目が削除される) 場合、すべての表スペース (カタログ表スペースおよびユーザー表スペースを含む) のバックアップを取るようにしてください。そのことを怠ると、データベースが回復不能になったり、データベース内のユーザー・データの一部が失われたりするおそれがあります。

関連資料:

- 318 ページの『db2HistoryUpdate - 履歴ファイルの更新』
- 313 ページの『db2HistoryOpenScan - 履歴ファイルのスキャンのオープン』
- 311 ページの『db2HistoryGetEntry - 履歴ファイルの次項目の入手』
- 309 ページの『db2HistoryCloseScan - 履歴ファイルのスキャンのクローズ』
- 「*管理 API リファレンス*」の『SQLCA』

関連サンプル:

- 『dbrecov.sqc -- How to recover a database (C)』
- 『dbrecov.sqC -- How to recover a database (C++)』

db2ReadLogNoConn - データベース接続なしのログの読み取り

ログ・レコードを DB2 UDB データベース・ログから抽出し、現在のログ状態の情報をログ・マネージャーに照会します。この API の使用に先立ち、db2ReadLogNoConnInit を使用して、この API に入力パラメーターとして渡されるメモリーを割り振ります。この API の使用後は、db2ReadLogNoConnTerm を使用して、メモリーを割り振り解除します。

権限:

なし

必要な接続:

なし

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2ReadLogNoConn */
/* ... */
SQL_API_RC SQL_API_FN
db2ReadLogNoConn (
  db2Uint32 versionNumber,
  void *pDB2ReadLogNoConnStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnStruct
{
  db2Uint32                   iCallerAction;
  SQLU_LSN                    *piStartLSN;
  SQLU_LSN                    *piEndLSN;
  char                        *poLogBuffer;
  db2Uint32                   iLogBufferSize;
  char                        *piReadLogMemPtr;
  db2ReadLogNoConnInfoStruct  *poReadLogInfo;
} db2ReadLogNoConnStruct;

typedef SQL_STRUCTURE db2ReadLogNoConnInfoStruct
{
  SQLU_LSN                    firstAvailableLSN;
  SQLU_LSN                    firstReadLSN;
  SQLU_LSN                    nextStartLSN;
  db2Uint32                   logRecsWritten;
  db2Uint32                   logBytesWritten;
  db2Uint32                   lastLogFullyRead;
  db2TimeOfLog                currentTimeValue;
} db2ReadLogNoConnInfoStruct;

/* ... */
```

API パラメーター:

versionNumber
入力。 2 番目のパラメーター *pDB2ReadLogNoConnStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2ReadLogNoConnStruct
入力。 *db2ReadLogNoConnStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

iCallerAction
入力。実行するアクションを指定します。有効な値は以下のとおりです。

DB2READLOG_READ
開始ログ・シーケンス番号から終了ログ・シーケンス番号までのデータベース・ログを読み取り、この範囲内にあるログ・レコードを戻します。

DB2READLOG_READ_SINGLE
開始ログ・シーケンス番号によって識別される単一のログ・レコード (伝搬可能または伝搬不可のいずれでも) を読み取ります。

DB2READLOG_QUERY
データベース・ログを照会します。照会した結果は、db2ReadLogNoConnInfoStruct 構造を介して戻されます。

piStartLSN
入力。開始ログ・シーケンス番号は、ログの読み取りを開始する相対バイト・アドレスを指定します。この値は、実際のログ・レコードの始まりでなければなりません。

piEndLSN
入力。終了ログ・シーケンス番号は、ログの読み取りを終了する相対バイト・アドレスを指定します。この値は、*piStartLsn* の値より大きくなければなりません。実際のログ・レコードの終わりである必要はありません。

poLogBuffer
出力。指定した範囲内で読み取られた、伝搬可能なすべてのログ・レコードが順番に格納されるバッファー。このバッファーは、単一のログ・レコードを保持するのに十分な大きさでなければなりません。目安として、このバッファーは最低限 32 バイトでなければなりません。最大サイズは、要求された範囲のサイズによって異なってきます。バッファー内の各ログ・レコードには、接頭部として 6 バイトのログ・シーケンス番号 (LSN) が付けられます。これは、次のログ・レコードの LSN を示します。

iLogBufferSize
入力。バイト単位でログ・バッファーのサイズを指定します。

piReadLogMemPtr
入力。初期設定呼び出しで割り振られた、サイズ iReadLogMemoryLimit のメモリー・ブロック。このメモリーには、呼び出しのたびに API が必要とする永続データが含まれています。このメモリー・ブロックは、どのような方法であっても呼び出し側は再割り振りまたは変更してはなりません。

poReadLogInfo
出力。 *db2ReadLogNoConnInfoStruct* 構造を指すポインター。

firstAvailableLSN
使用可能なログ内で最初の使用可能な LSN。

firstReadLSN
この呼び出しでの最初の LSN 読み取り。

nextStartLSN
次の読み取り可能な LSN。

logRecsWritten
ログ・バッファー・フィールド *poLogBuffer* に書き込まれるログ・レコードの数。

logBytesWritten
ログ・バッファー・フィールド *poLogBuffer* に書き込まれるログ・バッファー・フィールドのバイト数。

lastLogFullyRead
読み取られて完了する最後のログ・ファイルを示す数。

使用上の注意:

db2ReadLogNoConn API は、db2ReadLogNoConnInit API を使用して割り振られるメモリー・ブロックを必要とします。メモリー・ブロックは、入力パラメーターとして、続くすべての db2ReadLogNoConn API に渡されなければならず、変更してはなりません。

ログの順次読み取りを要求する場合、API はログ・シーケンス番号 (LSN) の範囲および割り振られたメモリーを必要とします。 API は、初期設定されたときに指定されたフィルター・オプションおよび LSN 範囲に基づいて、ログ・レコードの順序を戻します。照会を要求する場合、ログ情報の読み取り構造に、読み取りの呼び出しで使用される有効な開始 LSN が入ります。読み取りでの終了 LSN として使用される値は、次のうちの 1 つになります。

- 呼び出し側が指定した startLSN の値より大きい値。
- 非同期ログ読み取りプログラムで、使用可能なログの終わりとして解釈される FFFF FFFF FFFF。

開始および終了 LSN の範囲内で読み取られた伝搬可能なログ・レコードは、ログ・バッファーに戻されます。ログ・レコードには、それの LSN は含まれません。 LSN は、バッファーの中で、実際のログ・レコードの前に付けられます。db2ReadLogNoConn によって戻されるさまざまな DB2 UDB ログ・レコードの詳細については、 『DB2 UDB ログ・レコード』セクションで説明しています。

最初の読み取りの後、次の順番のログ・レコードを読み取るには、db2ReadLogNoConnInfoStruct で戻された nextStartLSN 値を使用します。この新しい開始 LSN と有効な終了 LSN を使用して呼び出しをもう一度サブミットすると、次のレコード・ブロックが読み取られます。 SQLU_RLOG_READ_TO_CURRENT の sqlca コードは、使用可能なログ・ファイルが最後まで読み取られたことを示します。

この API がもう使用されない場合、db2ReadLogNoConnTerm を使用してメモリーを終了します。

関連資料:

- 「*管理 API リファレンス*」の『SQLCA』

- 327 ページの『db2ReadLogNoConnInit - データベース接続なしのログ読み取りの初期設定』
- 329 ページの『db2ReadLogNoConnTerm - データベース接続なしのログ読み取りの終了』

db2ReadLogNoConnInit - データベース接続なしのログ読み取りの初期設定

db2ReadLogNoConn によって使用されるメモリーを割り振り、ログ・レコードを DB2 UDB データベース・ログから抽出します。また、ログ・マネージャーを照会して現行のログ状態に関する情報を取得できるようにします。

権限:

なし

必要な接続:

なし

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2ReadLogNoConnInit */
/* ... */
SQL_API_RC SQL_API_FN
db2ReadLogNoConnInit (
  db2Uint32      versionNumber,
  void * pDB2ReadLogNoConnInitStruct,
  struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnInitStruct
{
  db2Uint32                   iFilterOption;
  char                        *piLogFilePath;
  char                        *piOverflowLogPath;
  db2Uint32                   iRetrieveLogs;
  char                        *piDatabaseName;
  char                        *piNodeName;
  db2Uint32                   iReadLogMemoryLimit;
  char                        **poReadLogMemPtr;
} db2ReadLogNoConnInitStruct;
/* ... */
```

API パラメーター:

versionNumber
入力。 2 番目のパラメーター *pDB2ReadLogNoConnInitStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2ReadLogNoConnInitStruct
入力。 *db2ReadLogNoConnInitStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

iFilterOption
入力。ログ・レコードを読み取るときに使用されるログ・レコード・フィルターのレベルを指定します。有効な値は以下のとおりです。

DB2READLOG_FILTER_OFF
指定された LSN 範囲内ですべてのログ・レコードを読み取ります。

DB2READLOG_FILTER_ON
伝搬可能とマークされた LSN 範囲内でログ・レコードのみを読み取ります。これは非同期ログ読み取り API の基本的な動作です。

piLogFilePath
入力。読み取られるログ・ファイルがある場所のパス。

piOverflowLogPath
入力。読み取られるログ・ファイルがある場所の代替パス。

iRetrieveLogs
入力。ログ・ファイル・パス、またはオーバーフロー・ログ・パスのどちらでも検索できないログ・ファイルを検索するために、ユーザー出口を呼び出すかどうかを指定するオプション。有効な値は以下のとおりです。

DB2READLOG_RETRIEVE_OFF
欠落したログ・ファイルを検索するために、ユーザー出口は呼び出しません。

DB2READLOG_RETRIEVE_LOGPATH
欠落したログ・ファイルを検索するために、指定されたログ・ファイル・パスにユーザー出口を呼び出します。

DB2READLOG_RETRIEVE_OVERFLOW
欠落したログ・ファイルを検索するために、指定されたオーバーフロー・ログ・パスにユーザー出口を呼び出します。

piDatabaseName
入力。読み取り中のリカバリー・ログを所有するデータベースの名前。これは、上記の検索オプションが指定された場合に必要です。

piNodeName
入力。読み取り中のリカバリー・ログを所有するノードの名前。これは、上記の検索オプションが指定された場合に必要です。

iReadLogMemoryLimit
入力。 API が内部に割り振る最大バイト数。

poReadLogMemPtr
出力。 API が割り振った、サイズ *iReadLogMemoryLimit* のメモリーのブロック。このメモリーには、呼び出しのたびに API が必要とする永続データが含まれています。このメモリー・ブロックは、どのような方法であっても呼び出し側は再割り振りまたは変更してはなりません。

使用上の注意:

db2ReadLogNoConnInit によって初期設定されたメモリーは変更してはなりません。

db2ReadLogNoConn がもう使用されない場合、db2ReadLogNoConnTerm を呼び出し、 db2ReadLogNoConnInit によって初期設定されたメモリーを割り振り解除します。

関連資料:

- 「*管理 API リファレンス*」の『SQLCA』
- 324 ページの『db2ReadLogNoConn - データベース接続なしのログの読み取り』
- 329 ページの『db2ReadLogNoConnTerm - データベース接続なしのログ読み取りの終了』

db2ReadLogNoConnTerm - データベース接続なしのログ読み取りの終了

本来は db2ReadLogNoConnInit によって初期設定され、db2ReadLogNoConn によって使用されるメモリーを割り振り解除します。

権限:

なし

必要な接続:

なし

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2ReadLogNoConnTerm */
/* ... */
SQL_API_RC SQL_API_FN
db2ReadLogNoConnTerm (
  db2Uint32      versionNumber,
  void * pDB2ReadLogNoConnTermStruct,
  struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnTermStruct
{
  char                         **poReadLogMemPtr;
} db2ReadLogNoConnTermStruct;
/* ... */
```

API パラメーター:

versionNumber
入力。 2 番目のパラメーター *pDB2ReadLogNoConnTermStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2ReadLogNoConnTermStruct
入力。 *db2ReadLogNoConnTermStruct* 構造を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

poReadLogMemPtr
出力。初期設定呼び出しで割り振られたメモリーのブロックを指すポインター。このポインターは解放され、NULL に設定されます。

関連資料:

- 「*管理 API リファレンス*」の『SQLCA』
- 324 ページの『db2ReadLogNoConn - データベース接続なしのログの読み取り』
- 327 ページの『db2ReadLogNoConnInit - データベース接続なしのログ読み取りの初期設定』

db2ReadLog - ログの非同期読み取り

現行のログ状態に関する情報を入手するために、ログ・レコードを DB2 UDB データベース・ログおよびログ・マネージャーから抽出します。この API を使用できるのは、リカバリー可能データベースの場合だけです。データベースがリカバリー可能なのは、データベースが RECOVERY に設定された *logretain*、または ON に設定された *userexit* で構成されている場合です。

権限:

以下のいずれかが必要です。

- *sysadm*
- *dbadm*

必要な接続:

データベース

API 組み込みファイル:

db2ApiDf.h

C API 構文:

```
/* File: db2ApiDf.h */
/* API: db2ReadLog */
/* ... */
SQL_API_RC SQL_API_FN
  db2ReadLog (
    db2Uint32 versionNumber,
    void *pDB2ReadLogStruct,
    struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2ReadLogStruct
{
  db2Uint32                   iCallerAction;
  SQLU_LSN                    *piStartLSN;
  SQLU_LSN                    *piEndLSN;
  char                        *poLogBuffer;
  db2Uint32                   iLogBufferSize;
  db2Uint32                   iFilterOption;
  db2ReadLogInfoStruct        *poReadLogInfo;

typedef SQL_STRUCTURE db2ReadLogInfoStruct
{
```

```
  SQLU_LSN                    initialLSN;
  SQLU_LSN                    firstReadLSN;
  SQLU_LSN                    nextStartLSN;
  db2Uint32                   logRecsWritten;
  db2Uint32                   logBytesWritten;
  SQLU_LSN                    firstReusedLSN;
  db2Uint32                   timeOfLSNReuse;
  db2TimeOfLog                currentTimeValue;
} db2ReadLogInfoStruct;

typedef SQL_STRUCTURE db2TimeOfLog
{
  db2Uint32                   seconds;
  db2Uint32                   accuracy;
} db2TimeOfLog;
/* ... */
```

API パラメーター:

versionNumber
入力。 2 番目のパラメーター *pDB2ReadLogStruct* として渡される構造のバージョンとリリースのレベルを指定します。

pDB2ReadLogStruct
入力。 *db2ReadLogStruct* を指すポインター。

pSqlca
出力。*sqlca* 構造へのポインター。

iCallerAction
入力。実行するアクションを指定します。

DB2READLOG_READ
開始ログ・シーケンス番号から終了ログ・シーケンス番号までのデータベース・ログを読み取り、この範囲内にあるログ・レコードを戻します。

DB2READLOG_READ_SINGLE
開始ログ・シーケンス番号によって識別される単一のログ・レコード (伝搬可能または伝搬不可のいずれでも) を読み取ります。

DB2READLOG_QUERY
データベース・ログを照会します。照会した結果は、*db2ReadLogInfoStruct* 構造を介して戻されます。

piStartLsn
入力。開始ログ・シーケンス番号は、ログの読み取りを開始する相対バイト・アドレスを指定します。この値は、実際のログ・レコードの始まりでなければなりません。

piEndLsn
入力。終了ログ・シーケンス番号は、ログの読み取りを終了する相対バイト・アドレスを指定します。この値は、*startLsn* の値より大きくなければなりません。実際のログ・レコードの終わりである必要はありません。

poLogBuffer
出力。指定した範囲内で読み取られた、伝搬可能なすべてのログ・レコードが順番に格納されるバッファー。このバッファーは、単一のログ・レコードを保持するのに十分な大きさでなければなりません。目安として、このバッ

ファーは最低限 32 バイトでなければなりません。最大サイズは、要求された範囲のサイズによって異なってきます。バッファー内の各ログ・レコードには、接頭部として 6 バイトのログ・シーケンス番号 (LSN) が付けられます。これは、次のログ・レコードの LSN を示します。

iLogBufferSize
入力。バイト単位でログ・バッファーのサイズを指定します。

iFilterOption
入力。ログ・レコードを読み取るときに使用されるログ・レコード・フィルターのレベルを指定します。有効な値は以下のとおりです。

DB2READLOG_FILTER_OFF
指定された LSN 範囲内ですべてのログ・レコードを読み取ります。

DB2READLOG_FILTER_ON
伝搬可能とマークされた LSN 範囲内でログ・レコードのみを読み取ります。これは非同期ログ読み取り API の基本的な動作です。

poReadLogInfo
出力。呼び出しとデータベース・ログに関する情報を詳述する構造。

使用上の注意:

要求されるアクションがログの読み取りであれば、呼び出し側はログ・シーケンス番号の範囲と、ログ・レコードを保持するバッファーを提供します。この API は要求された LSN の範囲にあるログを順番に読み取ります。さらに、DATA CAPTURE オプションが CHANGES である表に関連付けられたログ・レコードと、現在アクティブなログ情報が入った db2ReadLogInfoStruct 構造を戻します。要求されたアクションが照会であれば、API は現在アクティブなログ情報が入った db2ReadLogInfoStruct 構造を戻します。

非同期ログ読み取りプログラムを使用するには、まずデータベース・ログを照会して有効な開始 LSN を探します。照会の呼び出しに続き、ログ情報の読み取り構造 (db2ReadLogInfoStruct) に、読み取りの呼び出しで使用される有効な開始 LSN (initialLSN メンバー内) が入ります。読み取りでの終了 LSN として使用される値は、次のうちの 1 つになります。

- initialLSN より大きい値
- 非同期ログ読み取りプログラムで現行ログの終わりとして解釈される、FFFF FFFF FFFF

開始および終了 LSN の範囲内で読み取られた伝搬可能なログ・レコードは、ログ・バッファーに戻されます。ログ・レコードには、その LSN は含まれません。これは、バッファーの中で、実際のログ・レコードの前に付けられます。db2ReadLog によって戻されるさまざまな DB2 ログ・レコードの詳細については、『DB2 UDB ログ・レコード』セクションで説明しています。

最初の読み取りの後、次の順番のログ・レコードを読み取るには、db2ReadLogStruct 構造で戻された nextStartLSN フィールドを使用します。この新しい開始 LSN と有効な終了 LSN を使用して、呼び出しをもう一度サブミットします。そうすると、次のブロックのレコードが読み取られます。

SQLU_RLOG_READ_TO_CURRENT の *sqlca* コードは、現在アクティブであるログが最後まで読み取られたことを示します。

関連資料:

- 「*管理 API リファレンス*」の『SQLCA』
- 「*管理 API リファレンス*」の『db2Reorg - 再編成』

関連サンプル:

- 『dbrecov.sqc -- How to recover a database (C)』
- 『dbrecov.sqC -- How to recover a database (C++)』

db2HistData

この構造は、db2HistoryGetEntry への呼び出し後に情報を戻すために使用されます。

表 2. db2HistData 構造のフィールド

フィールド名	データ・タイプ	説明
ioHistDataID	char(8)	記憶域ダンプ用の 8 バイトの構造 ID および「目印」。有効な値は "SQLUHINF" だけです。このストリングには、記号の定義はありません。
oObjectPart	db2Char	最初の 14 文字は、*yyyymmddhhnnss* の形式のタイム・スタンプで、操作を開始した時を示します。次の 3 文字はシーケンス番号です。バックアップ操作では、バックアップ・イメージが複数のファイルまたは複数のテープに保管されるときに、このファイルに複数の項目が入る可能性があります。シーケンス番号によって複数の位置を指定することができます。リストアおよびロード操作では、このファイルに 1 つの項目 (対応するバックアップのシーケンス番号 '001' に該当) だけが入ります。シーケンス番号と結合されるタイム・スタンプは、ユニークなものであることが必要です。
oEndTime	db2Char	操作が完了した時を示す *yyyymmddhhnnss* の形式のタイム・スタンプ。
oFirstLog	db2Char	最も古いログ・ファイル ID (範囲は S0000000 から S9999999)。 • オンライン・バックアップのロールフォワード・リカバリーを適用するために必要 • オフライン・バックアップのロールフォワード・リカバリーを適用するために必要 • ロードの開始時に接続していたデータベースのフル・バックアップまたは表スペース・バックアップのリストア後に適用

表 2. *db2HistData* 構造のフィールド (続き)

フィールド名	データ・タイプ	説明
oLastLog	db2Char	最新のログ・ファイル ID (範囲は S0000000 から S9999999)。 • オンライン・バックアップのロールフォワード・リカバリーを適用するために必要 • オフライン・バックアップの現時点へのロールフォワード・リカバリーを適用するために必要 • ロード操作の終了時に接続していたデータベースのフル・バックアップまたは表スペース・バックアップのリストア後に適用 (ロールフォワード・リカバリーが適用されない場合は、*oFirstLog* と同じ)
oID	db2Char	ユニークなバックアップまたは表 ID。
oTableQualifier	db2Char	表修飾子。
oTableName	db2Char	表名。
oLocation	db2Char	バックアップおよびロード・コピーの場合、このフィールドはデータが保管された場所を示します。ファイルに複数の項目が入る操作の場合、*oObjectPart* によって定義されるシーケンス番号は、指定された位置でバックアップのどの部分が検出されるかを識別します。リストアおよびロード操作の場合、ロケーションは、常に、リストアまたはロードされたデータの最初の部分 (複数パーツ・バックアップの順序 '001' に該当) が保管された場所を識別します。 *oLocation* のデータは、*oDeviceType* によって解釈方法が異なります。 • ディスクまたはディスケット (D または K) の場合、完全修飾ファイル名 • テープ (T) の場合、ボリューム・レベル • TSM (A) の場合、サーバー名 • ユーザー出口またはその他 (U または 0) の場合、自由形式のテキスト
oComment	db2Char	自由形式のテキスト注釈。
oCommandText	db2Char	コマンド・テキストまたは DDL。
oLastLSN	SQLU_LSN	最新のログ・シーケンス番号。
oEID	構造体	ユニークな項目 ID。
poEventSQLCA	構造体	記録されたイベントの結果 *sqlca*。
poTablespace	db2Char	表スペース名のリスト。
ioNumTablespaces	db2Uint32	*poTablespace* リストの項目数。各表スペース・バックアップには 1 つ以上の表スペースが含まれます。各表スペース・リストア操作は 1 つ以上の表スペースを置換します。このフィールドがゼロでない (表スペース・レベル・バックアップまたはリストアを示している) 場合、このファイルの次の行には、18 文字のストリングで表される、バックアップまたはリストアされた表スペースの名前が含まれます。各行に 1 つの表スペース名が入ります。
oOperation	char	335 ページの表 3 を参照してください。

表 2. db2HistData 構造のフィールド (続き)

フィールド名	データ・タイプ	説明
oObject	char	操作の細分性。D (全データベース)、P (表スペース)、および T (表)。
oOptype	char	336 ページの表 4 を参照してください。
oStatus	char	項目の状況。 A (処理)、 D (削除 (将来の利用))、E (期限切れ)、 I (非アクティブ)、 N (未コミット)、 Y (コミット済みまたはアクティブ)、 a (アクティブ・バックアップ。ただしバックアップをまだ完了していないデータ・リンク・サーバーもある)、および i (非アクティブ・バックアップ。ただしバックアップをまだ完了していないデータ・リンク・サーバーもある)。
oDeviceType	char	装置タイプ。このフィールドは *oLocation* フィールドの解釈方法を判別します。 A (TSM)、 C (クライアント)、 D (ディスク)、 K (ディスケット)、 L (ローカル)、 O (その他 (他のベンダーの装置をサポート))、 P (パイプ)、 Q (カーソル)、S (サーバー)、 T (テープ)、および U (ユーザー出口)。

表 3. db2HistData 構造の有効な oOperation 値

値	説明	C 定義	COBOL/FORTRAN 定義
A	表スペースの追加	DB2HISTORY_OP_ADD_ TABLESPACE	DB2HIST_OP_ADD_ TABLESPACE
B	バックアップ	DB2HISTORY_OP_BACKUP	DB2HIST_OP_BACKUP
C	ロード・コピー	DB2HISTORY_OP_LOAD_COPY	DB2HIST_OP_LOAD_COPY
D	ドロップされた表	DB2HISTORY_OP_DROPPED_ TABLE	DB2HIST_OP_DROPPED_TABLE
F	ロールフォワード	DB2HISTORY_OP_ROLLFWD	DB2HIST_OP_ROLLFWD
G	表の再編成	DB2HISTORY_OP_REORG	DB2HIST_OP_REORG
L	ロード	DB2HISTORY_OP_LOAD	DB2HIST_OP_LOAD
N	表スペースの名前変更	DB2HISTORY_OP_REN_ TABLESPACE	DB2HIST_OP_REN_ TABLESPACE
O	表スペースのドロップ	DB2HISTORY_OP_DROP_ TABLESPACE	DB2HIST_OP_DROP_ TABLESPACE
Q	静止	DB2HISTORY_OP_QUIESCE	DB2HIST_OP_QUIESCE
R	リストア	DB2HISTORY_OP_RESTORE	DB2HIST_OP_RESTORE
T	表スペースの変更	DB2HISTORY_OP_ALT_ TABLESPACE	DB2HIST_OP_ALT_TBS
U	アンロード	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD

表 4. db2HistData 構造の有効な oOptype 値

oOperation	oOptype	説明	C/COBOL/FORTRAN 定義
B	F	オフライン	DB2HISTORY_OPTYPE_OFFLINE
	N	オンライン	DB2HISTORY_OPTYPE_ONLINE
	I	増分オフライン	DB2HISTORY_OPTYPE_INCR_ OFFLINE
	O	増分オンライン	DB2HISTORY_OPTYPE_INCR_ ONLINE
	D	差分オフライン	DB2HISTORY_OPTYPE_DELTA_ OFFLINE
	E	差分オンライン	DB2HISTORY_OPTYPE_DELTA_ ONLINE
F	E	ログの終わり	DB2HISTORY_OPTYPE_EOL
	P	時点	DB2HISTORY_OPTYPE_PIT
G	F	オフライン	DB2HISTORY_OPTYPE_OFFLINE
	N	オンライン	DB2HISTORY_OPTYPE_ONLINE
L	I	挿入	DB2HISTORY_OPTYPE_INSERT
	R	置換	DB2HISTORY_OPTYPE_REPLACE
Q	S	共有の静止	DB2HISTORY_OPTYPE_SHARE
	U	更新の静止	DB2HISTORY_OPTYPE_UPDATE
	X	排他の静止	DB2HISTORY_OPTYPE_EXCL
	Z	リセットの静止	DB2HISTORY_OPTYPE_RESET
R	F	オフライン	DB2HISTORY_OPTYPE_OFFLINE
	N	オンライン	DB2HISTORY_OPTYPE_ONLINE
	I	増分オフライン	DB2HISTORY_OPTYPE_INCR_ OFFLINE
	O	増分オンライン	DB2HISTORY_OPTYPE_INCR_ ONLINE
T	C	コンテナーの追加	DB2HISTORY_OPTYPE_ADD_CONT
	R	再バランス	DB2HISTORY_OPTYPE_REB

表 5. db2Char 構造のフィールド

フィールド名	データ・タイプ	説明
pioData	char	文字データ・バッファーを指すポインター。 NULL の場合、データは戻されません。
iLength	db2Uint32	入力。 *pioData* バッファーのサイズ。
oLength	db2Uint32	出力。 *pioData* バッファー内にあるデータの有効文字の数。

表 6. db2HistoryEID 構造のフィールド

フィールド名	データ・タイプ	説明
ioNode	SQL_PDB_NODE_TYPE	ノード番号。
ioHID	db2Uint32	ローカル履歴ファイルの項目 ID。

言語構文:

C 構造

```
/* 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;
/* ... */
```

関連資料:

- 311 ページの『db2HistoryGetEntry - 履歴ファイルの次項目の入手』
- 「*管理 API リファレンス*」の『SQLCA』

SQLU-LSN

db2ReadLog API によって使用されるこの共用体には、ログ・シーケンス番号の定義が含まれます。ログ・シーケンス番号 (LSN) は、データベース・ログ内の相対バイト・アドレスを示します。すべてのログ・レコードは、この番号によって識別されます。この番号は、ログ・レコードのデータベース・ログの始まりからのバイト・オフセットを示します。

表 7. SQLU-LSN 共用体のフィールド

フィールド名	データ・タイプ	説明
lsnChar	UNSIGNED CHAR の配列	6 メンバー文字配列のログ・シーケンス番号を指定します。
lsnWord	UNSIGNED SHORT の配列	3 メンバー文字配列のログ・シーケンス番号を指定します。

言語構文:

C 構造

```
typedef union SQLU_LSN
{
unsigned char  lsnChar  [6] ;
unsigned short lsnWord  [3] ;
} SQLU_LSN;
```

関連資料:

• 330 ページの『db2ReadLog - ログの非同期読み取り』

付録 E. リカバリー・サンプル・プログラム

組み込み SQL を使用したサンプル・プログラム

下記のサンプル・プログラムは、DB2 バックアップおよびリカバリー API の使用によって以下のことを実行する方法を示しています。

- データベース・リカバリー・ファイル項目の読み取りと更新
- データベース接続を使用したデータベース・ログ・ファイルの読み取り
- データベース接続を使用しないデータベース・ログ・ファイルの読み取り
- バックアップ・イメージからのデータベースのリストア
- データベース・リストア操作後のロールフォワード操作の実行

注: これらのサンプル・ファイルは、sqllib/samples/c および sqllib/samples/cpp ディレクトリーにあります。

dbredirect サンプル・プログラム:

dbredirect サンプル・ファイルには、リダイレクトしたデータベースのリストアを実行する方法が示されています。

```
/****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
****************************************************************************
**
** SOURCE FILE NAME: dbredirect.sqc
**
** SAMPLE: How to perform Redirected Restore of a database
**
**         This program ends in ".sqc" even though it does not contain
**         embedded SQL statements. It links in  the embedded SQL utility
**         file for database connection and disconnection, so it needs the
**         embedded SQL extension for the precompiler.
**
**         Note:
**           You must be disconnected from the sample database to run
**           this program. To ensure you are, enter 'db2 connect reset'
**           on the command line prior to running dbredirect. If the target
**           database for the redirected restore already exists, SQLCODE 2529
**           will be displayed.
**
** DB2 API USED:
**         db2CfgSet -- Set Configuration
**         db2Restore -- Restore Database
**
** OUTPUT FILE: dbredirect.out (available in the online documentation)
****************************************************************************
**
```

```
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
******************************************************************************/
#include "utilrecov.c"

/* local function prototypes */
int DbBackupAndRedirectedRestore(char *, char *, char *, char *, char *);

/* support function called by DbBackupAndRedirectedRestore() */
int InaccessableContainersRedefine(char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  char redirectedRestoredDbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("¥nTHIS SAMPLE SHOWS HOW TO PERFORM A REDIRECTED RESTORE¥n");
  printf("FROM A DATABASE BACKUP.¥n");

  strcpy(redirectedRestoredDbAlias, "RRDB");

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  printf("¥nNOTE: Backup images will be created on the server¥n");
  printf("      in the directory %s,¥n", serverWorkingPath);
  printf("      and will not be deleted by the program.¥n");

  /* call the sample function */
  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);
  CHECKRC(rc, "DbRecoveryHistoryFilePrune");

  rc = DbBackupAndRedirectedRestore(dbAlias,
                                    redirectedRestoredDbAlias,
                                            user, pswd, serverWorkingPath);
  CHECKRC(rc, "DbBackupAndRedirectedRestore");

  /* Detach from the local or remote instance */
```

```
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");

  return 0;
} /* end main */

int DbBackupAndRedirectedRestore(char dbAlias[],
                                 char restoredDbAlias[],
                                      char user[],
                          char pswd[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  unsigned short logretain = 0;

  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1] = { 0 };

  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32 backupImageSize = 0;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;

  printf("¥n*************************¥n");
  printf("*** REDIRECTED RESTORE ***¥n");
  printf("*************************¥n");
  printf("¥nUSE THE DB2 APIs:¥n");
  printf("  db2CfgSet -- Upate Configuration¥n");
  printf("  db2Backup -- Backup Database¥n");
  printf("  sqlecrea -- Create Database¥n");
  printf("  db2Restore -- Restore Database¥n");
  printf("  sqlbmtsq -- Tablespace Query¥n");
  printf("  sqlbtcq -- Tablespace Container Query¥n");
  printf("  sqlbstsc -- Set Tablespace Containers¥n");
  printf("  sqlefmem -- Free Memory¥n");
  printf("  sqledrpd -- Drop Database¥n");
  printf("TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.¥n");

  printf("¥n  Update ¥'%s¥' database configuration:¥n", dbAlias);
  printf("    - Disable the database configuration parameter LOGRETAIN ¥n");
  printf("        i.e., set LOGRETAIN = OFF/NO¥n");

  /* initialize cfgParameters */
  /* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
     parameter 'logretain'; it is used to update the database configuration
     file */
  cfgParameters[0].flags = 0;
  cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
  cfgParameters[0].ptrvalue = (char *)&logretain;

  /* disable the database configuration parameter 'logretain' */
  logretain = SQLF_LOGRETAIN_DISABLE;

  /* initialize cfgStruct */
  cfgStruct.numItems = 1;
  cfgStruct.paramArray = cfgParameters;
  cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
  cfgStruct.dbname = dbAlias;

  /* get database configuration */
  db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
  DB2_API_CHECK("Db Log Retain -- Disable");
```

```
/*******************************/
/*    BACK UP THE DATABASE    */
/*******************************/

/* Calling up the routine for database backup */
rc = DbBackup(dbAlias, user, pswd, serverWorkingPath, &backupStruct);
CHECKRC(rc, "DbBackup");

/*******************************/
/*    RESTORE THE DATABASE    */
/*******************************/

strcpy(restoreTimestamp, backupStruct.oTimestamp);

rtablespaceStruct.tablespaces = NULL;
rtablespaceStruct.numTablespaces = 0;

rmediaListStruct.locations = &serverWorkingPath;
rmediaListStruct.numLocations = 1;
rmediaListStruct.locationType = SQLU_LOCAL_MEDIA;

restoreStruct.piSourceDBAlias = dbAlias;
restoreStruct.piTargetDBAlias = restoredDbAlias;
restoreStruct.piTimestamp = restoreTimestamp;
restoreStruct.piTargetDBPath = NULL;
restoreStruct.piReportFile = NULL;
restoreStruct.piTablespaceList = &rtablespaceStruct;
restoreStruct.piMediaList = &rmediaListStruct;
restoreStruct.piUsername = user;
restoreStruct.piPassword = pswd;
restoreStruct.piNewLogPath = NULL;
restoreStruct.piVendorOptions = NULL;
restoreStruct.iVendorOptionsSize = 0;
restoreStruct.iParallelism = 1;
restoreStruct.iBufferSize = 1024;     /*  1024 x 4KB */
restoreStruct.iNumBuffers = 2;
restoreStruct.iOptions = DB2RESTORE_OFFLINE | DB2RESTORE_DB |
DB2RESTORE_NODATALINK | DB2RESTORE_NOROLLFWD;

printf("¥n  Restoring a database ...¥n");
printf("    - source image alias     : %s¥n", dbAlias);
printf("    - source image time stamp: %s¥n", restoreTimestamp);
printf("    - target database        : %s¥n", restoredDbAlias);

restoreStruct.iCallerAction = DB2RESTORE_RESTORE_STORDEF;

/* The API db2Restore is used to restore a database that has been backed
   up using the API db2Backup. */
db2Restore(db2Version810, &restoreStruct, &sqlca);

/* If restoring to a different database and restoreDbAlias already exists,
   SQLCODE 2529 is expected. */
if (strcmp(dbAlias, restoredDbAlias))
{
  printf("¥n  SQLCODE 2529 is expected if target database '%s' already exists¥n",
         restoredDbAlias);
}

EXPECTED_WARN_CHECK("database restore -- start");

while (sqlca.sqlcode != 0)
{
  /* continue the restore operation */
  printf("¥n  Continuing the restore operation...¥n");

  /* depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape */
```

```
    if (sqlca.sqlcode == SQLUD_INACCESSABLE_CONTAINER)
    {
      /* redefine the table space container layout */
      printf("¥n  Find and redefine inaccessable containers.¥n");
      rc = InaccessableContainersRedefine(serverWorkingPath);
      CHECKRC(rc, "InaccessableContainersRedefine");
    }

    restoreStruct.iCallerAction = DB2RESTORE_CONTINUE;

    /* restore the database */
  db2Restore(db2Version810, &restoreStruct, &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }

  printf("¥n  Restore finished.¥n");

  /* drop the restored database */
  rc = DbDrop(restoredDbAlias);
  CHECKRC(rc, "DbDrop");

  return 0;
} /* DbBackupAndRedirectedRestore */

int InaccessableContainersRedefine(char serverWorkingPath[])
{
  struct sqlca sqlca;
  sqluint32 numTablespaces = 0;
  struct SQLB_TBSPQRY_DATA **ppTablespaces = NULL;
  sqluint32 numContainers = 0;
  struct SQLB_TBSCONTQRY_DATA *pContainers = NULL;
  int tspNb = 0;
  int contNb = 0;
  char pathSep[2] = { 0 };

  /* The API sqlbmtsq provides a one-call interface to the table space query
     data. The query data for all table spaces in the database is returned
     in an array. */
  sqlbmtsq(&sqlca,
           &numTablespaces, &ppTablespaces, SQLB_RESERVED1, SQLB_RESERVED2);
  DB2_API_CHECK("tablespaces -- get");

  /* refedine the inaccessable containers */
  for (tspNb = 0; tspNb < numTablespaces; tspNb++)
  {
    /* The API sqlbtcq provides a one-call interface to the table space
       container query data. The query data for all the containers in a table
       space, or for all containers in all table spaces, is returned in an
       array. */
    sqlbtcq(&sqlca, ppTablespaces[tspNb]->id, &numContainers, &pContainers);
    DB2_API_CHECK("tablespace containers -- get");

    for (contNb = 0; contNb < numContainers; contNb++)
    {
      if (!pContainers[contNb].ok)
      {
        /* redefine inaccessable container */
        printf("¥n    Redefine inaccessable container:¥n");
        printf("      - table space name: %s¥n", ppTablespaces[tspNb]->name);
        printf("      - default container name: %s¥n",
                   pContainers[contNb].name);
        if (strstr(pContainers[contNb].name, "/"))
        { /* UNIX */
          strcpy(pathSep, "/");
        }
      else
```

```
                { /* Intel */
                  strcpy(pathSep, "¥¥");
                }
                switch (pContainers[contNb].contType)
                {
                  case SQLB_CONT_PATH:
                    printf("      - container type: path¥n");

                    sprintf(pContainers[contNb].name, "%s%sSQLT%04d.%d",
                                                      serverWorkingPath, pathSep,
                            ppTablespaces[tspNb]->id, pContainers[contNb].id);
                    printf("      - new container name: %s¥n",
                           pContainers[contNb].name);
                  break;
                  case SQLB_CONT_DISK:
                  case SQLB_CONT_FILE:
              default:
                    printf("    Unknown container type.¥n");
                  break;
                }
              }
              }

              /* The API sqlbstsc is used to set or redefine table space containers
                 while performing a 'redirected' restore of the database. */
              sqlbstsc(&sqlca,
                       SQLB_SET_CONT_FINAL_STATE,
                       ppTablespaces[tspNb]->id, numContainers, pContainers);
              DB2_API_CHECK("tablespace containers -- redefine");

              /* The API sqlefmem is used here to free memory allocated by DB2 for use
                 with the API sqlbtcq (Tablespace Container Query). */
              sqlefmem(&sqlca, pContainers);
              DB2_API_CHECK("tablespace containers memory -- free");
            }

            /* The API sqlefmem is used here to free memory allocated by DB2 for
               use with the API sqlbmtsq (Tablespace Query). */
            sqlefmem(&sqlca, ppTablespaces);
            DB2_API_CHECK("tablespaces memory -- free");

            return 0;
          } /* InaccessableContainersRedefine */
```

dbhistfile サンプル・プログラム:

dbhistfile サンプル・ファイルには、データベース・リカバリー・ファイル項目の読み取りおよび更新の方法が示されています。

```
/*****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****************************************************************************
**
** SOURCE FILE NAME: dbhistfile.sqc
**
** SAMPLE: How to read and update a database recovery history file entry.
**
```

```
** This program ends in ".sqc" even though it does not contain
** embedded SQL statements. It links in  the embedded SQL utility
** file for database connection and disconnection, so it needs the
** embedded SQL extension for the precompiler.
**
** DB2 APIs USED:
** db2HistoryCloseScan -- Close Recovery History File Scan
** db2HistoryGetEntry -- Get Next Recovery History File Entry
** db2HistoryOpenScan -- Open Recovery History File Scan
** db2HistoryUpdate -- Update Recovery History File
**
** OUTPUT FILE: dbhistfile.out (available in the online documentation)
******************************************************************************
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
******************************************************************************/
#include "utilrecov.c"

/* local function prototypes */
int DbRecoveryHistoryFileRead(char *);
int DbFirstRecoveryHistoryFileEntryUpdate(char *, char *, char *);
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *);
int HistoryEntryDisplay(struct db2HistoryData );
int HistoryEntryDataFieldsFree(struct db2HistoryData *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  sqluint16 savedLogRetainValue = 0;
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("¥nTHIS SAMPLE SHOWS HOW TO READ A DATABASE RECOVERY HISTORY FILE ¥n");
  printf("AND UPDATE A RECOVERY HISTORY FILE ENTRY. ¥n");

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  rc = DbRecoveryHistoryFileRead(dbAlias);
  CHECKRC(rc, "DbRecoveryHistoryFileRead");

  rc = DbFirstRecoveryHistoryFileEntryUpdate(dbAlias, user, pswd);
  CHECKRC(rc, "DbFirstRecoveryHistoryFileEntryUpdate");
```

```
  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");

  return 0;
} /* end main */

int DbRecoveryHistoryFileRead(char dbAlias[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint32 numEntries = 0;
  sqluint16 recoveryHistoryFileHandle = 0;
  sqluint32 entryNb = 0;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;

  printf("¥n*********************************************¥n");
  printf("*** READ A DATABASE RECOVERY HISTORY FILE ***¥n");
  printf("*********************************************¥n");
  printf("¥nUSE THE DB2 APIs:¥n");
  printf("  db2HistoryOpenScan -- Open Recovery History File Scan¥n");
  printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry¥n");
  printf("  db2HistoryCloseScan -- Close Recovery History File Scan¥n");
  printf("TO READ A DATABASE RECOVERY HISTORY FILE.¥n");

  /* initialize the data structures */
  dbHistoryOpenParam.piDatabaseAlias = dbAlias;
  dbHistoryOpenParam.piTimestamp = NULL;
  dbHistoryOpenParam.piObjectName = NULL;
  dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;
  dbHistoryEntryGetParam.pioHistData = &histEntryData;
  dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;

  rc = HistoryEntryDataFieldsAlloc(&histEntryData);
  CHECKRC(rc, "HistoryEntryDataFieldsAlloc");

  /*****************************************/
  /* OPEN THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("¥n  Open recovery history file for '%s' database.¥n", dbAlias);

  /* open the recovery history file to scan */
  db2HistoryOpenScan(db2Version810, &dbHistoryOpenParam, &sqlca);
  DB2_API_CHECK("database recovery history file -- open");
  numEntries = dbHistoryOpenParam.oNumRows;

  /* dbHistoryOpenParam.oHandle returns the handle for scan access */
  recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
  dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

  /*********************************************/
  /* READ AN ENTRY IN THE RECOVERY HISTORY FILE */
  /*********************************************/
  for (entryNb = 0; entryNb < numEntries; entryNb++)
  {
    printf("¥n  Read entry number %u.¥n", entryNb);

    /* get the next entry from the recovery history file */
  db2HistoryGetEntry(db2Version810, &dbHistoryEntryGetParam, &sqlca);
    DB2_API_CHECK("database recovery history file entry -- read")

    /* display the entries in the recovery history file */
    printf("¥n  Display entry number %u.¥n", entryNb);
  rc = HistoryEntryDisplay(histEntryData);
  CHECKRC(rc, "HistoryEntryDisplay");
```

```
  }

  /*****************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("¥n  Close recovery history file for '%s' database.¥n", dbAlias);

  /* The API db2HistoryCloseScan ends the recovery history file scan and
     frees DB2 resources required for the scan. */
  db2HistoryCloseScan(db2Version810, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);
  CHECKRC(rc, "HistoryEntryDataFieldsFree");

  return 0;
} /* DbRecoveryHistoryFileRead */

int DbFirstRecoveryHistoryFileEntryUpdate(char dbAlias[], char user[],
                                          char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint16 recoveryHistoryFileHandle = 0;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;
  char newLocation[DB2HISTORY_LOCATION_SZ + 1] = { 0 };
  char newComment[DB2HISTORY_COMMENT_SZ + 1] = { 0 };
  struct db2HistoryUpdateStruct dbHistoryUpdateParam;

  printf("¥n*****************************************************¥n");
  printf("*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***¥n");
  printf("*****************************************************¥n");
  printf("¥nUSE THE DB2 APIs:¥n");
  printf("  db2HistoryOpenScan -- Open Recovery History File Scan¥n");
  printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry¥n");
  printf("  db2HistoryUpdate -- Update Recovery History File¥n");
  printf("  db2HistoryCloseScan -- Close Recovery History File Scan¥n");
  printf("TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.¥n");

  /* initialize data structures */
  dbHistoryOpenParam.piDatabaseAlias = dbAlias;
  dbHistoryOpenParam.piTimestamp = NULL;
  dbHistoryOpenParam.piObjectName = NULL;
  dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;

  dbHistoryEntryGetParam.pioHistData = &histEntryData;
  dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
  rc = HistoryEntryDataFieldsAlloc(&histEntryData);
  CHECKRC(rc, "HistoryEntryDataFieldsAlloc");

  /*****************************************/
  /* OPEN THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("¥n  Open the recovery history file for '%s' database.¥n",
         dbAlias);

  /* The API db2HistoryOpenScan starts a recovery history file scan */
  db2HistoryOpenScan(db2Version810, &dbHistoryOpenParam, &sqlca);
  DB2_API_CHECK("database recovery history file -- open");

  /* dbHistoryOpenParam.oHandle returns the handle for scan access */
  recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
  dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
```

```
| /*************************************************/
| /* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE */
| /*************************************************/
| printf("¥n  Read the first entry in the recovery history file.¥n");
|
| /* The API db2HistoryGetEntry gets the next entry from the recovery
|    history file. */
| db2HistoryGetEntry(db2Version810, &dbHistoryEntryGetParam, &sqlca);
| DB2_API_CHECK("first recovery history file entry -- read");
| printf("¥n  Display the first entry.¥n");
|
| /* HistoryEntryDisplay is a support function used to display the entries
|    in the recovery history file. */
| rc = HistoryEntryDisplay(histEntryData);
| CHECKRC(rc, "HistoryEntryDisplay");
|
| /* update the first history file entry */
| rc = DbConn(dbAlias, user, pswd);
| CHECKRC(rc, "DbConn");
|
| strcpy(newLocation, "this is the NEW LOCATION");
| strcpy(newComment, "this is the NEW COMMENT");
| printf("¥n  Update the first entry in the history file:¥n");
| printf("    new location = '%s'¥n", newLocation);
| printf("    new comment = '%s'¥n", newComment);
|
| dbHistoryUpdateParam.piNewLocation = newLocation;
| dbHistoryUpdateParam.piNewDeviceType = NULL;
| dbHistoryUpdateParam.piNewComment = newComment;
| dbHistoryUpdateParam.iEID.ioNode = histEntryData.oEID.ioNode;
| dbHistoryUpdateParam.iEID.ioHID = histEntryData.oEID.ioHID;
|
| /* The API db2HistoryUpdate can be used to update the location,
|    device type, or comment in a history file entry. */
|
| /* Call this API to update the location and comment of the first
|    entry in the history file: */
| db2HistoryUpdate(db2Version810, &dbHistoryUpdateParam, &sqlca);
| DB2_API_CHECK("first history file entry -- update");
|
| rc = DbDisconn(dbAlias);
| CHECKRC(rc, "DbDisconn");
|
| /*********************************************/
| /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
| /*********************************************/
| printf("¥n  Close recovery history file for '%s' database.¥n", dbAlias);
|
| /* The API db2HistoryCloseScan ends the recovery history file scan and
|    frees DB2 resources required for the scan. */
| db2HistoryCloseScan(db2Version810, &recoveryHistoryFileHandle, &sqlca);
| DB2_API_CHECK("database recovery history file -- close");
|
| /*********************************************/
| /* RE-OPEN THE DATABASE RECOVERY HISTORY FILE */
| /*********************************************/
| printf("¥n  Open the recovery history file for '%s' database.¥n",
|        dbAlias);
|
| /* starts a recovery history file scan */
| db2HistoryOpenScan(db2Version810, &dbHistoryOpenParam, &sqlca);
| DB2_API_CHECK("database recovery history file -- open");
|
| recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
| dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
| printf("¥n  Read the first recovery history file entry.¥n");
|
```

```
  /*****************************************************************************/
  /* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE AFTER MODIFICATION */
  /*****************************************************************************/
  db2HistoryGetEntry(db2Version810, &dbHistoryEntryGetParam, &sqlca);
  DB2_API_CHECK("first recovery history file entry -- read");

  printf("¥n  Display the first entry.¥n");
  rc = HistoryEntryDisplay(histEntryData);
  CHECKRC(rc, "HistoryEntryDisplay");

  /*****************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("¥n  Close the recovery history file for '%s' database.¥n",
         dbAlias);

  /* ends the recovery history file scan */
  db2HistoryCloseScan(db2Version810, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);
  CHECKRC(rc, "HistoryEntryDataFieldsFree");

  return 0;
} /* DbFirstRecoveryHistoryFileEntryUpdate */

/*****************************************************************************/
/* HistoryEntryDataFieldsAlloc         */
/* Allocates memory for all the fields in a database recovery history      */
/* file entry                                                                */
/*****************************************************************************/
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb = 0;

  strcpy(pHistEntryData->ioHistDataID, "SQLUHINF");

  pHistEntryData->oObjectPart.pioData = malloc(DB2HISTORY_OBJPART_SZ + 1);
  pHistEntryData->oObjectPart.iLength = DB2HISTORY_OBJPART_SZ + 1;

  pHistEntryData->oEndTime.pioData = malloc(DB2HISTORY_TIMESTAMP_SZ + 1);
  pHistEntryData->oEndTime.iLength = DB2HISTORY_TIMESTAMP_SZ + 1;

  pHistEntryData->oFirstLog.pioData = malloc(DB2HISTORY_LOGFILE_SZ + 1);
  pHistEntryData->oFirstLog.iLength = DB2HISTORY_LOGFILE_SZ + 1;

  pHistEntryData->oLastLog.pioData = malloc(DB2HISTORY_LOGFILE_SZ + 1);
  pHistEntryData->oLastLog.iLength = DB2HISTORY_LOGFILE_SZ + 1;

  pHistEntryData->oID.pioData = malloc(DB2HISTORY_ID_SZ + 1);
  pHistEntryData->oID.iLength = DB2HISTORY_ID_SZ + 1;

  pHistEntryData->oTableQualifier.pioData =
    malloc(DB2HISTORY_TABLE_QUAL_SZ + 1);
  pHistEntryData->oTableQualifier.iLength = DB2HISTORY_TABLE_QUAL_SZ + 1;

  pHistEntryData->oTableName.pioData = malloc(DB2HISTORY_TABLE_NAME_SZ + 1);
  pHistEntryData->oTableName.iLength = DB2HISTORY_TABLE_NAME_SZ + 1;

  pHistEntryData->oLocation.pioData = malloc(DB2HISTORY_LOCATION_SZ + 1);
  pHistEntryData->oLocation.iLength = DB2HISTORY_LOCATION_SZ + 1;

  pHistEntryData->oComment.pioData = malloc(DB2HISTORY_COMMENT_SZ + 1);
  pHistEntryData->oComment.iLength = DB2HISTORY_COMMENT_SZ + 1;
```

```
                        pHistEntryData->oCommandText.pioData = malloc(DB2HISTORY_COMMAND_SZ + 1);
                        pHistEntryData->oCommandText.iLength = DB2HISTORY_COMMAND_SZ + 1;

                        pHistEntryData->poEventSQLCA =
                          (struct sqlca *)malloc(sizeof(struct sqlca));

                        pHistEntryData->poTablespace = (db2Char *) malloc(3 * sizeof(db2Char));
                        for (tsNb = 0; tsNb < 3; tsNb++)
                        {
                          pHistEntryData->poTablespace[tsNb].pioData = malloc(18 + 1);
                          pHistEntryData->poTablespace[tsNb].iLength = 18 + 1;
                        }

                        pHistEntryData->iNumTablespaces = 3;

                        return 0;
                      } /* HistoryEntryDataFieldsAlloc */

                      /******************************************************************************/
                      /* HistoryEntryDisplay                */
                      /* Displays the fields of an entry in the database recovery history file   */
                      /******************************************************************************/
                      int HistoryEntryDisplay(struct db2HistoryData histEntryData)
                      {
                        int rc = 0;
                        int bufLen = 0;
                        char *buf = NULL;
                        sqluint32 tsNb = 0;

                          bufLen =
                          MIN(histEntryData.oObjectPart.oLength,
                       histEntryData.oObjectPart.iLength);
                          buf = malloc(bufLen + 1);
                        memcpy(buf, histEntryData.oObjectPart.pioData, bufLen);
                          buf[bufLen] = '¥0';
                        printf("    object part: %s¥n", buf);
                          free(buf);

                          bufLen =
                          MIN(histEntryData.oEndTime.oLength, histEntryData.oEndTime.iLength);
                          buf = malloc(bufLen + 1);
                        memcpy(buf, histEntryData.oEndTime.pioData, bufLen);
                          buf[bufLen] = '¥0';
                        printf("    end time: %s¥n", buf);
                          free(buf);

                          bufLen =
                          MIN(histEntryData.oFirstLog.oLength, histEntryData.oFirstLog.iLength);
                          buf = malloc(bufLen + 1);
                        memcpy(buf, histEntryData.oFirstLog.pioData, bufLen);
                          buf[bufLen] = '¥0';
                        printf("    first log: %s¥n", buf);
                          free(buf);

                          bufLen =
                          MIN(histEntryData.oLastLog.oLength, histEntryData.oLastLog.iLength);
                          buf = malloc(bufLen + 1);
                        memcpy(buf, histEntryData.oLastLog.pioData, bufLen);
                          buf[bufLen] = '¥0';
                        printf("    last log: %s¥n", buf);
                          free(buf);

                        bufLen = MIN(histEntryData.oID.oLength, histEntryData.oID.iLength);
                          buf = malloc(bufLen + 1);
                        memcpy(buf, histEntryData.oID.pioData, bufLen);
                          buf[bufLen] = '¥0';
                        printf("    ID: %s¥n", buf);
```

```
                        free(buf);

                        bufLen =
                        MIN(histEntryData.oTableQualifier.oLength,
                      histEntryData.oTableQualifier.iLength);
                        buf = malloc(bufLen + 1);
                      memcpy(buf, histEntryData.oTableQualifier.pioData, bufLen);
                        buf[bufLen] = '¥0';
                      printf("    table qualifier: %s¥n", buf);
                        free(buf);

                        bufLen =
                        MIN(histEntryData.oTableName.oLength, histEntryData.oTableName.iLength);
                        buf = malloc(bufLen + 1);
                      memcpy(buf, histEntryData.oTableName.pioData, bufLen);
                        buf[bufLen] = '¥0';
                      printf("    table name: %s¥n", buf);
                        free(buf);

                        bufLen =
                        MIN(histEntryData.oLocation.oLength, histEntryData.oLocation.iLength);
                        buf = malloc(bufLen + 1);
                      memcpy(buf, histEntryData.oLocation.pioData, bufLen);
                        buf[bufLen] = '¥0';
                      printf("    location: %s¥n", buf);
                        free(buf);

                        bufLen =
                        MIN(histEntryData.oComment.oLength, histEntryData.oComment.iLength);
                        buf = malloc(bufLen + 1);
                      memcpy(buf, histEntryData.oComment.pioData, bufLen);
                        buf[bufLen] = '¥0';
                      printf("    comment: %s¥n", buf);
                        free(buf);

                        bufLen =
                        MIN(histEntryData.oCommandText.oLength,
                      histEntryData.oCommandText.iLength);

                        buf = malloc(bufLen + 1);
                      memcpy(buf, histEntryData.oCommandText.pioData, bufLen);
                        buf[bufLen] = '¥0';
                      printf("    command text: %s¥n", buf);
                      printf("    history file entry ID: %u¥n", histEntryData.oEID.ioHID);
                      printf("    table spaces:¥n");
                        free(buf);

                      for (tsNb = 0; tsNb < histEntryData.oNumTablespaces; tsNb++)
                      {
                        bufLen =
                          MIN(histEntryData.poTablespace[tsNb].oLength,
                       histEntryData.poTablespace[tsNb].iLength);
                        buf = malloc(bufLen + 1);
                        memcpy(buf, histEntryData.poTablespace[tsNb].pioData, bufLen);
                        buf[bufLen] = '¥0';
                        printf("      %s¥n", buf);
                        free(buf);
                      }

                      printf("    type of operation: %c¥n", histEntryData.oOperation);
                      printf("    granularity of the operation: %c¥n", histEntryData.oObject);
                      printf("    operation type: %c¥n", histEntryData.oOptype);
                      printf("    entry status: %c¥n", histEntryData.oStatus);
                      printf("    device type: %c¥n", histEntryData.oDeviceType);
                      printf("    SQLCA:¥n");
                      printf("      sqlcode: %ld¥n", histEntryData.poEventSQLCA->sqlcode);
```

```
  bufLen = SQLUDF_SQLSTATE_LEN;
    buf = malloc(bufLen + 1);
  memcpy(buf, histEntryData.poEventSQLCA->sqlstate, bufLen);
    buf[bufLen] = '¥0';
  printf("      sqlstate: %s¥n", buf);
    free(buf);

  bufLen = histEntryData.poEventSQLCA->sqlerrml;
    buf = malloc(bufLen + 1);
  memcpy(buf, histEntryData.poEventSQLCA->sqlerrmc, bufLen);
    buf[bufLen] = '¥0';
  printf("      message: %s¥n", buf);
    free(buf);

  return 0;
} /* HistoryEntryDisplay */

/************************************************************************/
/* HistoryEntryDataFieldsFree                */
/* Deallocates the memory for database recovery history file structures    */
/************************************************************************/
int HistoryEntryDataFieldsFree(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb = 0;

  free(pHistEntryData->oObjectPart.pioData);
  free(pHistEntryData->oEndTime.pioData);
  free(pHistEntryData->oFirstLog.pioData);
  free(pHistEntryData->oLastLog.pioData);
  free(pHistEntryData->oID.pioData);
  free(pHistEntryData->oTableQualifier.pioData);
  free(pHistEntryData->oTableName.pioData);
  free(pHistEntryData->oLocation.pioData);
  free(pHistEntryData->oComment.pioData);
  free(pHistEntryData->oCommandText.pioData);
  free(pHistEntryData->poEventSQLCA);
  pHistEntryData->oObjectPart.pioData = NULL;
  pHistEntryData->oEndTime.pioData = NULL;
  pHistEntryData->oFirstLog.pioData = NULL;
  pHistEntryData->oLastLog.pioData = NULL;
  pHistEntryData->oID.pioData = NULL;
  pHistEntryData->oTableQualifier.pioData = NULL;
  pHistEntryData->oTableName.pioData = NULL;
  pHistEntryData->oLocation.pioData = NULL;
  pHistEntryData->oComment.pioData = NULL;
  pHistEntryData->oCommandText.pioData = NULL;
  pHistEntryData->poEventSQLCA = NULL;

  for (tsNb = 0; tsNb < 3; tsNb++)
  {
    free(pHistEntryData->poTablespace[tsNb].pioData);
    pHistEntryData->poTablespace[tsNb].pioData = NULL;
  }

  free(pHistEntryData->poTablespace);
  pHistEntryData->poTablespace = NULL;

  return 0;
} /* HistoryEntryDataFieldsFree */
```

dblogconn サンプル・プログラム:

dblogconn サンプル・ファイルには、データベース接続を使用したデータベース・ログ・ファイルの読み取り方法が示されています。

```
/*****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****************************************************************************
**
** SOURCE FILE NAME: dblogconn.sqc
**
** SAMPLE: How to read database log files asynchronously with a database
**         connection
**
**         Note:
**           You must be initially disconnected from the sample database
**           to run this program. To ensure you are, enter 'db2 connect
**           reset' on the command line prior to running dblogconn.
**
** DB2 API USED:
**         db2CfgSet -- Set Configuration
**         db2ReadLog -- Asynchronous Read Log
**
** SQL STATEMENTS USED:
**         ALTER TABLE
**         COMMIT
**         DELETE
**         INSERT
**         ROLLBACK
**         CONNECT RESET
**
** OUTPUT FILE: dblogconn.out (available in the online documentation)
*****************************************************************************
**
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
*****************************************************************************/
#include "utilrecov.c"
#include "utilemb.h"

/* local function prototypes */
int DbLogRecordsForCurrentConnectionRead(char *, char *, char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
```

```
  sqluint16 savedLogRetainValue = 0;
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("¥nTHIS SAMPLE SHOWS HOW TO READ DATABASE LOGS ASYNCHRONOUSLY¥n");
  printf("WITH A DATABASE CONNECTION.¥n");

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  /* call the function to do asynchronous log read */
  rc = DbLogRecordsForCurrentConnectionRead(dbAlias,
                                            user, pswd, serverWorkingPath);
  CHECKRC(rc, "DbLogRecordsForCurrentConnectionRead");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");

  return 0;
} /* end main */

int DbLogRecordsForCurrentConnectionRead(char dbAlias[],
                                         char user[],
                                        char pswd[],
                                        char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  unsigned short logretain = 0;

  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32 backupImageSize = 0;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;

  SQLU_LSN  startLSN;
  SQLU_LSN  endLSN;
  char      *logBuffer = NULL;
  sqluint32 logBufferSize = 0;
  db2ReadLogInfoStruct readLogInfo;
  db2ReadLogStruct     readLogInput;
  int i = 0;

  printf("¥n*****************************¥n");
  printf("*** ASYNCHRONOUS READ LOG ***¥n");
  printf("*****************************¥n");
  printf("¥nUSE THE DB2 APIs:¥n");
  printf("  db2CfgSet -- Set Configuration¥n");
  printf("  db2Backup -- Backup Database¥n");
  printf("  db2ReadLog -- Asynchronous Read Log¥n");
  printf("AND THE SQL STATEMENTS:¥n");
```

```
| printf("  CONNECT¥n");
| printf("  ALTER TABLE¥n");
| printf("  COMMIT¥n");
| printf("  INSERT¥n");
| printf("  DELETE¥n");
| printf("  ROLLBACK¥n");
| printf("  CONNECT RESET¥n");
| printf("TO READ LOG RECORDS FOR THE CURRENT CONNECTION.¥n");
|
| printf("¥n  Update ¥'%s¥' database configuration:¥n", dbAlias);
| printf("    - Enable the database configuration parameter LOGRETAIN ¥n");
| printf("        i.e., set LOGRETAIN = RECOVERY/YES¥n");
|
| /* initialize cfgParameters */
| cfgParameters[0].flags = 0;
| cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
| cfgParameters[0].ptrvalue = (char *)&logretain;
|
| /* enable LOGRETAIN */
| logretain = SQLF_LOGRETAIN_RECOVERY;
|
| /* initialize cfgStruct */
| cfgStruct.numItems = 1;
| cfgStruct.paramArray = cfgParameters;
| cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
| cfgStruct.dbname = dbAlias;
|
| /* get database configuration */
| db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
| DB2_API_CHECK("Db Log Retain -- Enable");
|
| tablespaceStruct.tablespaces = NULL;
| tablespaceStruct.numTablespaces = 0;
|
| mediaListStruct.locations = &serverWorkingPath;
| mediaListStruct.numLocations = 1;
| mediaListStruct.locationType = SQLU_LOCAL_MEDIA;
|
| /* Calling up the routine for database backup */
| rc = DbBackup(dbAlias, user, pswd, serverWorkingPath, &backupStruct);
| CHECKRC(rc, "DbBackup");
|
| /* connect to the database */
| rc = DbConn(dbAlias, user, pswd);
| CHECKRC(rc, "DbConn");
|
| /* invoke SQL statements to fill database log */
| printf("¥n  Invoke the following SQL statements:¥n"
|        "    ALTER TABLE emp_resume DATA CAPTURE CHANGES;¥n"
|        "    COMMIT;¥n"
|        "    INSERT INTO emp_resume¥n"
|        "      VALUES('000777', 'ascii', 'This is a new resume.');¥n"
|        "            ('777777', 'ascii', 'This is another new resume');¥n"
|        "    COMMIT;¥n"
|        "    DELETE FROM emp_resume WHERE empno = '000777';¥n"
|        "    DELETE FROM emp_resume WHERE empno = '777777';¥n"
|        "    COMMIT;¥n"
|        "    DELETE FROM emp_resume WHERE empno = '000140';¥n"
|        "    ROLLBACK;¥n"
|        "    ALTER TABLE emp_resume DATA CAPTURE NONE;¥n" "    COMMIT;¥n");
|
| EXEC SQL ALTER TABLE emp_resume DATA CAPTURE CHANGES;
| EMB_SQL_CHECK("SQL statement 1 -- invoke");
|
| EXEC SQL COMMIT;
| EMB_SQL_CHECK("SQL statement 2 -- invoke");
|
```

```
EXEC SQL INSERT INTO emp_resume
  VALUES('000777', 'ascii', 'This is a new resume.'),
        ('777777', 'ascii', 'This is another new resume');
EMB_SQL_CHECK("SQL statement 3 -- invoke");

EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 4 -- invoke");

EXEC SQL DELETE FROM emp_resume WHERE empno = '000777';
EMB_SQL_CHECK("SQL statement 5 -- invoke");

EXEC SQL DELETE FROM emp_resume WHERE empno = '777777';
EMB_SQL_CHECK("SQL statement 6 -- invoke");

EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 7 -- invoke");

EXEC SQL DELETE FROM emp_resume WHERE empno = '000140';
EMB_SQL_CHECK("SQL statement 8 -- invoke");

EXEC SQL ROLLBACK;
EMB_SQL_CHECK("SQL statement 9 -- invoke");

EXEC SQL ALTER TABLE emp_resume DATA CAPTURE NONE;
EMB_SQL_CHECK("SQL statement 10 -- invoke");

EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 11 -- invoke");

printf("¥n  Start reading database log.¥n");

logBuffer = NULL;
logBufferSize = 0;

/*
 * The API db2ReadLog (Asynchronous Read Log) is used to extract
 * records from the database logs, and to query the log manager for
 * current log state information. This API can only be used on
 * recoverable databases.
 */

/* Query the log manager for current log state information. */
readLogInput.iCallerAction   = DB2READLOG_QUERY;
readLogInput.piStartLSN      = NULL;
readLogInput.piEndLSN        = NULL;
readLogInput.poLogBuffer     = NULL;
readLogInput.iLogBufferSize  = 0;
readLogInput.iFilterOption  = DB2READLOG_FILTER_ON;
readLogInput.poReadLogInfo   = &readLogInfo;

  db2ReadLog(db2Version810, &readLogInput, &sqlca);
DB2_API_CHECK("database log info -- get");

logBufferSize = 64 * 1024; /* Maximum size of a log buffer */
logBuffer = (char *)malloc(logBufferSize);

memcpy(&startLSN, &(readLogInfo.initialLSN), sizeof(startLSN));
memcpy(&endLSN, &(readLogInfo.nextStartLSN), sizeof(endLSN));

/*
 * Extract a log record from the database logs, and read the first
 * log sequence asynchronously.
 */
readLogInput.iCallerAction   = DB2READLOG_READ;
readLogInput.piStartLSN      = &startLSN;
readLogInput.piEndLSN        = &endLSN;
readLogInput.poLogBuffer     = logBuffer;
```

```
  readLogInput.iLogBufferSize  = logBufferSize;
  readLogInput.iFilterOption  = DB2READLOG_FILTER_ON;
  readLogInput.poReadLogInfo   = &readLogInfo;

    db2ReadLog(db2Version810, &readLogInput, &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
      DB2_API_CHECK("database logs -- read");
  }
      else
  {
    if (readLogInfo.logRecsWritten == 0)
    {
      printf("¥n  Database log empty.¥n");
    }
  }

    /* display log buffer */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
    CHECKRC(rc, "LogBufferDisplay");

  while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    /* read the next log sequence */

  memcpy(&startLSN, &(readLogInfo.nextStartLSN), sizeof(startLSN));

    /*
     * Extract a log record from the database logs, and read the
     * next log sequence asynchronously.
     */
    db2ReadLog(db2Version810, &readLogInput, &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
    {
      DB2_API_CHECK("database logs -- read");
    }
    /* display log buffer */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
    CHECKRC(rc, "LogBufferDisplay");
  }

  /* free the log buffer */
  free(logBuffer);
  logBuffer = NULL;
  logBufferSize = 0;

  /* disconnect from the database */
  rc = DbDisconn(dbAlias);
  CHECKRC(rc, "DbDisconn");

  return 0;
} /* DbLogRecordsForCurrentConnectionRead */
```

dblognoconn サンプル・プログラム:

dblognoconn サンプル・ファイルには、データベース接続を使用しないデータベース・ログ・ファイルの読み取り方法が示されています。

```
/****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
```

```
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
******************************************************************************
**
** SOURCE FILE NAME: dblognoconn.sqc
**
** SAMPLE: How to read database log files asynchronously
**         with no database connection
**
**         This program ends in ".sqc" even though it does not contain
**         embedded SQL statements. It links in  the embedded SQL utility
**         file for database connection and disconnection, so it needs the
**         embedded SQL extension for the precompiler.
**
**         Note:
**           You must be disconnected from the sample database to run
**           this program. To ensure you are, enter 'db2 connect reset'
**           on the command line prior to running dblognoconn.
**
** DB2 API USED:
**         db2CfgSet -- Set Configuration
**         db2ReadLogNoConnInit -- Read log without a db connection
**         db2ReadLog -- Asynchronous Read Log
**
** OUTPUT FILE: dblognoconn.out (available in the online documentation)
******************************************************************************
**
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
******************************************************************************/
#include "utilrecov.c"
#include "utilemb.h"

/* local function prototypes */
int DbReadLogRecordsNoConn(char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  sqluint16 savedLogRetainValue = 0;
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("¥nTHIS SAMPLE SHOWS HOW TO READ DATABASE LOGS ASYNCHRONOUSLY¥n");
```

```
  printf("WITH NO DATABASE CONNECTION.¥n");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);
  CHECKRC(rc, "DbRecoveryHistoryFilePrune");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");

  rc = DbReadLogRecordsNoConn(dbAlias);
  CHECKRC(rc, "DbReadLogRecordsNoConn");

  return 0;
} /* end main */

int DbReadLogRecordsNoConn(char dbAlias[])
{
  int rc = 0;
  struct sqlca sqlca;
  char logPath[SQL_PATH_SZ + 1] = { 0 };
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  char      nodeName[] = "NODE0000¥0";
  db2Uint32 readLogMemSize = 0;
  char      *readLogMemory = NULL;
  struct db2ReadLogNoConnInitStruct readLogInit;
  struct db2ReadLogNoConnInfoStruct readLogInfo;
  struct db2ReadLogNoConnStruct readLogInput;
  SQLU_LSN  startLSN;
  SQLU_LSN  endLSN;
  char      *logBuffer = NULL;
  db2Uint32 logBufferSize = 0;
  struct db2ReadLogNoConnTermStruct readLogTerm;

  printf("¥n*********************************¥n");
  printf("*** NO DB CONNECTION READ LOG ***¥n");
  printf("*********************************¥n");
  printf("¥nUSE THE DB2 APIs:¥n");
  printf("  db2ReadLogNoConnInit -- Initialize No Db Connection Read Log¥n");
  printf("  db2ReadLogNoConn -- No Db Connection Read Log¥n");
  printf("  db2ReadLogNoConnTerm -- Terminate No Db Connection Read Log¥n");
  printf("TO READ LOG RECORDS FROM A DATABASE LOG DIRECTORY.¥n");

  /* Determine the logpath to read log files from */
  cfgParameters[0].flags = 0;
  cfgParameters[0].token = SQLF_DBTN_LOGPATH;
  cfgParameters[0].ptrvalue =
    (char *)malloc((SQL_PATH_SZ + 1) * sizeof(char));

  /* Initialize cfgStruct */
  cfgStruct.numItems = 1;
  cfgStruct.paramArray = cfgParameters;
  cfgStruct.flags = db2CfgDatabase;
  cfgStruct.dbname = dbAlias;

  db2CfgGet(db2Version810, (void *)&cfgStruct, &sqlca);
  DB2_API_CHECK("log path -- get");

  strcpy(logPath, cfgParameters[0].ptrvalue);
  free(cfgParameters[0].ptrvalue);
  cfgParameters[0].ptrvalue = NULL;

  /*
```

```
| * First we must allocate memory for the API's control blocks and log
| * buffer
| */
| readLogMemSize = 4 * 4096;
| readLogMemory = (char *)malloc(readLogMemSize);
|
| /* Invoke the initialization API to set up the control blocks */
| readLogInit.iFilterOption       = DB2READLOG_FILTER_ON;
| readLogInit.piLogFilePath       = logPath;
| readLogInit.piOverflowLogPath   = NULL;
| readLogInit.iRetrieveLogs = DB2READLOG_RETRIEVE_OFF;
| readLogInit.piDatabaseName      = dbAlias;
| readLogInit.piNodeName          = nodeName;
| readLogInit.iReadLogMemoryLimit = readLogMemSize;
| readLogInit.poReadLogMemPtr     = &readLogMemory;
|
| db2ReadLogNoConnInit(db2Version810, &readLogInit, &sqlca);
| if (sqlca.sqlcode != SQLU_RLOG_LSNS_REUSED)
| {
|   DB2_API_CHECK("database logs no db conn -- initialization");
| }
| /* Query for the current log information */
| readLogInput.iCallerAction   = DB2READLOG_QUERY;
| readLogInput.piStartLSN      = NULL;
| readLogInput.piEndLSN        = NULL;
| readLogInput.poLogBuffer     = NULL;
| readLogInput.iLogBufferSize  = 0;
| readLogInput.piReadLogMemPtr = readLogMemory;
| readLogInput.poReadLogInfo   = &readLogInfo;
|
|   db2ReadLogNoConn(db2Version810, &readLogInput, &sqlca);
| if (sqlca.sqlcode != 0)
| {
|   DB2_API_CHECK("database logs no db conn -- query");
| }
| /* Read some log records */
| logBufferSize = 64 * 1024; /* Maximum size of a log buffer */
| logBuffer = (char *)malloc(logBufferSize);
|
| memcpy(&startLSN, &(readLogInfo.nextStartLSN), sizeof(startLSN));
| endLSN.lsnWord[0] = 0xffff;
| endLSN.lsnWord[1] = 0xffff;
| endLSN.lsnWord[2] = 0xffff;
|
| readLogInput.iCallerAction   = DB2READLOG_READ;
| readLogInput.piStartLSN      = &startLSN;
| readLogInput.piEndLSN        = &endLSN;
| readLogInput.poLogBuffer     = logBuffer;
| readLogInput.iLogBufferSize  = logBufferSize;
| readLogInput.piReadLogMemPtr = readLogMemory;
| readLogInput.poReadLogInfo   = &readLogInfo;
|
|   db2ReadLogNoConn(db2Version810, &readLogInput, &sqlca);
|   if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
| {
|     DB2_API_CHECK("database logs no db conn -- read");
| }
|     else
| {
|   if (readLogInfo.logRecsWritten == 0)
|   {
|     printf("¥n  Database log empty.¥n");
|   }
| }
|
| /* Display the log records read */
|   rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
```

```
    CHECKRC(rc, "LogBufferDisplay");

  while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    /* read the next log sequence */
  memcpy(&startLSN, &(readLogInfo.nextStartLSN), sizeof(startLSN));

    /*
     * Extract a log record from the database logs, and read the
     * next log sequence asynchronously.
     */
    db2ReadLogNoConn(db2Version810, &readLogInput, &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
    {
      DB2_API_CHECK("database logs no db conn -- read");
    }
    /* display log buffer */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
    CHECKRC(rc, "LogBufferDisplay");
  }

  printf("¥nRead to end of logs.¥n¥n");
  free(logBuffer);
  logBuffer = NULL;
  logBufferSize = 0;

  readLogTerm.poReadLogMemPtr = &readLogMemory;

  db2ReadLogNoConnTerm(db2Version810, &readLogTerm, &sqlca);
    DB2_API_CHECK("database logs no db conn -- terminate");

  return 0;
} /* DbReadLogRecordsNoConn */
```

dbrestore サンプル・プログラム:

dbrestore サンプル・ファイルには、バックアップ・イメージからデータベースをリストアする方法が示されています。

```
/*****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****************************************************************************
**
** SOURCE FILE NAME: dbrestore.sqc
**
** SAMPLE: How to restore a database from a backup
**
**         This program ends in ".sqc" even though it does not contain
**         embedded SQL statements. It links in  the embedded SQL utility
**         file for database connection and disconnection, so it needs the
**         embedded SQL extension for the precompiler.
**
**         Note:
**           You must be disconnected from the sample database to run
**           this program. To ensure you are, enter 'db2 connect reset'
**           on the command line prior to running dbrestore.
**
```

```
** DB2 API USED:
**         db2CfgSet -- Set Configuration
**         db2Restore -- Restore Database
**
** OUTPUT FILE: dbrestore.out (available in the online documentation)
******************************************************************************
**
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
******************************************************************************/
#include "utilrecov.c"

/* local function prototypes */
int DbBackupAndRestore(char *, char *, char *, char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  char restoredDbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("¥nTHIS SAMPLE SHOWS HOW TO RESTORE A DATABASE FROM A¥n");
  printf("BACKUP.¥n");

  strcpy(restoredDbAlias, dbAlias);

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  printf("¥nNOTE: Backup images will be created on the server¥n");
  printf("      in the directory %s,¥n", serverWorkingPath);
  printf("      and will not be deleted by the program.¥n");

  /* prune the recovery history file */
  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);
  CHECKRC(rc, "DbRecoveryHistoryFilePrune");

  rc = DbBackupAndRestore(dbAlias,
                          restoredDbAlias, user, pswd, serverWorkingPath);
```

```
  CHECKRC(rc, "DbBackupAndRestore");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");

  return 0;
} /* end main */

int DbBackupAndRestore(char dbAlias[],
                       char restoredDbAlias[], char user[],
                       char pswd[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  unsigned short logretain = 0;
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1] = { 0 };

  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32 backupImageSize = 0;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;

  printf("¥n*************************************¥n");
  printf("*** BACK UP AND RESTORE A DATABASE ***¥n");
  printf("*************************************¥n");
  printf("¥nUSE THE DB2 APIs:¥n");
  printf("  db2CfgSet -- Set Configuration¥n");
  printf("  db2Backup -- Backup Database¥n");
  printf("  db2Restore -- Restore Database¥n");
  printf("TO BACK UP AND RESTORE A DATABASE.¥n");

  printf("¥n  Update ¥'%s¥' database configuration:¥n", dbAlias);
  printf("    - Disable the database configuration parameter LOGRETAIN¥n");
  printf("        i.e., set LOGRETAIN = OFF/NO¥n");

  /* initialize cfgParameters */
  /* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
     parameter 'logretain'; it is used to update the database configuration
     file */
  cfgParameters[0].flags = 0;
  cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
  cfgParameters[0].ptrvalue = (char *)&logretain;

  /* disable the database configuration parameter 'logretain' */
  logretain = SQLF_LOGRETAIN_DISABLE;

  /* initialize cfgStruct */
  cfgStruct.numItems = 1;
  cfgStruct.paramArray = cfgParameters;
  cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
  cfgStruct.dbname = dbAlias;

  /* set database configuration */
  db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
  DB2_API_CHECK("Db Log Retain -- Disable");

  /*****************************/
  /*    BACKUP THE DATABASE    */
  /*****************************/

  /* Calling up the routine for database backup */
```

```
| rc = DbBackup(dbAlias, user, pswd, serverWorkingPath, &backupStruct);
| CHECKRC(rc, "DbBackup");
|
| /*****************************/
| /*    RESTORE THE DATABASE    */
| /*****************************/
|
| strcpy(restoreTimestamp, backupStruct.oTimestamp);
|
| printf("¥n  Restoring a database ...¥n");
| printf("    - source image alias     : %s¥n", dbAlias);
| printf("    - source image time stamp: %s¥n", restoreTimestamp);
| printf("    - target database        : %s¥n", restoredDbAlias);
|
| rtablespaceStruct.tablespaces = NULL;
| rtablespaceStruct.numTablespaces = 0;
|
| rmediaListStruct.locations = &serverWorkingPath;
| rmediaListStruct.numLocations = 1;
| rmediaListStruct.locationType = SQLU_LOCAL_MEDIA;
|
| restoreStruct.piSourceDBAlias = dbAlias;
| restoreStruct.piTargetDBAlias = restoredDbAlias;
|
| restoreStruct.piTimestamp = restoreTimestamp;
| restoreStruct.piTargetDBPath = NULL;
| restoreStruct.piReportFile = NULL;
| restoreStruct.piTablespaceList = &rtablespaceStruct;
| restoreStruct.piMediaList = &rmediaListStruct;
| restoreStruct.piUsername = user;
| restoreStruct.piPassword = pswd;
| restoreStruct.piNewLogPath = NULL;
| restoreStruct.piVendorOptions = NULL;
| restoreStruct.iVendorOptionsSize = 0;
| restoreStruct.iParallelism = 1;
| restoreStruct.iBufferSize = 1024;     /*  1024 x 4KB */
| restoreStruct.iNumBuffers = 2;
| restoreStruct.iCallerAction = DB2RESTORE_RESTORE;
| restoreStruct.iOptions =
| DB2RESTORE_OFFLINE | DB2RESTORE_DB | DB2RESTORE_NODATALINK |
|   DB2RESTORE_NOROLLFWD;
|
| /* The API db2Restore is used to restore a database that has been backed
|    up using the API db2Backup. */
| db2Restore(db2Version810, &restoreStruct, &sqlca);
| EXPECTED_WARN_CHECK("database restore -- start");
|
| while (sqlca.sqlcode != 0)
| {
|   /* continue the restore operation */
|   printf("¥n  Continuing the restore operation...¥n");
|
|   /* depending on the sqlca.sqlcode value, user action may be
|      required, such as mounting a new tape */
|
|   restoreStruct.iCallerAction = DB2RESTORE_CONTINUE;
|
|   /* restore the database */
| db2Restore(db2Version810, &restoreStruct, &sqlca);
|   DB2_API_CHECK("database restore -- continue");
| }
|
| printf("¥n  Restore finished.¥n");
|
| return 0;
|} /* DbBackupAndRestore */
```

dbrollfwd サンプル・プログラム:

dbrollfwd サンプル・ファイルには、データベース・リストア操作後にロールフォワード操作を実行する方法が示されています。

```
/****************************************************************************
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 2003
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
****************************************************************************
**
** SOURCE FILE NAME: dbrollfwd.sqc
**
** SAMPLE: How to perform rollforward after restore of a database
**
**         This program ends in ".sqc" even though it does not contain
**         embedded SQL statements. It links in  the embedded SQL utility
**         file for database connection and disconnection, so it needs the
**         embedded SQL extension for the precompiler.
**
**         Note:
**           You must be disconnected from the sample database to run
**           this program. To ensure you are, enter 'db2 connect reset'
**           on the command line prior to running dbrollfwd.
**
** DB2 APIs USED:
**         db2CfgSet -- Set Configuration
**         db2Restore -- Restore Database
**         db2Rollforward -- Rollforward Database
**
** OUTPUT FILE: dbrollfwd.out (available in the online documentation)
****************************************************************************
**
** For detailed information about database backup and database recovery, see
** the Data Recovery and High Availability Guide and Reference. This manual
** will help you to determine which database and table space recovery methods
** are best suited to your business environment.
**
** For more information on the sample programs, see the README file.
**
** For information on developing C applications, see the Application
** Development Guide.
**
** For information on using SQL statements, see the SQL Reference.
**
** For information on DB2 APIs, see the Administrative API Reference.
**
** For the latest information on programming, building, and running DB2
** applications, visit the DB2 application development website:
**     http://www.software.ibm.com/data/db2/udb/ad
****************************************************************************/
#include "utilrecov.c"

/* local function prototypes */
int DbBackupRestoreAndRollforward(char *, char *, char *, char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1] = { 0 };
```

```
  char serverWorkingPath[SQL_PATH_SZ + 1] = { 0 };
  char rolledForwardDbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char dbAlias[SQL_ALIAS_SZ + 1] = { 0 };
  char user[USERID_SZ + 1] = { 0 };
  char pswd[PSWD_SZ + 1] = { 0 };

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  CHECKRC(rc, "CmdLineArgsCheck3");

  printf("¥nTHIS SAMPLE SHOWS HOW TO PERFORM ROLLFORWARD AFTER¥n");
  printf("RESTORE OF A DATABASE.¥n");
  strcpy(rolledForwardDbAlias, "RFDB");

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  CHECKRC(rc, "Instance Attach");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  CHECKRC(rc, "ServerWorkingPathGet");

  printf("¥nNOTE: Backup images will be created on the server¥n");
  printf("      in the directory %s,¥n", serverWorkingPath);
  printf("      and will not be deleted by the program.¥n");

  rc = DbBackupRestoreAndRollforward(dbAlias, rolledForwardDbAlias, user,
                                     pswd, serverWorkingPath);
  CHECKRC(rc, "DbBackupRestoreAndRollforward");

  /* Detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  CHECKRC(rc, "InstanceDetach");

  return 0;
} /* end main */

int DbBackupRestoreAndRollforward(char dbAlias[],
                                  char rolledForwardDbAlias[],
                                  char user[], char pswd[],
                                         char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  unsigned short logretain = 0;
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1] = { 0 };
  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32  backupImageSize;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;
  db2RfwdInputStruct   rfwdInput;
  db2RfwdOutputStruct  rfwdOutput;
  db2RollforwardStruct  rfwdStruct;
  char rollforwardAppId[SQLU_APPLID_LEN + 1] = { 0 };
  sqlint32 numReplies = 0;
  struct sqlurf_info nodeInfo;

  printf("¥n****************************¥n");
  printf("*** ROLLFORWARD RECOVERY ***¥n");
  printf("****************************¥n");
  printf("¥nUSE THE DB2 APIs:¥n");
  printf("  db2CfgSet -- Set Configuration¥n");
```

```
printf("  db2Backup -- Backup Database¥n");
printf("  sqlecrea -- Create Database¥n");
printf("  db2Restore -- Restore Database¥n");
printf("  db2Rollforward -- Rollforward Database¥n");
printf("  sqledrpd -- Drop Database¥n");
printf("TO BACK UP, RESTORE, AND ROLLFORWARD A DATABASE.¥n");
printf("¥n  Update ¥'%s¥' database configuration:¥n", dbAlias);
printf("    - Enable the configuration parameter LOGRETAIN ¥n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES¥n");

/* initialize cfgParameters */
cfgParameters[0].flags = 0;
cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
cfgParameters[0].ptrvalue = (char *)&logretain;

/* enable the configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* initialize cfgStruct */
cfgStruct.numItems = 1;
cfgStruct.paramArray = cfgParameters;
cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
cfgStruct.dbname = dbAlias;

/* get database configuration */
db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/*****************************/
/*    BACKUP THE DATABASE    */
/*****************************/

/* Calling the routine for database backup */
rc = DbBackup(dbAlias, user, pswd, serverWorkingPath, &backupStruct);
CHECKRC(rc, "DbBackup");

/* To restore a remote database, you will first need to create an empty database
   if the client's code page is different from the server's code page.
   If this is the case, uncomment the call to DbCreate(). It will create
   an empty database on the server with the server's code page. */

/*
rc = DbCreate(dbAlias, rolledForwardDbAlias);
CHECKRC(rc, "DbCreate");
*/

/*****************************/
/*    RESTORE THE DATABASE    */
/*****************************/
strcpy(restoreTimestamp, backupStruct.oTimestamp);
rtablespaceStruct.tablespaces = NULL;
rtablespaceStruct.numTablespaces = 0;
rmediaListStruct.locations = &serverWorkingPath;
rmediaListStruct.numLocations = 1;
rmediaListStruct.locationType = SQLU_LOCAL_MEDIA;
restoreStruct.piSourceDBAlias = dbAlias;
restoreStruct.piTargetDBAlias = rolledForwardDbAlias;
restoreStruct.piTimestamp = restoreTimestamp;
restoreStruct.piTargetDBPath = NULL;
restoreStruct.piReportFile = NULL;
restoreStruct.piTablespaceList = &rtablespaceStruct;
restoreStruct.piMediaList = &rmediaListStruct;
restoreStruct.piUsername = user;
restoreStruct.piPassword = pswd;
restoreStruct.piNewLogPath = NULL;
restoreStruct.piVendorOptions = NULL;
restoreStruct.iVendorOptionsSize = 0;
```

```
                        restoreStruct.iParallelism = 1;
                        restoreStruct.iBufferSize = 1024;     /*  1024 x 4KB */
                        restoreStruct.iNumBuffers = 2;
                        restoreStruct.iCallerAction = DB2RESTORE_RESTORE;
                        restoreStruct.iOptions =
                        DB2RESTORE_OFFLINE | DB2RESTORE_DB | DB2RESTORE_NODATALINK |
                        DB2RESTORE_ROLLFWD;

                        printf("¥n  Restoring a database ...¥n");
                        printf("    - source image alias     : %s¥n", dbAlias);
                        printf("    - source image time stamp: %s¥n", restoreTimestamp);
                        printf("    - target database        : %s¥n", rolledForwardDbAlias);

                        /* The API db2Restore is used to restore a database that has been backed
                           up using the API db2Backup. */
                        db2Restore(db2Version810, &restoreStruct, &sqlca);
                        DB2_API_CHECK("database restore -- start");
                        while (sqlca.sqlcode != 0)

                        {

                          /* continue the restore operation */
                          printf("¥n  Continuing the restore operation...¥n");

                          /* Depending on the sqlca.sqlcode value, user action may be
                             required, such as mounting a new tape. */
                          restoreStruct.iCallerAction = DB2RESTORE_CONTINUE;

                          /* restore the database */
                        db2Restore(db2Version810, &restoreStruct, &sqlca);
                          DB2_API_CHECK("database restore -- continue");
                        }
                        printf("¥n  Restore finished.¥n");

                        /*****************************/
                        /*    ROLLFORWARD RECOVERY    */
                        /*****************************/
                        printf("¥n  Rolling forward database '%s'...¥n", rolledForwardDbAlias);
                        rfwdInput.iVersion = SQLUM_RFWD_VERSION;
                        rfwdInput.piDbAlias = rolledForwardDbAlias;
                        rfwdInput.iCallerAction = DB2ROLLFORWARD_RFWD_STOP;
                        rfwdInput.piStopTime = SQLUM_INFINITY_TIMESTAMP;
                        rfwdInput.piUserName = user;
                        rfwdInput.piPassword = pswd;
                        rfwdInput.piOverflowLogPath = serverWorkingPath;
                        rfwdInput.iNumChngLgOvrflw = 0;
                        rfwdInput.piChngLogOvrflw = NULL;
                        rfwdInput.iConnectMode = DB2ROLLFORWARD_OFFLINE;
                        rfwdInput.piTablespaceList = NULL;
                        rfwdInput.iAllNodeFlag = DB2_ALL_NODES;
                        rfwdInput.iNumNodes = 0;
                        rfwdInput.piNodeList = NULL;
                        rfwdInput.piDroppedTblID = NULL;
                        rfwdInput.piExportDir = NULL;
                        rfwdInput.iNumNodeInfo = 1;
                        rfwdInput.iRollforwardFlags = DB2ROLLFORWARD_EMPTY_FLAG;
                        rfwdOutput.poApplicationId = rollforwardAppId;
                        rfwdOutput.poNumReplies = &numReplies;
                        rfwdOutput.poNodeInfo = &nodeInfo;
                        rfwdStruct.piRfwdInput = &rfwdInput;
                        rfwdStruct.poRfwdOutput = &rfwdOutput;

                        /* rollforward database */
                        /* The API db2Rollforward rollforward recovers a database by
                           applying transactions recorded in the database log files. */
                        db2Rollforward(db2Version810, &rfwdStruct, &sqlca);
                        DB2_API_CHECK("rollforward -- start");
```

```
                        printf("  Rollforward finished.¥n");

                        /* drop the restored database */
                        rc = DbDrop(rolledForwardDbAlias);
                        CHECKRC(rc, "DbDrop");
                        return 0;
                      } /* DbBackupRestoreAndRollforward */
```

付録 F. Tivoli Storage Manager

BACKUP DATABASE コマンドまたは RESTORE DATABASE コマンドを呼び出す時に、 Tivoli Storage Manager (TSM) 製品を使用してデータベースまたは表スペースのバックアップの管理またはリストア操作の管理を行うことを指定できます。以下のシステムを除き、TSM クライアント API の最低限必要なレベルは、バージョン 4.2.0 です。

- 64 ビット Solaris システムでは、TSM クライアント API バージョン 4.2.1 が必要です。
- 64 ビット Windows NT システムでは、TSM クライアント API バージョン 5.1 が必要です。
- SuSE Linux Enterprise Server 8 for IA64 および LinuxPPC では、 TSM クライアント API バージョン 5.1.5 以上が必要です。
- 64 ビット Linux on AMD Opteron システムでは、 TSM クライアント API バージョン 5.2.0 以上が必要です。
- 64 ビット Linux for S/390 では、 TSM クライアント API バージョン 5.2.2 以上が必要です。

Tivoli Storage Manager クライアントの構成

データベース・マネージャーで TSM オプションを使用できるようにするには、TSM 環境を構成するために以下のステップを行うことが必要になるかもしれません。

1. 機能している TSM クライアントおよびサーバーをインストールし、構成する必要があります。加えて、TSM クライアント API を各 DB2 サーバーにインストールする必要があります。
2. TSM API で使う環境変数を設定します。

   DSMI_DIR API トラステッド・エージェント・ファイル (`dsmtca`) が存在しているユーザー定義のディレクトリー・パスを識別します。

   DSMI_CONFIG
   `dsm.opt` ファイルへのユーザー定義ディレクトリー・パスを識別します。これには、TSM ユーザー・オプションが含まれています。他の 2 つの変数とは異なり、この変数には完全修飾パスおよびファイル名を含めなければなりません。

   DSMI_LOG エラー・ログ (`dsierror.log`) が作成されるユーザー定義のディレクトリー・パスを指定します。

   注: 複数パーティション・データベース環境では、これらの設定は、`sqllib/userprofile` ディレクトリー内で指定する必要があります。
3. それらの環境変数に変更がなされ、データベース・マネージャーが稼働している場合は、以下のようにしてください。
   - **db2stop** コマンドを使用して、データベース・マネージャーを停止します。
   - **db2start** コマンドを使用して、データベース・マネージャーを開始します。

4. サーバーの構成によっては、 Tivoli クライアントは TSM サーバーとインターフェースを持つためにパスワードが必要になるかもしれません。 TSM 環境が PASSWORDACCESS=generate を使用するように構成されている場合には、 Tivoli クライアントは、パスワードを設定する必要があります。

   実行可能ファイル dsmapipw は、インスタンス所有者の sqllib/adsm ディレクトリーにインストールされています。この実行可能ファイルにより、TSM パスワードの確立と再設定が可能になります。

   dsmapipw コマンドを実行するには、ローカル管理者または "root" ユーザーにログインする必要があります。このコマンドを実行すると、以下の情報を入力するよう求められます。

   - *旧パスワード。* TSM サーバーで認識されている、TSM ノードの現在のパスワード。このコマンドの初回実行時には、このパスワードは、ご使用のノードが TSM サーバーに登録された時に TSM 管理者から提供されたものになります。
   - *新規パスワード。* TSM サーバーに保管させる、TSM ノードの新しいパスワード。 (新規パスワードは、入力エラーがないかどうかを調べるため、2 回入力するよう求められます。)

   注: BACKUP DATABASE または RESTORE DATABASE コマンドを呼び出すユーザーは、このパスワードを知っている必要はありません。 dsmapipw コマンドを実行する必要があるのは、初期接続時のパスワードを確立する場合、および TSM サーバーのパスワードが初期化された場合のみです。

Tivoli Storage Manager を使用する際の考慮事項

TSM 内部の特定の機能を使用するには、その機能を使用するオブジェクトの完全修飾パス名を指定する必要があります。 (Windows オペレーティング・システムでは、/ の代わりに ¥ が使用されることに注意してください。) 完全修飾パス名は、次のようになります。

- データベース全体のバックアップ・オブジェクト
  /<database>/NODEnnnn/FULL_BACKUP.timestamp.seq_no
- 増分データベース・バックアップ・オブジェクト
  /<database>/NODEnnnn/DB_INCR_BACKUP.timestamp.seq_no
- 増分差分データベース・バックアップ・オブジェクト
  /<database>/NODEnnnn/DB_DELTA_BACKUP.timestamp.seq_no
- 満杯表スペース・バックアップ・オブジェクト
  /<database>/NODEnnnn/TSP_BACKUP.timestamp.seq_no
- 増分表スペース・バックアップ・オブジェクト
  /<database>/NODEnnnn/TSP_INCR_BACKUP.timestamp.seq_no
- 増分差分表スペース・バックアップ・オブジェクト
  /<database>/NODEnnnn/TSP_DELTA_BACKUP.timestamp.seq_no

ここで、<database> はデータベースの別名で、 NODEnnnn はノード番号です。大文字で表記されている名前は大文字で入力しなければなりません。

- 複数のバックアップ・イメージで同じデータベース別名が使用されている場合は、完全修飾名は、タイム・スタンプとシーケンス番号で区別されます。使用するバックアップ・バージョンを判別するには、TSM を照会する必要があります。
- 個々のバックアップ・イメージは、TSM が管理するファイル・スペースにプールされます。個別のバックアップ・イメージは、TSM API を介して、またはこれらの API を使用する **db2adutl** を介してのみ、操作できます。
- サーバーの構成ファイルの COMMTIMEOUT パラメーターで指定されている一定時間以内に Tivoli クライアントからの応答がないと、セッションは TSM サーバーによりタイムアウトになります。タイムアウト問題には、以下の 3 つの要素が起因している可能性があります。
  - TSM サーバーの COMMTIMEOUT パラメーターの設定値が低すぎる可能性がある。たとえば、リストア操作中に大きな DMS 表スペースが作成されていると、タイムアウトが起こる可能性があります。このパラメーターの推奨値は 6000 秒です。
  - DB2 バックアップまたはリストアのバッファーが大きすぎる可能性がある。
  - オンライン・バックアップ操作中のデータベース活動が多すぎる可能性がある。
- 複数セッションを使用して、スループットを増加させます (TSM サーバー上で十分なハードウェアが使用可能な場合のみ)。

関連概念:

- 49 ページの『ログ・ファイルの管理』
- 「*Data Links Manager 概説およびインストール*」の『Tivoli Space Manager Hierarchical Storage Manager (AIX)』

関連資料:

- 275 ページの『db2adutl - TSM 内の DB2 オブジェクトの管理』

付録 G. データベース・リカバリー用のユーザー出口

ユーザー出口プログラム を設定して、ログ・ファイルのアーカイブと検索を自動化することができます。ユーザー出口プログラムを呼び出してログ・ファイルのアーカイブや検索を行う前に、 *logarchmeth1* データベース構成パラメーターが USEREXIT に設定されていることを確認してください。これにより、データベースをロールフォワード・リカバリーすることもできるようになります。

ユーザー出口プログラムが呼び出されると、データベース・マネージャーは実行可能ファイルの db2uext2 に制御を渡します。データベース・マネージャーはパラメーターを db2uext2 に渡し、完了時にはプログラムがデータベース・マネージャーに戻りコードを渡します。データベース・マネージャーは戻り条件の限定された集まりを扱うので、ユーザー出口プログラムがエラー状態を扱う必要があります (377 ページの『エラー処理』 を参照)。データベース・マネージャー・インスタンス内ではユーザー出口プログラムは 1 つしか呼び出すことができないので、実行を要求される可能性のあるそれぞれの操作につき 1 つのセクションがなければなりません。

以下のトピックについて説明します。

ユーザー出口プログラムの例

サポートされているすべてのプラットフォーム用に、サンプル・ユーザー出口プログラムが提供されています。これらのプログラムを変更して、特定の必要に合わせることができます。サンプル・プログラムには、最も効果的に使用するために役立つ情報のコメントが付けられています。

ユーザー出口プログラムは、ログ・ファイルをアクティブ・ログ・パスからアーカイブ・ログ・パスにコピーする ものでなければならないことに注意してください。アクティブ・ログ・パスからログ・ファイルを削除しないでください。 (これはデータベース・リカバリー時に問題を引き起こすことがあります。) DB2® は、アーカイブ・ログ・ファイルを、リカバリーのために必要がなくなったときにアクティブ・ログ・パスから除去します。

以下に、DB2 に同梱されているサンプル・ユーザー出口プログラムについて説明します。

- **UNIX® ベースのシステム**

  UNIX ベースのシステム用の DB2 のユーザー出口サンプル・プログラムは、sqllib/samples/c サブディレクトリーにあります。提供されているサンプルは C 言語でコーディングされていますが、ユーザー出口プログラムは別のプログラミング言語で作成することもできます。

ユーザー出口プログラムは、実行可能ファイルで、その名前は db2uext2 でなければなりません。

UNIX ベースのシステムには 4 つのサンプル・ユーザー出口プログラムがあります。

– db2uext2.ctsm

  このサンプルは Tivoli® Storage Manager を使用して、データベース・ログ・ファイルのアーカイブと検索を行います。

– db2uext2.ctape

  このサンプルはテープ・メディアを使用して、データベース・ログ・ファイルのアーカイブと検索を行います。

– db2uext2.cdisk

  このサンプルはオペレーティング・システムの COPY コマンドおよびディスク・メディアを使用して、データベース・ログ・ファイルのアーカイブと検索を行います。

– db2uxt2.cxbsa

  このサンプルは、X/Open グループにより発行された XBSA Draft 0.8 で作動します。データベース・ログ・ファイルのアーカイブと検索に使用できます。このサンプルは、AIX でのみサポートされています。

• **Windows® オペレーティング・システム**

Windows オペレーティング・システム用の DB2 のユーザー出口サンプル・プログラムは、 sqllib¥samples¥c サブディレクトリーにあります。提供されているサンプルは C 言語でコーディングされていますが、ユーザー出口プログラムは別のプログラミング言語で作成することもできます。

ユーザー出口プログラムは、実行可能ファイルで、その名前は db2uext2 でなければなりません。

Windows オペレーティング・システムには 2 つのサンプル・ユーザー出口プログラムがあります。

– db2uext2.ctsm

  このサンプルは Tivoli Storage Manager を使用して、データベース・ログ・ファイルのアーカイブと検索を行います。

– db2uext2.cdisk

  このサンプルはオペレーティング・システムの COPY コマンドおよびディスク・メディアを使用して、データベース・ログ・ファイルのアーカイブと検索を行います。

呼び出し形式

データベース・マネージャーは、ユーザー出口プログラムを呼び出す時にパラメーターのセット (データ・タイプは CHAR) をプログラムに渡します。呼び出し形式は、ご使用のオペレーティング・システムによって異なります。

```
db2uext2 -OS<os> -RL<db2rel> -RQ<request> -DB<dbname>
-NN<nodenum> -LP<logpath> -LN<logname> -AP<tsmpasswd>
-SP<startpage> -LS<logsize>
```

os インスタンスが実行されているプラットフォームを指定します。有効な値は次のとおりです。 AIX®、Solaris、HP-UX、SCO、Linux、および NT。

db2rel DB2 リリース・レベルを指定します。たとえば、SQL07020。

request 要求タイプを指定します。有効な値は次のとおりです。 ARCHIVE および RETRIEVE.

dbname データベース名を指定します。

nodenum ローカル・ノード番号を指定します。たとえば、5。

logpath ログ・ファイルの完全修飾パスを指定します。このパスは、末尾にパス区切り記号を含んでいる必要があります。たとえば、次のとおりです。 /u/database/log/path/ または d:¥logpath¥

logname アーカイブまたは検索されるログ・ファイルの名前を指定します。たとえば S0000123.LOG のようになります。

tsmpasswd TSM パスワードを指定します。 (データベース構成パラメーター *tsm_password* の値が以前に指定されている場合は、その値がユーザー出口プログラムに渡されます。)

startpage ログ・エクステントが始まる装置の、4 KB のオフセット・ページの数を指定します。

logsize ログ・エクステントのサイズを 4 KB のページ数で指定します。パラメーターは、ロー・デバイスを使用してロギングを取る場合に限り有効です。

エラー処理

ユーザー出口プログラムは、意味のある特定の戻りコードを提供するように設計して、データベース・マネージャーがそれらを正しく解釈できるようにしなければなりません。ユーザー出口プログラムは基礎をなすオペレーティング・システムのコマンド・プロセッサーに呼び出されるので、オペレーティング・システム自体からエラー・コードが戻されることがあります。また、これらのエラー・コードは再マップされていないため、オペレーティング・システムのメッセージ・ヘルプ・ユーティリティーを使用して情報を入手してください。

378 ページの表 8 では、ユーザー出口プログラムによって戻されることのある戻りコードを示し、これらのコードがどのようにデータベース・マネージャーによって解釈されるかを説明します。戻りコードが表にリストされていない場合は、値 32 の場合と同じように扱われます。

表 8. ユーザー出口プログラム戻りコード： アーカイブおよび検索操作にのみ適用されます。

戻りコード	説明
0	正常実行。
4	一時リソース・エラーが検出された。[a]
8	オペレーター要介入。[a]
12	ハードウェア・エラー。[b]
16	ユーザー出口プログラムまたはそのプログラムで使用されているソフトウェア機能にエラー。[b]
20	ユーザー出口プログラムに渡された 1 つまたは複数のパラメーターにエラー。指定されたパラメーターをユーザー出口プログラムが正しく処理しているか調べてください。[b]
24	ユーザー出口プログラムが検出されなかった。 [b]
28	入出力 (I/O) の失敗またはオペレーティング・システムが原因で生じたエラー。[b]
32	ユーザー出口プログラムがユーザーにより終了させられた。[b]
255	ユーザー出口プログラムが、実行可能ファイルのライブラリー・ファイルをロードできなかったことが原因で生じたエラー。[c]

表 8. ユーザー出口プログラム戻りコード (続き): アーカイブおよび検索操作にのみ適用されます。

戻りコード	説明
[a] アーカイブおよび検索要求の場合、戻りコードが 4 または 8 であれば、5 分以内に再試行が行われます。ユーザー出口プログラムによって、同じログ・ファイルへの検索要求に 4 または 8 が継続して戻されると、DB2 は成功するまで再試行を継続します。 (このことはロールフォワード操作、または複製ユーティリティーが使用する **db2ReadLog** API への呼び出しについても当てはまります。) [b] ユーザー出口要求は、5 分間中断します。この間は、エラー状態を引き起こした要求を含めて、すべての要求が無視されます。この 5 分間の中断の後、次の要求が処理されます。この要求がエラーなしに処理された場合、新規ユーザー出口要求は継続します。 DB2 は、以前に失敗または中断されたアーカイブ要求を再発行します。再試行時に 8 より大きい戻りコードが生成される場合、要求はさらに 5 分間中断されます。このような 5 分間の中断は、問題が訂正されるまで、あるいはデータベースを停止して再始動するまで繰り返されます。すべてのアプリケーションがデータベースから切断されると、DB2 は、以前に正常にアーカイブされなかった可能性のあるすべてのログ・ファイルに対するアーカイブ要求を発行します。ユーザー出口プログラムがログ・ファイルのアーカイブに失敗した場合、使用しているディスクがログ・ファイルでいっぱいになることがあり、パフォーマンスが低下することがあります。ディスクがいっぱいになると、データベース・マネージャーはデータベース更新に関するアプリケーション要求を受け入れなくなります。ログ・ファイルを検索するためにユーザー出口プログラムが呼び出されると、ロールフォワード・リカバリーは中断されますが、 ROLLFORWARD STOP オプションが指定されていない限り停止することはありません。停止 (STOP) オプションが指定されていなかった場合は、問題を訂正してリカバリーを再開することができます。 [c] ユーザー出口プログラムによってエラー・コード 255 が戻された場合、プログラムが実行可能ファイルのライブラリー・ファイルをロードできなかった可能性があります。これを検証するには、ユーザー出口プログラムを手操作で起動してください。さらに情報が表示されます。 **注:** アーカイブおよび検索操作中、0、および 4 以外のすべての戻りコードについて、アラート・メッセージが発行されます。アラート・メッセージには、ユーザー出口プログラムからの戻りコード、およびユーザー出口プログラムに備えられた入力パラメーターのコピーが含まれています。	

付録 H. ベンダー製品用のバックアップおよびリストア API

Storage Manager に対するバックアップおよびリストアの API

DB2 では、サード・パーティーのメディア管理製品が、バックアップおよびリストア操作およびログ・ファイルのデータを保管および検索するために使用できるインターフェースが用意されています。この機能は、DB2 の標準部分としてサポートされている、ディスケット、ディスク、テープ、および Tivoli Storage Manager のバックアップ、リストア、およびログ・アーカイブ・データのターゲットを拡大できるように設計されています。

このようなサード・パーティーのメディア管理製品は、この付録ではベンダー製品と呼ばれています。

DB2 では、多くのベンダーが使用できる汎用データ・インターフェースを提供する関数プロトタイプのセットが定義されており、これらを用いてバックアップ、リストア、ログ・アーカイブを行います。これらの関数は、共用ライブラリー (UNIX ベースのシステムの場合) または DLL (Windows オペレーティング・システムの場合) において、ベンダーにより提供されます。関数が DB2 によって呼び出されると、バックアップ、リストア、またはログ・アーカイブ呼び出しルーチンによって指定された共用ライブラリーまたは DLL がロードされ、ベンダー提供の関数が呼び出されて必要なタスクを実行します。

DB2 ベンダー機能を紹介するサンプル・ファイルは、UNIX プラットフォームでは `sqllib/samples/BARVendor` ディレクトリーに、 Windows では `sqllib\samples\BARVendor` ディレクトリーにあります。

操作概要

DB2 とベンダー製品の間のインターフェースのために、 7 つの関数が定義されています。

- sqluvint - 初期設定と装置へのリンク
- sqluvget - 装置からのデータの読み取り
- sqluvput - 装置へのデータの書き込み
- sqluvend - 装置へのリンク解除
- sqluvdel - コミット済みセッションの削除
- db2VendorQueryApiVersion - 装置でサポートされる API レベルの照会
- db2VendorGetNextObj - 装置での次のオブジェクトの入手

DB2 がこれらの関数を呼び出したら、これらの関数は共用ライブラリー (UNIX ベースのシステムの場合) または DLL (Windows オペレーティング・システムの場合) において、ベンダー製品から提供されなければなりません。

注: 共用ライブラリーまたは DLL コードは、データベース・エンジン・コードの一部として実行されます。したがって、再入可能でなければならず、徹底的に

デバッグされなければなりません。関数に誤りがあると、データベースのデータ保全性を危険にさらす可能性があります。

特定のバックアップまたはリストア操作時に DB2 が呼び出す関数の順序は、以下の要因によって異なります。

- 使用されるセッションの数。
- 操作の内容 (バックアップ、リストア、ログ・アーカイブ、またはログ検索)。
- バックアップまたはリストア操作で指定された PROMPTING モード。
- データが格納されている装置の特性。
- 操作中に生じたエラー。

セッションの数

DB2 は、1 つまたは複数のデータ・ストリームまたはセッションを使用したデータベース・オブジェクトのバックアップおよびリストアをサポートしています。バックアップまたはリストアに 3 つのセッションを使用しているときには、 3 つの物理装置または LU が使用できなければなりません。ベンダー装置サポートが使用されているときには、それぞれの物理装置または LU へのインターフェースを管理するのはベンダーの関数です。 DB2 の側では、ベンダー提供の関数との間でデータ・バッファーの送受信を行うだけです。

使用されるセッションの数は、データベースのバックアップまたはリストア関数を呼び出すアプリケーションによってパラメーターとして指定されます。この値は、sqluvint によって使用される INIT-INPUT 構造で提供されます。

DB2 は、指定された数値に達するか、または sqluvint 呼び出しから SQLUV_MAX_LINK_GRANT 警告戻りコードを受信するまで、セッションの初期設定を継続します。サポートされているセッションの最大数に達したことを DB2 へ警告するために、ベンダー製品にはアクティブ・セッションの数を追跡するためのコードが必要です。 DB2 への警告が失敗すると、DB2 のセッションの初期設定要求が失敗し、結果としてすべてのセッションが終了し、バックアップまたはリストア操作全体が失敗に終わる可能性があります。

この操作がバックアップであれば、DB2 は各セッションの開始時にメディア・ヘッダー・レコードを書き込みます。このレコードには、DB2 がリストア操作時にセッションを識別するために使用する情報が入ります。 DB2 は、バックアップ・イメージの名前にシーケンス番号を付加することによって、各セッションを固有に識別します。この番号は、最初のセッションを 1 として始まり、バックアップまたはリストア操作の sqluvint 呼び出しで別のセッションが開始されるたびに 1 ずつ増加します。

バックアップ操作が正常に完了すると、DB2 はクローズする最後のセッションにメディア・トレーラーを書き込みます。このトレーラーには、バックアップ操作を実行するために使用したセッションの数を DB2 に通知する情報が組み込まれています。この情報は、リストア操作時に、すべてのセッションまたはデータ・ストリームがリストアされたことを確認するために使用されます。

エラー、警告、またはプロンプトなしの操作

バックアップの場合、DB2 は、*それぞれの* セッションごとに次の順序の呼び出しを発行します。

```
sqluvint, action = SQLUV_WRITE
```

続いて、1 個 ～ n 個の

```
sqluvput
```

続いて、1 個の

```
sqluvend, action = SQLUV_COMMIT
```

DB2 が sqluvend 呼び出し (アクション SQLUV_COMMIT) を発行するときには、ベンダー製品が出力データを適切に保管することを予期します。 DB2 へ SQLUV_OK の戻りコードが戻されれば、成功です。

sqluvint 呼び出しで使用される DB2-INFO 構造には、バックアップを識別するのに必要な情報が含まれます。シーケンス番号が提供されます。ベンダー製品は、この情報を保管することを選択する場合があります。 DB2 は、この情報をリストア時に使用し、リストアされるバックアップを識別します。

リストアの場合、セッションごとの呼び出しの順序は次のとおりです。

```
sqluvint, action = SQLUV_READ
```

続いて、1 個 ～ n 個の

```
sqluvget
```

続いて、1 個の

```
sqluvend, action = SQLUV_COMMIT
```

sqluvint 呼び出しで使用される DB2-INFO 構造内の情報には、バックアップを識別するのに必要な情報が含まれます。シーケンス番号は提供されません。 DB2 はすべてのバックアップ・オブジェクト (バックアップ時にコミットされたセッション出力) が戻されることを予期します。最初に戻されるバックアップ・オブジェクトは、シーケンス番号 1 で生成されたオブジェクトです。他のすべてのオブジェクトは順番に関係なくリストアされます。 DB2 は、すべてのオブジェクトが処理されたかどうか確認するために、メディアの末尾をチェックします。

注: すべてのベンダー製品が、バックアップ・オブジェクトの名前のレコードを保持するわけではありません。これは、テープや、容量が限られているその他のメディアにバックアップが行われる場合に特に当てはまります。リストア・セッションの初期設定時に、識別情報を利用して、必要なバックアップ・オブジェクトを、それらが必要とされるときに使用可能になるように計画することができます。これは、バックアップの保管にジュークボックスまたはロボット・システムを使用する場合に最も役立ちます。 DB2 は、必ず正しいデータがリストアされるようにするために、必ずメディア・ヘッダー (各セッションの出力の先頭レコード) をチェックします。

プロンプト・モード

バックアップまたはリストア操作の開始時には、次の 2 つのプロンプト・モードが使用可能です。

- ベンダー製品がユーザーに対してメッセージを書き込んだり、ユーザーがそれらに応答する機会のない WITHOUT PROMPTING または NOINTERRUPT。
- ユーザーがベンダー製品からメッセージを受け取り、それに応答することができる PROMPTING または INTERRUPT。

PROMPTING モードの場合、バックアップおよびリストアについて次の 3 つの応答が可能です。

- 続行

  装置へのデータの読み取りまたは書き込みの操作が再開されます。
- 装置の終了

  装置が追加のデータを受信せず、セッションが終了します。
- 終了

  バックアップまたはリストア操作全体が終了します。

PROMPTING および WITHOUT PROMPTING モードの使用法については、以下のセクションで説明されています。

装置の特性

ベンダー装置サポート API のために、 2 つの一般タイプの装置が定義されています。

- メディアを交換するためのユーザー・アクションが必要とされる、容量が限られている装置。たとえばテープ・ドライブ、ディスケット、または CD-ROM ドライブ。
- 通常の操作ではユーザーがメディアを扱う必要のない大容量の装置。たとえば、ジュークボックスや高機能のロボット・メディア処理装置。

容量が限られている装置では、バックアップまたはリストア操作時に、追加メディアをロードするようユーザーに指示することが必要になる可能性があります。通常、DB2 では、バックアップまたはリストア操作のいずれにおいても、メディアがロードされる順序は重要ではありません。また、DB2 では、ベンダー・メディア処理メッセージをユーザーに渡すための機能も用意されています。このプロンプトでは、PROMPTING をオンにしてバックアップまたはリストア操作を開始することが必要です。メディア処理メッセージのテキストは、戻りコード構造の記述フィールドで指定されます。

PROMPTING がオンになっており、DB2 が sqluvput (書き込み) または sqluvget (読み取り) 呼び出しから SQLUV_ENDOFMEDIA または SQLUV_ENDOFMEDIA_NO_DATA の戻りコードを受信すると、 DB2 は次のことを行います。

- 呼び出しが sqluvput であれば、セッションに送信された最後のバッファーが再送されるようにマークする。これは、後でセッションに置かれます。

- sqluvend (アクション = SQLUV_COMMIT) を用いてセッションを呼び出す。成功すると (SQLUV_OK 戻りコード)、DB2 は次のことを行います。
  - メディア終端条件を示している戻りコード構造からのベンダー・メディア処理メッセージをユーザーに送信する。
  - 継続、装置終了、または終了の応答をするようユーザーに指示する。
- 応答が*継続* の場合、 DB2 は sqluvint 呼び出しを使用して別のセッションを初期設定します。成功すると、セッションへのデータの書き込みまたはセッションからのデータの読み取りを開始します。書き込み時にセッションを固有に識別するために、 DB2 はシーケンス番号を増加させます。シーケンス番号は、sqluvint で使用される DB2-INFO 構造内で使用できます。この番号は、セッションへ送信される最初のデータ・レコードであるメディア・ヘッダー・レコード内にあります。

  DB2 は、バックアップまたはリストア操作の開始時に要求された数、または sqluvint 呼び出し時に SQLUV_MAX_LINK_GRANT 警告を出してベンダー製品が表示した数よりも多くのセッションを開始しません。
- 応答が装置終了 の場合、 DB2 は別のセッションを初期設定せず、アクティブ・セッションの数が 1 減ります。 DB2 は、装置終了の応答によってすべてのセッションを終了させることはありません。バックアップまたはリストア操作が完了するまで、少なくとも 1 つのセッションがアクティブなままでなければなりません。
- 応答が終了 の場合は、 DB2 はバックアップまたはリストア操作を終了します。セッションを終了させるときの DB2 の処理については、 386 ページの『エラー条件が DB2 に戻される場合』を参照してください。

バックアップまたはリストア時のパフォーマンスは、使用している装置の数によって異なってくるため、並列処理を保持しておくことが重要になります。バックアップ操作の場合、残りのアクティブ・セッションで、書き込まれるデータが保持されることが分かっていない限り、継続の応答を行うようお勧めします。リストア操作の場合も、すべてのメディアが処理されるまで継続の応答を行うようお勧めします。

バックアップまたはリストア・モードが WITHOUT PROMPTING であり、 DB2 がセッションから SQLUV_ENDOFMEDIA または SQLUV_ENDOFMEDIA_NO_DATA の戻りコードを受信すると、 DB2 はセッションを終了し、別のセッションをオープンしません。バックアップまたはリストア操作が完了する前に、すべてのセッションが DB2 へメディア終端を戻すと、操作が失敗します。このため、容量が限られた装置で WITHOUT PROMPTING を使用する際には注意が必要です。ただし、大容量の装置にとっては、このモードで稼働することは有意義です。

ベンダー製品では、装置に限りなく容量があるように見えるように、メディアのマウントおよび交換アクションを DB2 から隠すことができます。一部の大容量装置は、このモードで稼働します。そのような場合は、バックアップされたすべてのデータがリストア操作の進行中に同じ順序で DB2 に戻されることが重要です。そうでなければ、データが消失する可能性がありますが、 DB2 には消失したデータを検出する方法がないため、リストア操作が成功したものと見なします。

DB2 は、各バッファーが 1 つのメディア (たとえば、テープ) に入れられるということを前提にして、データをベンダー製品へ書き込みます。ベンダー製品は、DB2 側には知らせずに、バッファーを複数のメディアに分割することができます。このような場合、複数のメディアから再構成したバッファーを DB2 に戻すのはベンダー製品の責任であるため、リストア操作中にメディアが処理される順序は重要になります。順序が正しくなければ、リストア操作は失敗します。

エラー条件が DB2 に戻される場合

バックアップまたはリストア操作時に、DB2 は、すべてのセッションが正常に完了するか、そうでなければバックアップまたはリストア操作全体が失敗することを予期します。セッションは、sqluvend (アクション = SQLUV_COMMIT) の呼び出し時に、 SQLUV_OK 戻りコードで DB2 に正常終了を知らせます。

リカバリー不能エラーが検出されると、セッションは DB2 によって終了されます。これらのエラーは DB2 エラーとなるか、ベンダー製品から DB2 へ戻されるエラーとなります。バックアップまたはリストア操作が完了するにはすべてのセッションが正常にコミットされなければならないため、 1 つが失敗すると、DB2 はその操作と関連した他のセッションも終了させてしまいます。

ベンダー製品が DB2 からの呼び出しに対してリカバリー不能を示す戻りコードで応答する場合、ベンダー製品は、オプションで、RETURN-CODE 構造の記述フィールドに置かれたメッセージ・テキストを使用して、追加情報を提供することができます。このメッセージ・テキストは、訂正の処置をとれるように DB2 情報と共に表示されます。

バックアップのシナリオとして、あるセッションが正常にコミットされたものの、バックアップ操作に関連した他のセッションでリカバリー不能エラーが発生したというケースが考えられます。バックアップ操作が成功するためにはすべてのセッションが正常に完了しなければならないため、 DB2 はコミットされたセッション内の出力データを削除する必要があります。 DB2 は sqluvdel 呼び出しを発行して、オブジェクトの削除を要求します。この呼び出しは入出力セッションとは見なされず、バックアップ・オブジェクトの削除に必要な接続を初期化したり終了したりする機能を果たします。

DB2-INFO 構造にはシーケンス番号は含まれていません。 sqluvdel は、 DB2-INFO 構造内の残りのパラメーターと一致するバックアップ・オブジェクトをすべて削除します。

警告条件

DB2 は、ベンダー製品から警告戻りコードを受信する可能性があります。たとえば、装置が作動不能である場合や、その他の訂正可能条件が生じた場合などです。これは、読み取りと書き込みの両方の操作に当てはまります。

sqluvput および sqluvget 呼び出し時に、ベンダーは戻りコードを SQLUV_WARNING に設定することができます。さらに、オプションで RETURN-CODE 構造の記述フィールドに置かれたメッセージ・テキストを使用して、追加情報を提供することができます。このメッセージ・テキストはユーザーに表示され、訂正の処置をとることができます。ユーザーは、3 つの方法 (継続、装置終了、または終了) の 1 つで応答することができます。

- 応答が*継続* の場合は、DB2 はバックアップ操作中に sqluvput を使用してバッファーへの再書き込みを試みます。リストア操作時に、DB2 は sqluvget 呼び出しを発行して、次のバッファーを読み取ります。
- 応答が*装置終了* または*終了* の場合、DB2 はリカバリー不能エラー発生後に応答するときと同じ方法で、バックアップまたはリストア操作全体を終了させます (たとえば、アクティブ・セッションを終了し、コミット済みセッションを削除する)。

操作上のヒント

このセクションでは、ベンダー製品を作成するためのヒントをいくつか提供します。

履歴ファイル

履歴ファイルは、データベース・リカバリー操作に役立てることができます。このファイルは、各データベースに関連しており、バックアップまたはリストア操作が行われるたびに自動的に更新されます。ファイル内の情報は、以下の機能を使用して表示、更新、または除去することができます。

- コントロール・センター
- コマンド行プロセッサー (CLP)
  - LIST HISTORY コマンド
  - UPDATE HISTORY FILE コマンド
  - PRUNE HISTORY コマンド
- API
  - db2HistoryOpenScan
  - db2HistoryGetEntry
  - db2HistoryCloseScan
  - db2HistoryUpdate
  - db2Prune

ファイルのレイアウトについては、db2HistData を参照してください。

バックアップ操作が完了すると、1 つまたは複数のレコードがファイルへ書き込まれます。バックアップ操作の出力がベンダー装置に送られ、LOAD キーワードが使用された場合、履歴レコードの DEVICE フィールドには `0` が入ります。バックアップ操作が TSM に送られた場合、DEVICE フィールドには `A` が入ります。LOCATION フィールドには、以下のいずれかが入ります。

- バックアップ操作が呼び出されたときに指定されたベンダー・ファイル名。
- ベンダー・ファイル名が指定されていない場合は、共用ライブラリーの名前。

このオプションの指定についての詳細は、 388 ページの『ベンダー製品を使用してバックアップまたはリストア操作を呼び出す』を参照してください。

LOCATION フィールドはコントロール・センター、CLP、または API を使用して更新できます。バックアップ・イメージを保存するために容量の限られた装置 (たとえば、取り外し可能メディア) が使用されており、そのメディアが異なる記憶場所 (たとえばオフ・サイト) へ物理的に移動された場合、バックアップ情報のロケー

ションを更新することができます。この場合に、リストア操作が必要になったら、履歴ファイルを使用してバックアップ・イメージを見つけることができます。

ベンダー製品を使用してバックアップまたはリストア操作を呼び出す

DB2 バックアップまたは DB2 リストア・ユーティリティーを以下の方法で呼び出す場合に、ベンダー製品を指定できます。

- コントロール・センター
- コマンド行プロセッサー (CLP)
- アプリケーション・プログラミング・インターフェース (API)。

コントロール・センター

コントロール・センターは、 DB2 と共に出荷されるデータベース管理用のグラフィカル・ユーザー・インターフェースです。

指定内容	バックアップまたはリストア操作用のコントロール・センター入力変数
ベンダー装置を使用することと、ライブラリー名	*Use Library*。ライブラリー名 (UNIX ベースのシステムの場合) または DLL 名 (Windows オペレーティング・システムの場合) を指定します。
セッションの数	*Sessions*
ベンダー・オプション	サポートされていません
ベンダー・ファイル名	サポートされていません
転送バッファー・サイズ	バックアップの場合: *Size of each Buffer*。リストアの場合: 適用されません。

コマンド行プロセッサー (CLP)

コマンド行プロセッサー (CLP) を使用して DB2 BACKUP DATABASE または RESTORE DATABASE コマンドを呼び出すことができます。

指定内容	コマンド行プロセッサー・パラメーター	
	バックアップ用	リストア用
ベンダー装置を使用することと、ライブラリー名	*library-name*	*shared-library*
セッションの数	*num-sessions*	*num-sessions*
ベンダー・オプション	サポートされていません	サポートされていません
ベンダー・ファイル名	サポートされていません	サポートされていません
転送バッファー・サイズ	*buffer-size*	*buffer-size*

バックアップおよびリストア API 関数呼び出し

2 つの API 関数呼び出しがバックアップおよびリストア操作をサポートしています。バックアップ用には db2Backup、リストア用には db2Restore です。

指定内容	API パラメーター (db2Backup および db2Restore)
ベンダー装置を使用することと、ライブラリー名	構造 *sqlu_media_list* にはメディア・タイプ SQLU_OTHER_MEDIA を指定し、構造 *sqlu_vendor* には *shr_lib* に共用ライブラリーまたは DLL を指定します。
セッションの数	構造 *sqlu_media_list* に *sessions* を指定します。
ベンダー・オプション	*PVendorOptions*
ベンダー・ファイル名	構造 *sqlu_media_list* にはメディア・タイプ SQLU_OTHER_MEDIA を指定し、構造 *sqlu_vendor* には *filename* にファイル名を指定します。
転送バッファー・サイズ	*BufferSize*

関連資料:

- 389 ページの『sqluvint - 初期設定と装置へのリンク』
- 392 ページの『sqluvget - 装置からのデータの読み取り』
- 394 ページの『sqluvput - 装置へのデータの書き込み』
- 396 ページの『sqluvend - 装置のリンク解除およびリソースの解放』
- 398 ページの『sqluvdel - コミット済みセッションの削除』
- 402 ページの『DB2-INFO』
- 404 ページの『VENDOR-INFO』
- 405 ページの『INIT-INPUT』
- 406 ページの『INIT-OUTPUT』
- 406 ページの『DATA』
- 407 ページの『RETURN-CODE』
- 399 ページの『db2VendorQueryApiVersion - 装置でサポートされる API レベルの照会』
- 400 ページの『db2VendorGetNextObj - 装置での次のオブジェクトの入手』

sqluvint - 初期設定と装置へのリンク

この関数は、 DB2 とベンダー製品との間の論理リンクの初期設定および確立に関する情報を提供するために呼び出されます。

権限:

以下のいずれかが必要です。

- *sysadm*
- *dbadm*

必要な接続:

データベース

API 組み込みファイル:

sql.h

C API 構文:

```
/* File: sqluvend.h */
/* API: Initialize and Link to Device */
/* ... */
int sqluvint (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API パラメーター:

Init_input
入力。ベンダー装置との論理リンクを確立するための、 DB2 が提供する情報を含む情報。

Init_output
出力。ベンダー装置が戻した出力を含む構造。

Return_code
出力。 DB2 へ渡される戻りコードと短いテキスト記述を含む構造。

使用上の注意:

それぞれのメディア入出力セッションごとに、 DB2 はこの関数を呼び出して装置ハンドルを取得します。何かの理由で初期設定時にベンダー関数にエラーが発生すると、戻りコードを通してそのことを知らせます。エラーを示す戻りコードであれば、DB2 は、 **sqluvend** 関数を呼び出して操作を終了させることがあります。可能な戻りコードと、これらのそれぞれに対する DB2 の反応については、戻りコード表 (391 ページの表 9 を参照) に記載されています。

INIT-INPUT 構造には、ベンダー製品がバックアップまたはリストアを続行できるかどうかを判別するために使用できるエレメントが含まれます。

- size_HI_order および size_LOW_order

  バックアップの見積サイズです。これらを使用すると、ベンダー装置がバックアップ・イメージのサイズを処理できるかどうかを判別することができます。また、バックアップを保存するために必要な取り外し可能メディアの容量を見積もることができます。問題が予期される場合は、最初の **sqluvint** 呼び出しで失敗した方が有益である可能性があります。

- req_sessions

  要求したセッションの数を見積サイズやプロンプト・レベルと関連付けて使用して、バックアップまたはリストア操作が可能かどうかを判別することができます。

- prompt_lvl

プロンプト・レベルは、取り外し可能メディアの変更 (たとえば、テープ・ドライブへ他のテープを入れる) などのアクションを指示できるかどうかをベンダーに示します。これは、ユーザーへの指示方法がないために操作が続行できないことを示唆する場合があります。

プロンプト・レベルが WITHOUT PROMPTING であり、取り外し可能メディアの容量が要求されたセッション数よりも大きい場合、 DB2 は操作を正常に完了することはできません。

DB2 は、DB2-INFO 構造内のフィールドを使用して、書き込まれているバックアップまたは読み取られるリストアを指定します。アクション = SQLUV_READ の場合、ベンダー製品は指定されたオブジェクトの存在を調べなければなりません。見つからなければ、DB2 が適切な処置をとれるように戻りコードが SQLUV_OBJ_NOT_FOUND に設定される必要があります。

初期設定が正常に完了した後、DB2 は他のデータ転送機能を発行して継続しますが、 **sqluvend** 呼び出しでいつでもセッションを終了させることができます。

戻りコード:

表 9. sqluvint についての有効な戻りコードと結果の DB2 アクション

ヘッダー・ファイルのリテラル	説明	予想される次の呼び出し	その他のコメント
SQLUV_OK	操作が成功した。	sqluvput、sqluvget (コメント参照)	アクション = SQLUV_WRITE であれば、次の呼び出しは sqluvput (データのバックアップ) です。アクション = SQLUV_READ であれば、SQLUV_OK を戻す前に、指定されたオブジェクトの存在を確認します。次の呼び出しは、sqluvget (データのリストア) になります。
SQLUV_LINK_EXIST	セッションがすでに活動化されている。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_COMM_ERROR	装置との通信エラー	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_INV_VERSION	DB2 とベンダー製品に互換性がない。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_INV_ACTION	無効なアクションが要求された。これは、パラメーターの組み合わせにより不可能な操作が生じたことを示すためにも使用される。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_NO_DEV_AVAIL	現在使用できる装置がない。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_OBJ_NOT_FOUND	指定されたオブジェクトが見つからない。これは、sqluvint 呼び出しでのアクションが 'R' (読み取り) であり、 DB2-INFO 構造で指定された基準に基づいて要求されたオブジェクトが見つからないときに使用される。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。

表 9. sqluvint についての有効な戻りコードと結果の DB2 アクション (続き)

ヘッダー・ファイルのリテラル	説明	予想される次の呼び出し	その他のコメント
SQLUV_OBJS_FOUND	2 つ以上のオブジェクトが、指定された基準に一致する。これは、sqluvint 呼び出しでのアクションが 'R' (読み取り) であり、 DB2-INFO 構造内の基準と一致するオブジェクトが 2 つ以上あるときに発生する。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_INV_USERID	指定されたユーザー ID が無効である。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_INV_PASSWORD	提供されたパスワードが無効である。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_INV_OPTIONS	ベンダー・オプション・フィールド内で無効なオプションが検出された。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_INIT_FAILED	初期設定が失敗し、セッションが終了される。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_DEV_ERROR	装置エラー	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_MAX_LINK_GRANT	最大数のリンクが確立された。	sqluvput、sqluvget (コメント参照)	これは、DB2 からの警告として扱われます。この警告は、サポート可能なセッションの最大数に達しているので、これ以上ベンダー製品とのセッションをオープンしないよう DB2 に知らせるものです (注意: これは、装置の可用性が原因となっている可能性もあります)。アクション = SQLUV_WRITE (BACKUP) であれば、次の呼び出しは sqluvput です。アクション = SQLUV_READ であれば、SQLUV_MAX_LINK_GRANT を戻す前に、指定されたオブジェクトの存在を確認します。次の呼び出しは、sqluvget (データのリストア) になります。
SQLUV_IO_ERROR	入出力エラー	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。
SQLUV_NOT_ENOUGH_SPACE	バックアップ・イメージ全体を格納するための十分なスペースがない (サイズの見積もりは 64 ビットのバイト数で提供される)。	ありません	セッションの開始に失敗しました。セッションに割り振られていたメモリーを解放し、終了します。セッションが確立されなかったため、sqluvend 呼び出しは受信されません。

sqluvget - 装置からのデータの読み取り

初期設定の後、この関数を呼び出して装置からデータを読み取ることができます。

権限:

以下のいずれかが必要です。

- *sysadm*

- *dbadm*

必要な接続:

データベース

API 組み込みファイル:

sqluvend.h

C API 構文:

```
/* 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 パラメーター:

pVendorCB
入力。 DATA 構造 (データ・バッファーを含む) および Return_code 用に割り振られたスペースを指すポインター。

Data 入出力。データ 構造を指すポインター。

Return_code
出力。 API 呼び出しからの戻りコード。

obj_num
検索するバックアップ・オブジェクトを指定します。

buff_size
使用するバッファー・サイズを指定します。

actual_buff_size
読み取られる、または書き込まれる実際のバイト数を指定します。この値は、実際に読み取られたデータのバイト数を示す出力に設定してください。

dataptr
データ・バッファーを指すポインター。

reserve
将来の利用のために予約されています。

使用上の注意:

この関数はリストア・ユーティリティーで使用されます。

戻りコード:

表 10. sqluvget についての有効な戻りコードと結果の DB2 アクション

ヘッダー・ファイルのリテラル	説明	予想される次の呼び出し	その他のコメント
SQLUV_OK	操作が成功した。	sqluvget	DB2 はデータを処理します。
SQLUV_COMM_ERROR	装置との通信エラー	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_INV_ACTION	無効なアクションが要求された。	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_INV_DEV_HANDLE	無効な装置ハンドル	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_INV_BUFF_SIZE	無効なバッファー・サイズが指定された。	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_DEV_ERROR	装置エラー	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_WARNING	警告。これは、DB2 にメディアの終わりを示すためには使用されない (その目的では、SQLUV_ENDOFMEDIA または SQLUV_ENDOFMEDIA_NO_DATA が使用される)。ただし、装置の作動不能条件がこの戻りコードによって示される可能性がある。	sqluvget、または sqluvend、アクション =SQLU_ABORT	
SQLUV_LINK_NOT_EXIST	リンクが現在存在していない。	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_MORE_DATA	操作が成功し、さらにデータが使用可能である。	sqluvget	
SQLUV_ENDOFMEDIA_NO_DATA	メディアが終了し、0 バイトが読み取られた (たとえば、テープの終わり)。	sqluvend	
SQLUV_ENDOFMEDIA	メディアが終了し、> 0 バイトが読み取られた (たとえば、テープの終わり)。	sqluvend	DB2 はデータを処理し、メディア終端条件を処理します。
SQLUV_IO_ERROR	入出力エラー	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
次の呼び出し: [a] 次の呼び出しが sqluvend、action = SQLU_ABORT である場合には、このセッションおよび他のすべてのセッションは終了します。			

sqluvput - 装置へのデータの書き込み

初期設定の後、この関数を使用して装置へデータを書き込むことができます。

権限:

以下のいずれかが必要です。

- *sysadm*
- *dbadm*

必要な接続:

データベース

API 組み込みファイル:

sqluvend.h

C API 構文:

```
/* 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 パラメーター:

pVendorCB
入力。 DATA 構造 (データ・バッファーを含む) および Return_code 用に割り振られたスペースを指すポインター。

Data 出力。書き込まれるデータが入れられたデータ・バッファー。

Return_code
出力。 API 呼び出しからの戻りコード。

obj_num
検索するバックアップ・オブジェクトを指定します。

buff_size
使用するバッファー・サイズを指定します。

actual_buff_size
読み取られる、または書き込まれる実際のバイト数を指定します。この値は、実際に読み取られたデータのバイト数を示すように設定してください。

dataptr
データ・バッファーを指すポインター。

reserve
将来の利用のために予約されています。

使用上の注意:

この関数はバックアップ・ユーティリティーで使用されます。

戻りコード:

表 11. sqluvput についての有効な戻りコードと結果の DB2 アクション

ヘッダー・ファイルのリテラル	説明	予想される次の呼び出し	その他のコメント
SQLUV_OK	操作が成功した。	完了 (たとえば、DB2 にこれ以上データがなくなった) 後、 sqluvput または sqluvend	他のプロセスに操作の成功を通知します。
SQLUV_COMM_ERROR	装置との通信エラー	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_INV_ACTION	無効なアクションが要求された。	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。

表 11. sqluvput についての有効な戻りコードと結果の DB2 アクション (続き)

ヘッダー・ファイルのリテラル	説明	予想される次の呼び出し	その他のコメント
SQLUV_INV_DEV_HANDLE	無効な装置ハンドル	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_INV_BUFF_SIZE	無効なバッファー・サイズが指定された。	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_ENDOFMEDIA	メディアが終了した (たとえば、テープの終了)。	sqluvend	
SQLUV_DATA_RESEND	装置が再びバッファーを送信するよう要求した。	sqluvput	DB2 は最新のバッファーを再度送信します。これは一度だけ行われます。
SQLUV_DEV_ERROR	装置エラー	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_WARNING	警告。これは、DB2 にメディアの終わりを示すためには使用されない (その目的には SQLUV_ENDOFMEDIA が使用される)。ただし、装置の作動不能条件がこの戻りコードによって示される可能性がある。	sqluvput	
SQLUV_LINK_NOT_EXIST	リンクが現在存在していない。	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。
SQLUV_IO_ERROR	入出力エラー	sqluvend、アクション = SQLU_ABORT[a]	セッションは終了します。

次の呼び出し:

[a] 次の呼び出しが sqluvend、action = SQLU_ABORT である場合には、このセッションおよび他のすべてのセッションは終了します。コミット済みセッションは、sqluvint、sqluvdel、 sqluvend の順序の呼び出しで削除されます。

sqluvend - 装置のリンク解除およびリソースの解放

装置を終了またはリンク解除し、関連したリソースをすべて解放します。ベンダーは DB2 へ戻る前に、使用していないリソース (たとえば、割り振られたスペースやファイル・ハンドル) を解放しなければなりません。

権限:

以下のいずれかが必要です。

- *sysadm*
- *dbadm*

必要な接続:

データベース

API 組み込みファイル:

sql.h

C API 構文:

```
/* File: sqluvend.h */
/* API: Unlink the Device and Release its Resources */
/* ... */
int sqluvend (
  sqlint32 action,
```

```
  void * pVendorCB,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API パラメーター:

action 入力。セッションのコミットまたは打ち切りに使用されます。

- SQLUV_COMMIT (0 = コミット)
- SQLUV_ABORT (1 = 打ち切り)

pVendorCB
入力。 Init_output 構造を指すポインター。

Init_output
出力。割り振り解除された Init_output 用のスペース。アクションがコミットであれば、データはバックアップのために保管用のストレージへコミットされています。アクションが打ち切りであれば、データはバックアップのために除去されます。

Return code
出力。 API 呼び出しからの戻りコード。

使用上の注意:

この関数はオープンされたセッションごとに呼び出されます。可能なアクション・コードは、次の 2 つがあります。

- コミット

  このセッションへのデータの出力またはセッションからのデータの読み取りが完了します。

  書き込み (バックアップ) セッションの場合、ベンダーが SQLUV_OK の戻りコードと共に DB2 へ戻ると、 DB2 は、出力データがベンダー製品によって適切に保管されており、後の **sqluvint** 呼び出しで参照されたときにアクセスできるものと判断します。

  読み取り (リストア) セッションで、ベンダーが SQLUV_OK の戻りコードと共に DB2 へ戻った場合、データは再び必要になる可能性があるため、削除すべきではありません。

  ベンダーが SQLUV_COMMIT_FAILED を戻す場合、 DB2 はバックアップ操作またはリストア操作全体に問題があると判断します。すべてのアクティブ・セッションは、アクション = SQLUV_ABORT の **sqluvend** 呼び出しで終了されます。バックアップ操作の場合、コミット済みセッションは **sqluvint**、**sqluvdel**、および **sqluvend** の順序の呼び出しを受信します。

- 打ち切り

  DB2 によって問題が生じているときには、セッションではデータの読み取りまたは書き込みは行われません。

  書き込み (バックアップ) セッションの場合、ベンダーは部分的な出力データ・セットを削除しなければなりません。削除されていれば、SQLUV_OK 戻りコード

を使用します。 DB2 はバックアップ全体に問題があると判断します。すべてのアクティブ・セッションは、action = SQLUV_ABORT の **sqluvend** 呼び出しで終了され、コミット済みセッションは、**sqluvint**、**sqluvdel**、および **sqluvend** の順序の呼び出しを受信します。

読み取り (リストア) セッションの場合、ベンダーはデータを削除してはなりません (再び必要になる可能性があるからです)。ただし、終結処理を行い、SQLUV_OK 戻りコードを出して DB2 に戻ってください。 DB2 は action = SQLUV_ABORT の **sqluvend** 呼び出しですべてのリストア・セッションを終了させます。ベンダーが SQLUV_ABORT_FAILED を DB2 へ戻す場合は、呼び出し側にこのエラーは通知されません。これは DB2 が最初の致命的な障害は戻し、その後に続く障害は無視するためです。この場合、action = SQLUV_ABORT の **sqluvend** を呼び出した DB2 に関して、最初の致命的なエラーが発生しています。

戻りコード:

表 12. sqluvend についての有効な戻りコードと結果の DB2 アクション

ヘッダー・ファイルのリテラル	説明	予想される次の呼び出し	その他のコメント
SQLUV_OK	操作が成功した。	ありません	このセッションに割り振られていたメモリーをすべて解放し、終了します。
SQLUV_COMMIT_FAILED	コミット要求が失敗した。	ありません	このセッションに割り振られていたメモリーをすべて解放し、終了します。
SQLUV_ABORT_FAILED	打ち切り要求が失敗した。	ありません	

sqluvdel - コミット済みセッションの削除

コミット済みセッションを削除します。

権限:

以下のいずれかが必要です。

- *sysadm*
- *dbadm*

必要な接続:

データベース

API 組み込みファイル:

sqluvend.h

C API 構文:

```
/* File: sqluvend.h */
/* API: Delete Committed Session */
/* ... */
int sqluvdel (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API パラメーター:

Init_input
入力。 Init_input および Return_code 用に割り振られたスペース。

Return_code
出力。 API 呼び出しからの戻りコード。 Init_input 構造によって指示されたオブジェクトは削除されます。

使用上の注意:

複数のセッションがオープンしていて、いくつかのセッションはコミットされたが 1 つが失敗したというような場合、この関数が呼び出されて、コミット済みセッションを削除します。シーケンス番号は指定されません。特定のバックアップ操作時に作成されたすべてのオブジェクトを検出して削除するのは、 **sqluvdel** の責任です。 INIT-INPUT 構造の情報が、削除する出力データを識別するために使用されます。ベンダー装置からバックアップ・オブジェクトを削除するのに必要な接続またはセッションを確立するのは、 **sqluvdel** の責任です。この呼び出しからの戻りコードが SQLUV_DELETE_FAILED であれば、DB2 は呼び出し側へ通知しません。DB2 は最初の致命的な障害は戻し、その後に続く障害は無視するという方式なので、このように行います。この場合、**sqluvdel** を呼び出した DB2 に関して、最初の致命的エラーが発生しています。

戻りコード:

表 13. sqluvdel についての有効な戻りコードと結果の DB2 アクション

ヘッダー・ファイルのリテラル	説明	予想される次の呼び出し	その他のコメント
SQLUV_OK	操作が成功した。	ありません	
SQLUV_DELETE_FAILED	削除要求が失敗した。	ありません	

db2VendorQueryApiVersion - 装置でサポートされる API レベルの照会

この関数は、ベンダー・ライブラリーによってサポートされるベンダー API のレベルを判別するために呼び出されます。ベンダー・ライブラリーが DB2 と互換性がない場合、そのベンダー・ライブラリーは使用されません。

ベンダー・ライブラリーに、ログ用にこの API がインプリメントされていない場合、ベンダー・ライブラリーは使用できず、DB2 はエラーをレポートします。このことは、既存のベンダー・ライブラリーを処理しているイメージには影響しません。

権限:

以下のいずれかが必要です。

- *sysadm*
- *dbadm*

必要な接続:

データベース。

API 組み込みファイル:

db2VendorApi.h

C API 構文:

```
void db2VendorQueryApiVersion(db2Uint32 *supportedVersion);
```

API パラメーター:

supportedVersion
出力。ベンダー・ライブラリーでサポートされるベンダー API のバージョンを戻します。

使用上の注意:

この関数は、他のベンダー API の呼び出し前に呼び出されます。

db2VendorGetNextObj - 装置での次のオブジェクトの入手

この関数は、検索条件に一致する次のオブジェクトを入手するよう (sqluvint を使用して) 照会がセットアップされた後に呼び出されます。一度にセットアップできるのは、イメージかログ・ファイルに対する 1 つの検索だけです。

権限:

以下のいずれかが必要です。

- *sysadm*
- *dbadm*

必要な接続:

データベース。

API 組み込みファイル:

db2VendorApi.h

C API 構文:

```
int db2VendorGetNextObj(void *vendorCB,
  struct db2VendorQueryInfo *queryInfo,
  struct Return_code *returnCode);

typedef struct db2VendorQueryInfo
{
   char               db2Instance[SQL_INSTNAME_SZ + 1];
   char               dbname[SQL_DBNAME_SZ + 1];
   char               dbalias[SQL_ALIAS_SZ + 1];
   char               timestamp[SQLU_TIME_STAMP_LEN + 1];
   char               filename[DB2VENDOR_MAX_FILENAME_SZ + 1];
   char               owner[DB2VENDOR_MAX_OWNER_SZ + 1];
   char               mgmtClass[DB2VENDOR_MAX_MGMTCLASS_SZ + 1];
   char               oldestLogfile[DB2_LOGFILE_NAME_LEN + 1];
   db2Uint16          sequenceNum
   SQL_PDB_NODE_TYPE  dbPartitionNum;
   db2Uint32          type;
   db2Uint64          sizeEstimate;
} db2VendorQueryInfo;
```

API パラメーター:

vendorCB
入力。ベンダー・ライブラリーによって割り振られたスペースへのポインター。

queryInfo
出力。ベンダー・ライブラリーによって記入される *db2VendorQueryInfo* 構造へのポインター。

returnCode
出力。 API 呼び出しからの戻りコード。

db2Instance
オブジェクトが属するインスタンスの名前を指定します。

dbname
オブジェクトが属するデータベースの名前を指定します。

dbalias
オブジェクトが属するデータベースの別名を指定します。

timestamp
バックアップ・イメージを識別するのに使用されるタイム・スタンプを指定します。オブジェクトがバックアップ・イメージである場合にのみ有効です。

filename
オブジェクトがロード・コピー・イメージかアーカイブ・ログ・ファイルの場合に、オブジェクトの名前を指定します。

owner オブジェクトの所有者を指定します。

mgmtClass
オブジェクトが保管される管理クラスを指定します (TSM によって使用される)。

oldestLogfile
バックアップ・イメージに保管された一番古いログ・ファイルを指定します。

sequenceNum
バックアップ・イメージのファイル拡張子を指定します。オブジェクトがバックアップである場合にのみ有効です。

dbPartitionNum
オブジェクトが属するデータベース・パーティションの数を指定します。

type オブジェクトがバックアップ・イメージである場合に、イメージ・タイプを指定します。

sizeEstimate
オブジェクトの見積もりサイズを指定します。

使用上の注意:

すべてのフィールドが、各オブジェクトまたは各ベンダーに関係するわけではありません。記入する必要のある必須フィールドは、 db2Instance、 dbname、 dbalias、 timestamp (イメージ用)、 filename (ログおよびロード・コピー・イメージ用)、 owner、 sequenceNum (イメージ用)、および dbPartitionNum です。残りのフィール

ドは、定義する特定のベンダー用に残されます。特定のフィールドが関係しない場合、ストリングの場合には "" に、そして数値タイプの場合には 0 に初期設定する必要があります。

DB2-INFO

この構造には、ベンダー装置に DB2 を識別させる情報が含まれます。

表 14. DB2-INFO 構造のフィールド： フィールドはすべて、NULL 終了ストリングです。

フィールド名	データ・タイプ	説明
DB2_id	char	DB2 製品の ID。指すストリングの最大長は 8 文字です。
version	char	DB2 製品の現行バージョン。指すストリングの最大長は 8 文字です。
release	char	DB2 製品の現行リリース。重要性がなければ NULL に設定します。指すストリングの最大長は 8 文字です。
level	char	DB2 製品の現行レベル。重要性がなければ NULL に設定します。指すストリングの最大長は 8 文字です。
action	char	実行するアクションを指定します。指すストリングの最大長は 1 文字です。
filename	char	バックアップ・イメージの識別に使用されるファイル名。 NULL であれば、 *server_id*、*db2instance*、 *dbname* および *timestamp* によってバックアップ・イメージが固有に識別されます。指すストリングの最大長は 255 文字です。
server_id	char	データベースが存在するサーバーを識別するユニーク名。指すストリングの最大長は 8 文字です。
db2instance	char	db2instance ID。これはコマンドを呼び出すユーザー ID です。指すストリングの最大長は 8 文字です。
type	char	作成するバックアップのタイプ、または実行するリストアのタイプを指定します。以下の値を指定することができます。 アクションが SQLUV_WRITE の場合: 0 - データベースの全バックアップ 3 - 表スペース・レベルのバックアップ アクションが SQLUV_READ の場合: 0 - 全リストア 3 - オンラインの表スペース・リストア 4 - 表スペース・リストア 5 - 履歴ファイルのリストア
dbname	char	バックアップまたはリストアするデータベースの名前。指すストリングの最大長は 8 文字です。
alias	char	バックアップまたはリストアするデータベースの別名。指すストリングの最大長は 8 文字です。
timestamp	char	バックアップ・イメージを識別するのに使用されるタイム・スタンプ。指すストリングの最大長は 26 文字です。

表 14. DB2-INFO 構造のフィールド (続き): フィールドはすべて、NULL 終了ストリングです。

フィールド名	データ・タイプ	説明
sequence	char	バックアップ・イメージのファイル拡張子を指定します。書き込み操作の場合、最初のセッションの値は 1 で、 sqluvint 呼び出しで他のセッションが開始されるたびに、値が 1 ずつ増加します。読み取り操作の場合、値は常にゼロです。指すストリングの最大長は 3 文字です。
obj_list	struct sqlu_gen_list	将来の利用のために予約されています。
max_bytes_per_txn	sqlint32	ユーザーによって指定された転送バッファー・サイズをバイト単位でベンダーに指定します。
image_filename	char	将来の利用のために予約されています。
reserve	void	将来の利用のために予約されています。
nodename	char	バックアップが生成されたノードの名前。
password	char	バックアップが生成されたノードのパスワード。
owner	char	バックアップの開始元の ID。
mcNameP	char	管理クラス。
nodeNum	SQL_PDB_NODE_TYPE	ノード番号。ベンダー・インターフェースでは、255 より大きい番号がサポートされます。

バックアップ・イメージは、*filename* か、または *server_id*、*db2instance*、*type*、*dbname*、および *timestamp* によって固有に識別されます。 *sequence* によって指定されるシーケンス番号はファイル拡張子を識別します。バックアップ・イメージをリストアするときには、同じ値を使用してバックアップ・イメージを検索しなければなりません。ベンダー製品によっては、*filename* が使用された場合に他のパラメーターが NULL に設定されたり、その逆が起こることがあります。

言語構文:

C 構造

```
/* 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

この構造には、装置のベンダーとバージョンを識別するための情報が含まれています。

表 15. VENDOR-INFO 構造のフィールド： フィールドはすべて、NULL 終了ストリングです。

フィールド名	データ・タイプ	説明
vendor_id	char	ベンダーを表す ID。指すストリングの最大長は 64 文字です。
version	char	ベンダー製品の現行バージョン。指すストリングの最大長は 8 文字です。
release	char	ベンダー製品の現行リリース。重要性がなければ NULL に設定します。指すストリングの最大長は 8 文字です。
level	char	ベンダー製品の現行レベルです。重要性がなければ NULL に設定します。指すストリングの最大長は 8 文字です。
server_id	char	データベースが存在するサーバーを識別するユニーク名。指すストリングの最大長は 8 文字です。
max_bytes_per_txn	sqlint32	サポートされる最大の転送バッファー・サイズ。ベンダーが指定します (バイト単位)。これは、ベンダーの初期設定関数からの戻りコードが SQLUV_BUFF_SIZE (無効なバッファー・サイズが指定されたことを示す) である場合にのみ使用されます。
num_objects_in_backup	sqlint32	あるバックアップを完了するために使用されたセッションの数。これは、リストア操作時に、すべてのバックアップ・イメージがいつ処理されたのかを判別するために使用されます。
reserve	void	将来の利用のために予約されています。

言語構文:

C 構造

```
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

この構造には、ベンダー装置との論理リンクを設定し、確立するために DB2 が提供する情報が含まれます。

表 16. INIT-INPUT 構造のフィールド： フィールドはすべて、NULL 終了ストリングです。

フィールド名	データ・タイプ	説明
DB2_session	struct DB2_info	DB2 側から見たセッションの説明。
size_options	unsigned short	オプション・フィールドの長さ。 DB2 バックアップ関数またはリストア関数を使用している場合、このフィールドのデータは *VendorOptionsSize* パラメーターから直接渡されます。
size_HI_order	sqluint32	バイト単位で見積もられた DB サイズの高位 32 ビット。合計サイズは 64 ビットです。
size_LOW_order	sqluint32	バイト単位で見積もられた DB サイズの低位 32 ビット。合計サイズは 64 ビットです。
options	void	この情報は、バックアップまたはリストア関数の呼び出し時にアプリケーションから渡されます。このデータ構造はフラットでなければなりません。つまり、間接のレベルはサポートされません。このデータについてはバイト反転は行われず、コード・ページがチェックされません。 DB2 バックアップ関数、またはリストア関数を使用している場合、このフィールドのデータは *pVendorOptions* パラメーターから直接渡されます。
reserve	void	将来の利用のために予約されています。
prompt_lvl	char	バックアップ操作またはリストア操作を呼び出したときにユーザーが要求したプロンプト・レベル。指すストリングの最大長は 1 文字です。
num_sessions	unsigned short	バックアップ操作またはリストア操作を呼び出したときにユーザーが要求したセッションの数。

言語構文:

C 構造

```
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

この構造には、ベンダー装置が戻した出力が含まれます。

表 17. INIT-OUTPUT 構造のフィールド

フィールド名	データ・タイプ	説明
vendor_session	struct Vendor_info	ベンダーを DB2 に識別させるための情報が含まれます。
pVendorCB	void	ベンダーの制御ブロック。
reserve	void	将来の利用のために予約されています。

言語構文:

C 構造

```
typedef struct Init_output
{
  struct Vendor_info  *vendor_session;
  void                *pVendorCB;
  void                *reserve;
} Init_output;
```

DATA

この構造には、DB2 とベンダー装置との間で転送されるデータが含まれます。

表 18. DATA 構造のフィールド

フィールド名	データ・タイプ	説明
obj_num	sqlint32	バックアップ操作時に DB2 によって割り当てられるシーケンス番号。
buff_size	sqlint32	バッファーのサイズ。
actual_buf_size	sqlint32	送受信された実際のバイト数。これは *buff_size* を超えてはなりません。
dataptr	void	データ・バッファーを指すポインター。 DB2 はこのバッファーにスペースを割り振ります。
reserve	void	将来の利用のために予約されています。

言語構文:

C 構造

```
typedef struct Data
{
  sqlint32  obj_num;
  sqlint32  buff_size;
  sqlint32  actual_buff_size;
  void      *dataptr;
  void      *reserve;
} Data;
```

RETURN-CODE

この構造には、DB2 に戻される戻りコードとエラーの短い説明が含まれます。

表 19. RETURN-CODE 構造のフィールド

フィールド名	データ・タイプ	説明
return_code[a]	sqlint32	ベンダー関数からの戻りコード。
説明	char	戻りコードの短い記述。
reserve	void	将来の利用のために予約されています。
[a] これはベンダー特有の戻りコードで、さまざまな DB2 API によって戻される値とは異なります。ベンダー製品から受け入れられる戻りコードについては、個々の API の説明を参照してください。		

言語構文:

C 構造

```
typedef struct Return_code
{
  sqlint32  return_code,
  char      description[30],
  void      *reserve,
} Return_code;
```

圧縮バックアップ用の API

圧縮プラグイン・インターフェース

DB2 には、COMPR_DB2INFO 構造の定義が用意されています。ベンダーは、以下の構造および API についてその他の定義をそれぞれ用意しています。以下の構造、プロトタイプ、および定数は、DB2 に付属するファイル sqlucompr.h で定義されています。

DB2 環境の記述 - COMPR_DB2INFO:

```
struct COMPR_DB2INFO {
   char    tag[16];
   db2Uint32    version;
   db2Uint32    size;
   char         dbalias[SQLU_ALIAS_SZ+1];
   char         instance[SQL_INSTNAME_SZ+1];
   SQL_PDB_NODE_TYPE node;
   SQL_PDB_NODE_TYPE catnode;
   char         timestamp[SQLU_TIME_STAMP_LEN+1];
   db2Uint32    bufferSize;
   db2Uint32    options;
   db2Uint32    bkOptions;

   db2Uint32    db2Version;
   db2Uint32    platform;
   db2int32     comprOptionsByteOrder;
   db2Uint32    comprOptionsSize;
   void        *comprOptions;
   db2Uint32    savedBlockSize;
   void        *savedBlock;
};
```

COMPR_DB2INFO

DB2 は、この構造を割り振って定義し、InitCompression および InitDecompression API に対して、パラメーターとして渡します。この構造は、バックアップまたはリストアされるデータベースについて記述し、操作が行われる DB2 環境についての詳細を示しています。構造のフィールドは、以下のとおりです。

tag[16]
構造の目印として使用。これは、必ず ″COMPR_DB2INFO \0″ ストリングに設定されます。

version
使用される構造のバージョンを示します。これにより、 API は追加のフィールドが存在することを認識できます。現在のバージョンは 1 です。将来、この構造にさらにフィールドが追加される可能性があります。

size COMPR_DB2INFO 構造のサイズを指定します (バイト単位)。

dbalias[SQLU_ALIAS_SZ+1]

instance[SQL_INSTNAME_SZ+1]

node

catnode

timestamp[SQLU_TIME_STAMP_LEN+1]
バックアップまたはリストアされるデータベースについて記述します。これらは、バックアップ・イメージに命名する際に使用されるフィールドです。リストア操作の場合、dbalias はソース・データベースの別名を示します。

bufferSize
転送バッファーのサイズを指定します (4 K ページ単位)。

options
db2Backup API または db2Restore API で指定される iOptions フィールド。

bkOptions
リストア操作の場合、バックアップが作成されたときに db2Backup API で使用された iOptions フィールドを指定します。バックアップ操作の場合、ゼロに設定されます。

db2Version
DB2 エンジンのバージョンを指定します。

platform
DB2 エンジンが実行されているプラットフォームを指定します。この値は、<sqlmon.h> にリストされているいずれかの値になります。

comprOptionsByteOrder
API を実行するクライアントで使用するバイト・オーダーを指定します。 DB2 は、*comprOptions* として渡されたデータの解釈または変換を行わないため、このフィールドを使用して、データの使用前

にデータのバイトを反転する必要があるかどうかを判別しなければなりません。プラグイン・ライブラリー自体によって、何らかの変換を実行する必要があります。

comprOptionsSize
db2Backup および db2Restore API の *piComprOptionsSize* パラメーターの値を指定します。

***comprOptions**
db2Backup および db2Restore API の *piComprOptions* フィールドの値を指定します。

savedBlockSize

***savedBlock**
DB2 では、プラグイン・ライブラリーによって、任意のデータ・ブロックをバックアップ・イメージに保管できます。そのようなデータ・ブロックが特定のバックアップで保管された場合、リストア操作時にこれらのフィールドに戻されます。バックアップ操作の場合、これらのフィールドはゼロに設定されます。

プラグインの記述 - COMPR_PIINFO:

```
struct COMPR_PIINFO {
   char    tag[16];
   db2Uint32    version;
   db2Uint32    size;
   db2Uint32    useCRC;
   db2Uint32    useGran;
   db2Uint32    useAllBlocks;
   db2Uint32    savedBlockSize;
};
```

COMPR_PIINFO
この構造は、DB2 に対して記述するために、プラグイン・ライブラリーによって使用されます。この構造は、DB2 によって割り振られて初期設定され、主なフィールドは、 InitCompression 呼び出し時に、プラグイン・ライブラリーによって記入されます。

tag[16]
構造の目印として使用。 (DB2 側で設定します。) これは、必ず ″COMPR_PIINFO \0″ ストリングに設定されます。

version
使用される構造のバージョンを示します。これにより、 API は追加のフィールドが存在することを認識できます。現在のバージョンは 1 です。 (DB2 側で設定します。) 将来、この構造にさらにフィールドが追加される可能性があります。

size COMPR_PIINFO 構造のサイズを示します (バイト単位)。 (DB2 側で設定します。)

useCRC
DB2 では、圧縮プラグインは、32 ビット CRC またはチェックサム値を使用して、圧縮および解凍されるデータの保全性を検査でき

ます。ライブラリーがそのような検査を使用する場合、このフィールドは 1 に設定されます。それ以外の場合、フィールドは 0 に設定されます。

useGran
圧縮ルーチンでデータを任意のサイズ単位で圧縮できる場合、ライブラリーは、このフィールドを 1 に設定します。圧縮圧縮ルーチンが、データをバイト・サイズ単位でのみ圧縮する場合、ライブラリーは、このフィールドを 0 に設定します。この指標の設定に関係する詳細は、Compress の useGran パラメーターの説明を参照してください。リストア操作の場合、このフィールドは無視されます。

useAllBlocks
DB2 が、圧縮されていない元のブロックよりも大きい圧縮済みデータ・ブロックをバックアップするかどうかを指定します。デフォルトでは、DB2 は、圧縮されたバージョンのほうが大きい場合はデータを圧縮しないで保管しますが、特定の環境では、プラグイン・ライブラリーは、どの場合でも圧縮されたデータ・バックアップを保管します。 DB2 が、圧縮済みバージョンの全データ・ブロックを保管する予定である場合、ライブラリーは、この値を 1 に設定します。 DB2 が、元のデータよりも小さい場合にだけ、圧縮済みバージョンのデータを保管するのであれば、ライブラリーは、この値を 0 に設定します。リストア操作の場合、このフィールドは無視されます。

savedBlockSize
DB2 では、プラグイン・ライブラリーによって、任意のデータ・ブロックをバックアップ・イメージに保管できます。そのようなデータ・ブロックを特定のバックアップで保管するのであれば、ライブラリーは、このフィールドを、このデータに割り振られたブロックのサイズに設定します。 (実際のデータは、後続の API 呼び出しで DB2 に渡されます。) データを保管しない場合、プラグイン・ライブラリーは、このフィールドをゼロに設定します。リストア操作の場合、このフィールドは無視されます。

制御ブロックの記述 - COMPR_CB:

```
struct COMPR_CB;

extern "C" {

int InitCompression(
   const COMPR_DB2INFO    *db2Info,
   COMPR_PIINFO           *piInfo,
      COMPR_CB  **pCB);

int GetSavedBlock(
   COMPR_CB               *pCB,
   db2Uint32               blockSize,
      void       *data);

int Compress(
      COMPR_CB               *pCB,
      const char             *src,
      db2int32                srcLen,
         db2Uint32   srcGran,
      char                   *tgt,
```

```
      db2int32               tgtSize,
      db2int32              *srcAct,
      db2int32              *tgtAct,
         db2Uint32  *tgtCRC);

int GetMaxCompressedSize(
      COMPR_CB              *pCB,
         db2Uint32   srcLen);

int TermCompression(
         COMPR_CB   *pCB);

int InitDecompression(
      const COMPR_DB2INFO    *db2Info,
         COMPR_CB  **pCB);

   int Decompress(
      COMPR_CB              *pCB,
      const char            *src,
      db2int32               srcLen,
      char                  *tgt,
      db2int32               tgtSize,
      db2int32              *tgtAct,
         db2Uint32  *tgtCRC);

   int TermDecompression(
         COMPR_CB   *pCB);
   }
```

COMPR_CB

これは、プラグイン・ライブラリーによって内部的に使用される構造です。ここには、圧縮および解凍ルーチンによって内部的に使用されるデータが含まれます。 DB2 は、この構造をプラグイン・ライブラリーに対して行うそれぞれの呼び出しに渡しますが、構造の性質はすべて、構造のフィールドの定義や構造のメモリー管理を含め、ライブラリーに残されます。

```
int InitCompression(
      const COMPR_DB2INFO    *db2Info,
      COMPR_PIINFO           *piInfo,
      COMPR_CB              **pCB);
```

圧縮ライブラリーを初期設定します。 DB2 は、db2Info および piInfo 構造を渡します。ライブラリーは、piInfo の適切なフィールドを記入し、pCB を割り当てて、割り振られたメモリーへのポインターを戻します。

```
int GetSavedBlock(
      COMPR_CB              *pCB,
      db2Uint32              blockSize,
      void                  *data);
```

バックアップ・イメージに保管するベンダー固有のデータ・ブロックを入手します。ライブラリーが、piInfo->savedBlockSize にゼロ以外の値を戻した場合、DB2 は、その値を *blockSize* として使用して、GetSavedBlock を呼び出します。プラグイン・ライブラリーは、指定されたサイズのデータを、データによって参照されるメモリーに書き込みます。この機能は、バックアップの場合にのみ、BM1 での初期データ処理中に呼び出されます。 db2Backup API で > 1 の並列処理が指定される場合でも、この機能は、バックアップごとに一回だけ呼び出されます。

```
int Compress(
      COMPR_CB              *pCB,
      const char            *src,
      db2int32               srcLen,
      db2int32               srcGran,
      char                  *tgt,
      db2int32               tgtSize,
      db2int32              *srcAct,
      db2int32              *tgtAct,
      db2Uint32             *tgtCRC);
```

データ・ブロックを圧縮します。 *src* は、サイズが *srcLen* バイトのデータ・ブロックを示します。 *tgt* は、サイズが *tgtSize* バイトのバッファーを示します。プラグイン・ライブラリーは、アドレス *src* でデータを圧縮し、アドレス *tgt* で圧縮済みデータをバッファーに書き込みます。圧縮したデータの中で、圧縮されていないデータの実際量が、*srcAct* に保管されます。圧縮済みデータの実サイズは、*tgtAct* として戻されます。

ライブラリーが piInfo->useCRC に 1 の値を戻した場合、圧縮されていないブロックの CRC 値は *tgtCRC* として戻されます。ライブラリーが piInfo->useCRC に 0 の値を戻した場合、 *tgtCRC* は NULL ポインターになります。

ライブラリーが piInfo->useGran に 1 の値を戻した場合、 *srcGran* は、データのページ・サイズの log2 を指定します。 (たとえば、データのページ・サイズが 4096 バイトである場合、*srcGran* は 12 になります。) ライブラリーは、実際の圧縮されたデータの量 (*srcAct*) が、正確にこのページ・サイズの倍数になるようにします。ライブラリーが useGran フラグを設定する場合、DB2 では、圧縮済みデータをバックアップ・イメージに合わせるため、より効率的なアルゴリズムを使用することができます。そのようにすると、プラグインのパフォーマンスが改善されると同時に、圧縮済みバックアップ・イメージが小さくなります。ライブラリーが piInfo->srcGran に 0 の値を戻した場合、細分度は 1 バイトです。

```
int GetMaxCompressedSize(
      COMPR_CB              *pCB,
      db2Uint32              srcLen,
      db2Uint32             *tgtLen);
```

データ・ブロックを圧縮するのに必要な最大可能バッファー・サイズを見積もります。 *srcLen* は、圧縮する予定のデータ・ブロックのサイズを示します。ライブラリーは、*tgtLen* として圧縮した後で、理論上の最大バッファー・サイズを戻します。

DB2 では、*tgtLen* で戻された値を使用して、メモリー使用状況を内部的に最適化します。値を計算しない、または誤った値を計算する場合、DB2 は 1 つのデータ・ブロックで圧縮 API を複数回呼び出さなければならないか、ユーティリティー・ヒープのメモリーを浪費する結果になります。バックアップは、戻された値に関係なく正しく作成されます。

```
int TermCompression(
      COMPR_CB   *pCB);
```

圧縮ライブラリーを終了します。ライブラリーは、pCB に使用したメモリーを解放します。

```
int InitDecompression(
      const COMPR_DB2INFO    *db2Info,
      COMPR_CB              **pCB);
```

解凍ライブラリーを初期設定します。 DB2 は、db2Info 構造を渡します。ライブラリーは、*pCB* を割り振り、割り振られたメモリーへのポインターを戻します。

```
int Decompress(
      COMPR_CB              *pCB,
      const char            *src,
      db2int32               srcLen,
      char                  *tgt,
      db2int32               tgtSize,
      db2int32              *tgtLen,
      db2Uint32             *tgtCRC);
```

データ・ブロックを解凍します。 *src* は、サイズが *srcLen* バイトのデータ・ブロックを示します。 *tgt* は、サイズが *tgtSize* バイトのバッファーを示します。プラグイン・ライブラリーは、アドレス *src* でデータを解凍し、アドレス *tgt* で解凍済みデータをバッファーに書き込みます。解凍済みデータの実サイズは、*tgtLen* として戻されます。ライブラリーが piInfo->useCRC に 1 の値を戻した場合、圧縮されていないブロックの CRC は *tgtCRC* として戻されます。ライブラリーが piInfo->useCRC に 0 の値を戻した場合、 *tgtLen* は NULL ポインターになります。

```
int TermDecompression(
      COMPR_CB   *pCB);
```

解凍ライブラリーを終了します。ライブラリーは、*pCB* に使用したメモリーを解放します。これらの API によって内部的に使用されたメモリーはすべて、ベンダーによって管理されます。プラグイン・ライブラリーは、COMPR_CB 構造によって使用されるメモリーを管理します。 DB2 は、データ・バッファー (API の *src* および *tgt* パラメーター) に使用したメモリーを管理します。

プラグイン・インターフェース戻りコード:

これらは、API によって戻される可能性のある戻りコードです。指定されている箇所を除き、DB2 は、ゼロ以外の戻りコードが戻される場合に、バックアップまたはリストアを終了します。

SQLUV_OK	0	操作は成功しました。
SQLUV_BUFFER_TOO_SMALL	100	ターゲット・バッファーが小さすぎます。バックアップ時に示される場合、tgtAct フィールドには、オブジェクトを圧縮するのに必要な見積もりサイズが示されます。 DB2 は、少なくとも指定された大きさのバッファーで操作を再試行します。リストア時に示される場合、操作は失敗します。

SQLUV_PARTIAL_BUFFER	101	バッファーは部分的に圧縮されました。バックアップ時には、srcAct フィールドには、実際に圧縮されたデータの実際の量が示され、 tgtAct フィールドには、圧縮されたデータの実際のサイズが示されます。リストア時に示される場合、操作は失敗します。
SQLUV_NO_MEMORY	102	メモリー不足
SQLUV_EXCEPTION	103	コードでシグナルまたは例外が生じました。
SQLUV_INTERNAL_ERROR	104	内部エラーが検出されました。

SQLUV_BUFFER_TOO_SMALL と SQLUV_PARTIAL_BUFFER の違いは、SQLUV_PARTIAL_BUFFER が戻されると、 DB2 は出力バッファー内のデータを有効であると見なす点です。

関連資料:

- 83 ページの『db2Backup - データベースのバックアップ』
- 113 ページの『db2Restore - データベースのリストア』

付録 I. DB2 Universal Database 技術情報

DB2 資料とヘルプ

DB2® 技術情報は、以下のツールと方法を介して利用できます。

- DB2 インフォメーション・センター
  - トピック
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- ダウンロード可能な PDF ファイル、CD 上の PDF ファイル、および印刷された資料
  - ガイド
  - リファレンス・マニュアル
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ
  - SQL 状態ヘルプ
- インストール済みソース・コード
  - サンプル・プログラム

ibm.com® にある技術資料、白書、 Redbooks™ その他の DB2 Universal Database™ 技術情報にオンラインでアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (www.ibm.com/software/data/pubs/) にアクセスしてください。

DB2 資料の更新

IBM® は、 DB2 インフォメーション・センターの資料のフィックスパックやその他の資料更新を定期的に発行しています。 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) にアクセスすれば、常に最新の情報が掲載されます。 DB2 インフォメーション・センターをローカル・インストールしている場合、更新記事を表示するには、まず手動で更新をインストールしてください。新しい情報が発表されたときに資料を更新することにより、 *DB2 インフォメーション・センター CD* からインストールした情報を更新することができます。

インフォメーション・センターの方が、PDF 資料やハードコピー資料よりも頻繁に更新されます。 DB2 の最新の技術情報を入手するには、資料更新が発行されたときにそれをインストールするか、または www.ibm.com サイトの DB2 インフォメーション・センターにアクセスしてください。

関連概念:

- 「*コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻*」の『CLI サンプル・プログラム』

- 「*アプリケーション開発ガイド アプリケーションの構築および実行*」の『Java サンプル・プログラム』
- 416 ページの『DB2 インフォメーション・センター』

関連タスク:

- 437 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 427 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 438 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 438 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 439 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 429 ページの『DB2 PDF 資料および印刷された資料』

DB2 インフォメーション・センター

DB2® インフォメーション・センターを使用すると、 DB2 Universal Database™、 DB2 Connect™、 DB2 Information Integrator および DB2 Query Patroller™ などの DB2 ファミリー製品を最大限に活用するのに必要なすべての情報にアクセスできます。また、DB2 インフォメーション・センターは、 DB2 の主な機能とコンポーネントに関する情報を提供します (レプリケーション、データウェアハウジング、および DB2 の種々の Extender など)。

Mozilla 1.0 以上または Microsoft® Internet Explorer 5.5 以上で表示する場合、 DB2 インフォメーション・センターには以下の機能があります。以下のいくつかの機能では、 JavaScript™ のサポートを使用可能にする必要があります:

柔軟なインストール・オプション
以下の中から、 ご使用の環境に最も適したオプションを使って DB2 資料を表示できます。

- 最新の資料を常に自動的に利用できるようにするには、 IBM® の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターからすべての資料に直接アクセスします。
- 更新処理を最小化し、イントラネット内のネットワーク・トラフィックだけに制限するには、イントラネット上の 1 つのサーバーに DB2 資料をインストールします。
- 柔軟性を改善し、ネットワーク接続への依存を軽減するには、個々のコンピューターに DB2 資料をインストールします。

検索 「**検索**」テキスト・フィールドに検索語を入力することにより、 DB2 インフォメーション・センターのすべてのトピックを検索できます。複数の語句を引用符で囲めば、完全一致を検索できます。また、ワイルドカード演算子 (*、?) とブール演算子 (AND、 NOT、OR) を使用して検索を絞り込むことができます。

タスク指向の目次
単一の目次の中から、DB2 資料のトピックを見付けることができます。目

次は、主に実行するタスクの種類に従って編成されていますが、そのほかに製品概要、特定のゴール (目的) の情報、参照情報、索引、および用語集も含まれます。

- 製品概要では、DB2 ファミリーで使用可能な製品間の関係、そうした各製品で提供される機能、および各製品の最新リリース情報について説明されています。
- インストール、管理および開発などのゴール・カテゴリーには、タスクを迅速に完了し、そのための背景情報をよく理解できるようにするトピックが含まれています。
- 「参照」トピックでは、その対象に関する詳細な情報 (ステートメントとコマンドの構文、メッセージ・ヘルプ、構成パラメーターなど) が説明されています。

現在のトピックを目次に表示する

現在のトピックが目次のどの部分に該当するかを表示するには、目次フレーム内の**「リフレッシュ/現在のトピックの表示 (Refresh/Show Current Topic)」**ボタンをクリックするか、コンテンツ・フレーム内の**「目次に表示 (Show in Table of Contents)」**ボタンをクリックします。幾つかのファイルで関連トピックへの複数のリンクをたどった場合、または検索結果からトピックにアクセスした場合には、この機能が役立ちます。

索引 索引から、すべての資料にアクセスすることができます。索引では、用語が 50 音順に編成されています。

用語集 用語集を見れば、DB2 資料で使われているさまざまな用語の定義を調べることができます。用語集では、用語が 50 音順に編成されています。

組み込まれているローカライズ情報

DB2 インフォメーション・センターは、ブラウザーで設定された言語でトピックを表示します。設定された言語のトピックが利用できない場合、DB2 インフォメーション・センターにはそのトピックの英語版が表示されます。

iSeries™ 技術情報については、 IBM eServer™ iSeries Information Center (www.ibm.com/eserver/iseries/infocenter/) を参照してください。

関連概念:

- 418 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 427 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 428 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 426 ページの『DB2 インフォメーション・センターの呼び出し』
- 420 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 423 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 インフォメーション・センターのインストール・シナリオ

さまざまに異なる業務環境のもとでは、 DB2® 情報にどのようにアクセスするかの要件もそれぞれ異なります。 DB2 インフォメーション・センターにアクセスするには、 IBM® の Web サイト、サーバーまたは組織のネットワーク、あるいはコンピューターへのインストールという 3 つの方法が可能です。この 3 つのケースのいずれも、資料は DB2 インフォメーション・センター内に置かれます。インフォメーション・センターは、ブラウザーを使って表示できるように設計されたトピック・ベースの情報の Web サイトです。デフォルトでは、DB2 製品から、 IBM Web サイト上の DB2 インフォメーション・センターにアクセスします。これに対して、 イントラネット・サーバーまたはご自分のコンピューターから DB2 インフォメーション・センターにアクセスしたい場合、製品メディア・パック内にある DB2 インフォメーション・センター CD から DB2 インフォメーション・センターをインストールする必要があります。以下では、DB2 資料へのアクセス・オプションの要約、および 3 つのインストール・シナリオを示します。これを参考にして、お客様の業務環境で DB2 インフォメーション・センターにアクセスするにはどの方法が最適か、どのようなインストール上の問題に配慮する必要があるかを判別してください。

DB2 資料にアクセスするオプションの要約:

以下の表は、お客様の実際の業務環境で、 DB2 インフォメーション・センターの DB2 製品情報にアクセスする方法としてどんなオプションが推奨されるかを示します。

インターネット・アクセス	イントラネット・アクセス	推奨されるアクション
はい	はい	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス、またはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターへのアクセス
はい	いいえ	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス
いいえ	はい	イントラネット・サーバーにインストール済みの DB2 インフォメーション・センターへのアクセス
いいえ	いいえ	ローカル・コンピューター上の DB2 インフォメーション・センターへのアクセス

シナリオ: コンピューター上の DB2 インフォメーション・センターへのアクセス:

Tsu-Chen 氏は小さな町で工場を経営していますが、その町には、インターネット・アクセスを提供する地元のインターネット・サービス・プロバイダーがありません。彼は、在庫、製品オーダー、銀行口座情報、および営業経費を管理するために DB2 Universal Database™ を購入しました。 Tsu-Chen 氏は以前に DB2 製品を利用したことがないので、 DB2 の使用方法を習得するために、DB2 製品資料を参照する必要があります。

Tsu-Chen 氏は 標準インストール・オプションを使って DB2 Universal Database を自分のコンピューターにインストールした後、 DB2 資料にアクセスしようとします。しかし、開こうとしているページが見つからないというエラー・メッセージがブラウザーから通知されました。 Tsu-Chen 氏は DB2 製品のインストール・マニュアルを調べた結果、 DB2 資料を自分のコンピューター上で利用するには、 DB2 インフォメーション・センターをインストールしなければならないことに気付きます。そしてメディア・パックの中にあった *DB2 インフォメーション・センター CD* を見つけ出して、インストールしました。

これで、Tsu-Chen 氏はオペレーティング・システムのアプリケーション・ランチャーから DB2 インフォメーション・センターにアクセスできるようになり、より良い業務成果をあげるために DB2 製品を利用する方法を習得できます。

シナリオ: IBM Web サイト上の DB2 インフォメーション・センターへのアクセス:

Colin は、あるセミナー企業に所属する情報技術コンサルタントです。彼の専門はデータベース・テクノロジーおよび SQL で、 DB2 Universal Database を使って北米一帯の企業を対象にこれらの科目のセミナーを開催しています。 Colin のセミナーでは、教材として DB2 資料も使用されます。たとえば、SQL の講習コースでは、データベース照会の基本構文と拡張構文を教えるために SQL に関する DB2 資料が使用されます。

Colin が教えている企業の大半はインターネット・アクセスを配備しています。このような状況から判断して、Colin は、最新バージョンの DB2 Universal Database を自分のモバイル・コンピューターにインストールしたとき、 IBM Web サイト上の DB2 インフォメーション・センターにアクセスするよう構成しました。この構成によって、Colin はセミナーで教えるときに最新の DB2 資料にオンライン・アクセスすることができます。

しかし、時折、Colin は移動中にインターネット・アクセスを利用できないことがあります。これは問題となります。担任するセミナーの準備のために DB2 資料にアクセスする必要のある場合には、とくにそうです。このような事態が起きないようにするために、 Colin は自分のモバイル・コンピューターに DB2 インフォメーション・センターのコピーをインストールしました。

こうして、Colin は常に DB2 資料のコピーを自在に活用できるようになりました。**db2set** コマンドを使って自分のモバイル・コンピューターのレジストリー変数を簡単に構成し、どこにいるかに応じて、IBM Web サイトまたは自分のモバイル・コンピューターから DB2 インフォメーション・センターにアクセスできます。

シナリオ: イントラネット・サーバー上の DB2 インフォメーション・センターへのアクセス:

Eva は、生命保険会社のデータベース上級管理者です。彼女は管理業務の一環として、会社の UNIX® データベース・サーバーに最新バージョンの DB2 Universal Database をインストールおよび構成します。彼女の会社は最近、セキュリティー上の理由から、インターネット・アクセスをもはや業務で利用できないようにすると社員に通知しました。同社はネットワーク環境を装備しているため、 Eva は DB2 インフォメーション・センターのコピーをイントラネット・サーバー上にインスト

ールして、社内のデータウェアハウスを定期的に利用するすべての社員 (営業担当者、営業部長、および業務分析担当者) から DB2 資料へのアクセスを可能にすることにしました。

Eva は、応答ファイルを使って全社員のコンピューター上に最新バージョンの DB2 Universal Database をインストールするようデータベース・チームに指示します。その際、イントラネット・サーバーのホスト名とポート番号を使って DB2 インフォメーション・センターにアクセスできるよう、確実に各コンピューターを構成します。

しかし、Eva のチームの下級データベース管理者である Migual の誤解によって、数人の社員のコンピューター上で、イントラネット・サーバーの DB2 インフォメーション・センターにアクセスするよう DB2 Universal Database を構成する代わりに、 DB2 インフォメーション・センターのコピーをそれらのコンピューターにインストールしてしまいました。これを訂正するために、Eva は、 **db2set** コマンドを使ってこれらのコンピューター上の DB2 インフォメーション・センターのレジストリー変数 (ホスト名は DB2_DOCHOST、ポート番号は DB2_DOCPORT) を変更するよう Migual に指示しました。これで、ネットワーク上の適切なすべてのコンピューターが DB2 インフォメーション・センターにアクセスできるようになり、社員は DB2 に関する質問の答えを DB2 資料から見つけることができます。

関連概念:

- 416 ページの『DB2 インフォメーション・センター』

関連タスク:

- 427 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 420 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 423 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

関連資料:

- 「コマンド・リファレンス」の『db2set - DB2 プロファイル・レジストリー・コマンド』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)

DB2 製品資料にアクセスする方法として、 IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、 *DB2 インフォメーション・センター CD* から資料をインストールする必要があります。 DB2 セットアップ・ウィザードを使用すれば、インストール設定を定義し、 UNIX オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、UNIX コンピューターに DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

- **ハードウェア要件**

  以下のいずれかのプロセッサーが必要です。

  - PowerPC (AIX)
  - HP 9000 (HP-UX)
  - Intel 32 ビット (Linux)
  - Solaris UltraSPARC コンピューター (Solaris オペレーティング環境)

- **オペレーティング・システム要件**

  以下のいずれかのオペレーティング・システムが必要です。

  - IBM AIX 5.1 (PowerPC 上)
  - HP-UX 11i (HP 9000 上)
  - Red Hat Linux 8.0 (Intel 32 ビット上)
  - SuSE Linux 8.1 (Intel 32 ビット上)
  - Sun Solaris バージョン 8 (Solaris オペレーティング環境の UltraSPARC コンピューター上)

  注: DB2 インフォメーション・センターは、 DB2 クライアントをサポートする UNIX オペレーティング・システム上で稼動します。このため、IBM Web サイトから DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

- **ソフトウェア要件**
  - 以下のブラウザーがサポートされています。
    - Mozilla バージョン 1.0 以上
- DB2 セットアップ・ウィザードは、グラフィック・インストーラーです。ご使用のマシンで DB2 セットアップ・ウィザードのグラフィカル・ユーザー・インターフェースを表示可能にする X Window システム・ソフトウェアをインプリメントする必要があります。 DB2 セットアップ・ウィザードを実行する前に、ディスプレイを正しくエクスポートしたことを確認してください。たとえば、コマンド・プロンプトで

  ```
  export DISPLAY=9.26.163.144:0.
  ```

  というコマンドを入力します。

- **通信要件**
  - TCP/IP

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようにします。

1. システムにログオンします。

2. DB2 インフォメーション・センター製品 CD を挿入してシステムにマウントします。
3. 次のコマンドを入力して、CD がマウントされているディレクトリーに移動します。

   ```
   cd /cd
   ```

   /cd は、CD のマウント・ポイントを表します。
4. **./db2setup** コマンドを入力して、 DB2 セットアップ・ウィザードを開始します。
5. IBM DB2 セットアップ・ランチパッドが開きます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、**「ヘルプ」**をクリックします。**「キャンセル」**をクリックすれば、いつでもインストールを終了できます。
6. **「インストールしたい製品を選択します」**ページでは、**「次へ」**をクリックします。
7. **「DB2 セットアップ・ウィザードにようこそ (Welcome to the DB2 Setup wizard)」**ページで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
8. インストールを続行するには、使用許諾条件に同意する必要があります。**「ご使用条件」**ページで、**「ご使用条件に同意します (I accept the terms in the license agreement)」** をクリックして、**「次へ」**をクリックします。
9. **「インストール・アクションの選択」**で、**「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」**を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、**「設定を応答ファイルに保管する」**を選択します。**「次へ」**をクリックします。
10. **「インストールする言語の選択」**ページでは、 DB2 インフォメーション・センターをインストールする言語を選択します。**「次へ」**をクリックします。
11. **「DB2 インフォメーション・センター・ポートの指定」**ページでは、 DB2 インフォメーション・センターへの着信通信を構成します。**「次へ」**をクリックしてインストールを続けます。
12. **「ファイルのコピーの開始」**ページでは、インストールの選択項目を確認します。設定を変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

このほか、応答ファイルを使って DB2 インフォメーション・センターをインストールすることもできます。

インストール・ログ db2setup.his、 db2setup.log、および db2setup.err は、デフォルトでは /tmp ディレクトリーに置かれます。

db2setup.log ファイルは、エラーも含めた DB2 製品のインストール情報をすべてキャプチャーします。 db2setup.his ファイルは、コンピューター上の DB2 製品

インストール内容をすべて記録します。 DB2 は、db2setup.log ファイルを db2setup.his に付加します。 db2setup.err ファイルは、 Java から戻されるすべてのエラー出力 (例外やトラップの情報など) をキャプチャーします。

インストールが完了したら、ご使用の UNIX オペレーティング・システムに応じて、 DB2 は以下のいずれかのディレクトリーにインストールされます。

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris オペレーティング環境: /opt/IBM/db2/V8.1

関連概念:

- 416 ページの『DB2 インフォメーション・センター』
- 418 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 のインストール (UNIX)』
- 427 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 428 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 426 ページの『DB2 インフォメーション・センターの呼び出し』
- 423 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)

DB2 製品資料にアクセスする方法として、 IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、 *DB2 インフォメーション・センター CD* から DB2 資料をインストールする必要があります。 DB2 セットアップ・ウィザードを使用すれば、インストール設定を定義し、 Windows オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、Windows に DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

- **ハードウェア要件**

  以下のいずれかのプロセッサーが必要です。

  – 32 ビット・コンピューター: Pentium または Pentium 互換の CPU

- **オペレーティング・システム要件**

  以下のいずれかのオペレーティング・システムが必要です。

  – Windows 2000
  – Windows XP

  注: DB2 インフォメーション・センターは、 DB2 クライアントをサポートする Windows オペレーティング・システム上で稼動します。このため、IBM Web サイトの DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

- **ソフトウェア要件**
  – 以下のブラウザーがサポートされています。
    - Mozilla 1.0 以上
    - Internet Explorer バージョン 5.5 または 6.0 (Windows XP の場合はバージョン 6.0)
- **通信要件**
  – TCP/IP

制約事項:

- DB2 インフォメーション・センターをインストールするには、管理権限をもつアカウントが必要です。

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようにします。

1. DB2 インフォメーション・センターのインストールで定義したアカウントで、システムにログオンします。
2. CD をドライブに挿入します。自動実行機能が使用可能になっていれば、IBM DB2 セットアップ・ランチパッドが起動します。
3. DB2 セットアップ・ウィザードは、システム言語を判別して、その言語用のセットアップ・プログラムを立ち上げます。英語以外の言語でセットアップ・プログラムを実行したい場合、またはセットアップ・プログラムの自動始動が失敗した場合には、 DB2 セットアップ・ウィザードを手動で開始できます。

   次のようにして、DB2 セットアップ・ウィザードを手動で開始します。

   a. **「スタート」**をクリックし、**「ファイル名を指定して実行」**を選択します。
   b. **「開く」**フィールドで、以下のコマンドを入力します。

      ```
      x:¥setup.exe /i 2-letter language identifier
      ```

      ここで、*x:* は CD ドライブ、 *2-letter language identifier* (2 文字の言語識別子) はセットアップ・プログラムを実行する言語を表します。
   c. **「OK」**をクリックします。
4. IBM DB2 セットアップ・ランチパッドが開きます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用

できます。オンライン・ヘルプを呼び出すには、**「ヘルプ」**をクリックします。**「キャンセル」**をクリックすれば、いつでもインストールを終了できます。

5. **「インストールしたい製品を選択します」**ページでは、**「次へ」**をクリックします。
6. **「DB2 セットアップ・ウィザードにようこそ (Welcome to the DB2 Setup wizard)」**ページで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
7. インストールを続行するには、使用許諾条件に同意する必要があります。**「ご使用条件」**ページで、**「ご使用条件に同意します (I accept the terms in the license agreement)」** をクリックして、**「次へ」**をクリックします。
8. **「インストール・アクションの選択」**で、**「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」**を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、**「設定を応答ファイルに保管する」**を選択します。**「次へ」**をクリックします。
9. **「インストールする言語の選択」**ページでは、 DB2 インフォメーション・センターをインストールする言語を選択します。**「次へ」**をクリックします。
10. **「DB2 インフォメーション・センター・ポートの指定」**ページでは、 DB2 インフォメーション・センターへの着信通信を構成します。**「次へ」**をクリックしてインストールを続けます。
11. **「ファイルのコピーの開始」**ページでは、インストールの選択項目を確認します。設定を変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

応答ファイルを使って DB2 インフォメーション・センターをインストールすることができます。また、**db2rspgn** コマンドを使って、既存のインストール内容に基づく応答ファイルを生成することもできます。

インストール時に検出されるエラーの詳細については、 「マイ ドキュメント」¥DB2LOG¥ ディレクトリー内の db2.log ファイルと db2wi.log ファイルを参照してください。 「マイ ドキュメント」 ディレクトリーの場所は、ご使用のコンピューターの設定によって異なります。

db2wi.log ファイルは、 DB2 の最新のインストール情報をキャプチャーします。db2.log は、DB2 製品のインストールの履歴をキャプチャーします。

関連概念:

- 416 ページの『DB2 インフォメーション・センター』
- 418 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 製品のインストール (Windows)』

- 427 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 428 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 426 ページの『DB2 インフォメーション・センターの呼び出し』
- 420 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』

関連資料:

- 「コマンド・リファレンス」の『db2rspgn - 応答ファイル生成プログラム・コマンド』

DB2 インフォメーション・センターの呼び出し

DB2 インフォメーション・センターは、Linux、UNIX、および Windows オペレーティング・システム用の DB2 製品 (DB2 Universal Database、 DB2 Connect、DB2 Information Integrator、 DB2 Query Patroller など) を使用するために必要なすべての情報を提供します。

DB2 インフォメーション・センターは、以下の場所から呼び出すことができます。

- DB2 UDB クライアントまたはサーバーがインストールされているコンピューター
- DB2 インフォメーション・センターがインストールされているイントラネット・サーバーまたはローカル・コンピューター
- IBM の Web サイト

前提条件:

DB2 インフォメーション・センターを呼び出すための要件は、以下のとおりです。

- *オプショナル*: 希望する言語でトピックを表示するようブラウザーを構成する
- *オプショナル*: コンピューターまたはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターを使用するよう DB2 クライアントを構成する

手順:

DB2 UDB クライアントまたはサーバーがインストールされているコンピューターから DB2 インフォメーション・センターを呼び出すには、以下のようにします。

- (Windows オペレーティング・システムの)「スタート」メニューから: **「スタート」→「プログラム」→「IBM DB2」→「情報」→「インフォメーション・センター」**をクリックします。
- コマンド行プロンプトから:
  - Linux および UNIX オペレーティング・システムの場合、 **db2icdocs** コマンドを発行します。
  - Windows オペレーティング・システムの場合、 **db2icdocs.exe** コマンドを発行します。

イントラネット・サーバーまたはローカル・コンピューターにインストール済みの DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ http://<host-name>:<port-number>/ を開きます (<host-name> はホスト名、 <port-number> は DB2 インフォメーション・センターを利用可能なポート番号)。

IBM Web サイトにある DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ publib.boulder.ibm.com/infocenter/db2help/ を開きます。

関連概念:

- 416 ページの『DB2 インフォメーション・センター』

関連タスク:

- 428 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 437 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 427 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 438 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 438 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 439 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール

http://publib.boulder.ibm.com/infocenter/db2help/ から利用できる DB2 インフォメーション・センターは、資料の新規追加または変更によって定期的に更新されます。さらに、更新された DB2 インフォメーション・センターをコンピューターまたはイントラネット・サーバーにダウンロードしてインストールできる場合もあります。DB2 インフォメーション・センターを更新しても、 DB2 クライアント製品またはサーバー製品は更新されません。

前提条件:

インターネットに接続されたコンピューターへのアクセスが必要です。

手順:

DB2 インフォメーション・センターの更新をコンピューターまたはイントラネット・サーバーにインストールするには、以下のようにします。

1. IBM の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターを開きます。
2. 「DB2 インフォメーション・センターにようこそ」ページの見出し「サービスおよびサポート」の「ダウンロード」セクションで、**「DB2 資料」**リンクをクリックします。

3. 最新のドキュメンテーション・イメージのレベルと、インストール済みのドキュメンテーション・レベルを比較して、 DB2 インフォメーション・センターを更新する必要があるかどうかを確認します。「DB2 インフォメーション・センターにようこそ」ページに、インストール済みのドキュメンテーションのレベルがリストされます。
4. より新しいバージョンの DB2 インフォメーション・センターが存在する場合、ご使用のオペレーティング・システムに対応する最新の *DB2 インフォメーション・センター・イメージをダウンロードします。*
5. 最新の *DB2 インフォメーション・センター・イメージをインストールするには、* Web ページの指示に従ってください。

関連概念:

- 418 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 426 ページの『DB2 インフォメーション・センターの呼び出し』
- 420 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 423 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 インフォメーション・センターにおける特定の言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザーの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、 DB2 インフォメーション・センターでは英語でトピックが表示されます。

手順:

Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。

1. Internet Explorer の**「ツール」**—> **「インターネット オプション」**—> **「言語...」**ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
   - リストに新しい言語を追加するには、**「追加...」**ボタンをクリックします。

     注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。
   - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで**「上へ」**ボタンをクリックします。
3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

Mozilla Web ブラウザーの場合に、使いたい言語でトピックを表示するには、以下のようにします。

1. Mozilla の**「編集」**—> **「設定」**—> **「言語」**ボタンをクリックします。「設定」ウィンドウに「言語」パネルが表示されます。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
   - リストに新しい言語を追加するには、**「追加...」**ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
   - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで**「上に移動」**ボタンをクリックします。
3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

関連概念:

- 416 ページの『DB2 インフォメーション・センター』

DB2 PDF 資料および印刷された資料

以下の表は、正式な資料名、資料番号、および PDF ファイル名を示しています。ハードコピー版の資料を注文するには、正式な資料名を知っておく必要があります。 PDF ファイルを印刷するには、PDF ファイル名を知っておく必要があります。

DB2 資料は、以下のカテゴリーに分類されています。

- DB2 中核情報
- 管理情報
- アプリケーション開発情報
- ビジネス・インテリジェンス情報
- DB2 Connect 情報
- 入門情報
- チュートリアル情報
- オプショナル・コンポーネント情報
- リリース・ノート

以下の表は、DB2 ライブラリー内の各資料について、その資料のハードコピー版を注文したり、PDF 版を印刷または表示したりするのに必要な情報を示しています。DB2 ライブラリー内の各資料に関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。

DB2 の基本情報

こうした資料の情報は、すべての DB2 ユーザーに基本的なもので、プログラマーおよびデータベース管理者にとって役立つ情報であるとともに、DB2 Connect、DB2 Warehouse Manager、または他の DB2 製品を使用するユーザーにとっても役立つ内容です。

表 20. DB2 の基本情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database コマンド・リファレンス*」	SC88-9140	db2n0j81
「*IBM DB2 Universal Database 用語集*」	資料番号なし	db2t0j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 1 巻*」	GC88-9152 (ハードコピーなし)	db2m1j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 2 巻*」	GC88-9153 (ハードコピーなし)	db2m2j81
「*IBM DB2 Universal Database 新機能*」	SC88-9158	db2q0j81

管理情報

これらの資料の情報は、DB2 データベース、データウェアハウス、およびフェデレーテッド・システムを効果的に設計し、インプリメントし、保守するために必要なトピックを扱っています。

表 21. 管理情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database 管理ガイド: プランニング*」	SC88-9135	db2d1j81
「*IBM DB2 Universal Database 管理ガイド: インプリメンテーション*」	SC88-9133	db2d2j81
「*IBM DB2 Universal Database 管理ガイド: パフォーマンス*」	SC88-9134	db2d3j81
「*IBM DB2 Universal Database 管理 API リファレンス*」	SC88-9136	db2b0j81
「*IBM DB2 Universal Database データ移動ユーティリティー ガイドおよびリファレンス*」	SC88-9142	db2dmj81
「*IBM DB2 Universal Database データ・リカバリーと高可用性 ガイドおよびリファレンス*」	SC88-9143	db2haj81
「*IBM DB2 Universal Database データウェアハウス・センター 管理ガイド*」	SC88-9165	db2ddj81
「*IBM DB2 Universal Database SQL リファレンス 第 1 巻*」	SC88-9155	db2s1j81
「*IBM DB2 Universal Database SQL リファレンス 第 2 巻*」	SC88-9156	db2s2j81

表 21. 管理情報 (続き)

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database システム・モニター ガイドおよびリファレンス*」	SC88-9157	db2f0j81

アプリケーション開発情報

これらの資料の情報は、 DB2 Universal Database (DB2 UDB) のアプリケーション開発者またはプログラマーが特に関心を持つ内容です。サポートされるさまざまなプログラミング・インターフェース (組み込み SQL、ODBC、JDBC、SQLJ、CLI など) を使用して DB2 UDB にアクセスするのに必要な資料とともに、サポートされる言語およびコンパイラーについても紹介されています。また、DB2 インフォメーション・センターをご使用の場合には、サンプル・プログラムのソース・コードの HTML バージョンにアクセスすることもできます。

表 22. アプリケーション開発情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database アプリケーション開発ガイド アプリケーションの構築および実行*」	SC88-9137	db2axj81
「*IBM DB2 Universal Database アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」	SC88-9138	db2a1j81
「*IBM DB2 Universal Database アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」	SC88-9139	db2a2j81
「*IBM DB2 Universal Database コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻*」	SC88-9159	db2l1j81
「*IBM DB2 Universal Database コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻*」	SC88-9160	db2l2j81
「*IBM DB2 Universal Database データウェアハウス・センター アプリケーション統合ガイド*」	SC88-9166	db2adj81
「*IBM DB2 Universal Database XML Extender 管理およびプログラミングのガイド*」	SC88-9172	db2sxj81

ビジネス・インテリジェンス情報

これらの資料の情報は、さまざまなコンポーネントを使用して、 DB2 Universal Database のデータウェアハウジング機能および分析機能を拡張する方法を説明しています。

表 23. ビジネス・インテリジェンス情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Warehouse Manager Standard Edition インフォメーション・カタログ・センター 管理ガイド*」	SC88-9167	db2dij81
「*IBM DB2 Warehouse Manager Standard Edition インストール・ガイド*」	GC88-9164	db2idj81
「*IBM DB2 Warehouse Manager Standard Edition DB2 Warehouse Manager を使用時の ETI ソリューション・コンバージョン・プログラムの管理*」	SC88-9894	iwhe1mstx80

DB2 Connect 情報

このカテゴリーの情報は、 DB2 Connect Enterprise Edition または DB2 Connect Personal Edition を使用して、メインフレーム・サーバーおよびミッドレンジ・サーバー上のデータにアクセスする方法を説明しています。

表 24. DB2 Connect 情報

資料名	資料番号	PDF ファイル名
「*IBM コネクティビティー 補足*」	資料番号なし	db2h1j81
「*IBM DB2 Connect Enterprise Edition 概説およびインストール*」	GC88-9145	db2c6j81
「*IBM DB2 Connect Personal Edition 概説およびインストール*」	GC88-9146	db2c1j81
「*IBM DB2 Connect ユーザーズ・ガイド*」	SC88-9147	db2c0j81

入門情報

このカテゴリーの情報は、サーバー、クライアント、および他の DB2 製品をインストールして構成する場合に役立ちます。

表 25. 入門情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database DB2 クライアント機能 概説およびインストール*」	GC88-9144 (ハードコピーなし)	db2itj81
「*IBM DB2 Universal Database DB2 サーバー機能 概説およびインストール*」	GC88-9148	db2isj81
「*IBM DB2 Universal Database DB2 Personal Edition 概説およびインストール*」	GC88-9150	db2i1j81
「*IBM DB2 Universal Database インストールおよび構成 補足*」	GC88-9149 (ハードコピーなし)	db2iyj81
「*IBM DB2 Universal Database DB2 Data Links Manager 概説およびインストール*」	GC88-9141	db2z6j81

チュートリアル情報

チュートリアル情報は、DB2 機能を紹介し、さまざまなタスクを実行する方法を示します。

表 26. チュートリアル情報

資料名	資料番号	PDF ファイル名
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介*」	資料番号なし	db2tuj81
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド*」	資料番号なし	db2taj81
「*インフォメーション・カタログ・センター チュートリアル*」	資料番号なし	db2aij81
「*Video Central for e-business チュートリアル*」	資料番号なし	db2twj81
「*Visual Explain チュートリアル*」	資料番号なし	db2tvj81

オプショナル・コンポーネント情報

このカテゴリーの情報は、 DB2 のオプショナル・コンポーネントを使用する方法について説明しています。

表 27. オプショナル・コンポーネント情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Cube Views Guide and Reference*」	SC18–7298	db2aax81
「*IBM DB2 Query Patroller インストール、管理、使用法のガイド*」	GC88–9154	db2dwj81
「*IBM DB2 Spatial Extender and Geodetic Extender ユーザーズ・ガイドおよびリファレンス*」	SC88-9171	db2sbj81
「*IBM DB2 Universal Database Data Links Manager 管理ガイドおよびリファレンス*」	SC88-9169	db2z0x82
「*DB2 Net Search Extender 管理およびユーザーズ・ガイド*」 **注:** この資料の HTML 版は、HTML ドキュメンテーション CD からインストールされません。	SH88-8546	N/A

リリース・ノート

リリース・ノートは、ご使用の製品のリリースおよびフィックスパック・レベルに特有の追加情報を紹介します。また、リリース・ノートには、各リリース、アップデート、およびフィックスパックで組み込まれた資料上の更新の要約も含まれています。

表 28. リリース・ノート

資料名	資料番号	PDF ファイル名
「*DB2 リリース・ノート*」	「注」を参照。	「注」を参照。
「*DB2 インストール情報*」	製品 CD-ROM でのみ参照可能。	使用できません。

注: リリース・ノートは以下の形式で入手できます。

- XHTML およびテキスト形式 (製品 CD 内)
- PDF 形式 (PDF ドキュメンテーション CD 内)

さらに、リリース・ノートの中で、『*既知の問題と予備手段*』および『*リリース間の非互換性*』に関する部分は DB2 インフォメーション・センターにも表示されます。

UNIX ベースのプラットフォームでテキスト形式でリリース・ノートを確認するには、 `Release.Notes` ファイルを参照してください。このファイルは、`DB2DIR/Readme/%L` ディレクトリーに収録されています。 `%L` はロケール名を表しています。 `DB2DIR` は以下になります。

- AIX オペレーティング・システムの場合: /usr/opt/db2_08_01
- その他のすべての UNIX ベースのオペレーティング・システムの場合: /opt/IBM/db2/V8.1

関連概念:

- 415 ページの『DB2 資料とヘルプ』

関連タスク:

- 435 ページの『PDF ファイルからの DB2 資料の印刷方法』
- 436 ページの『DB2 の印刷資料の注文方法』
- 437 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』

PDF ファイルからの DB2 資料の印刷方法

DB2 PDF ドキュメンテーション CD に収録されている DB2 資料を印刷することができます。 Adobe Acrobat Reader を使用すれば、資料全体または特定のページを印刷できます。

前提条件:

Adobe Acrobat Reader がインストールされていることを確認してください。 Adobe Acrobat Reader をインストールする必要がある場合、 Adobe Web サイト (www.adobe.com) から入手できます。

手順:

PDF ファイルから DB2 資料を印刷するには以下のようにします。

1. *DB2 PDF* ドキュメンテーション CD をドライブに挿入します。 UNIX オペレーティング・システムの場合、 DB2 PDF ドキュメンテーション CD をマウントします。 UNIX オペレーティング・システムで CD をマウントする方法については、「*概説およびインストール*」を参照してください。
2. index.htm を開きます。ブラウザー・ウィンドウにファイルが開きます。
3. 参照したい PDF のタイトルをクリックします。 Acrobat Reader で PDF が開きます。
4. **「ファイル」 → 「印刷」**を選択して、所要の資料の任意の部分を印刷します。

関連概念:

- 416 ページの『DB2 インフォメーション・センター』

関連タスク:

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (AIX)』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『HP-UX 上での CD-ROM のマウント』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Linux)』
- 436 ページの『DB2 の印刷資料の注文方法』

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Solaris)』

関連資料:

- 429 ページの『DB2 PDF 資料および印刷された資料』

DB2 の印刷資料の注文方法

ハードコピー版の資料を望む場合には、以下のいずれかの方法で注文できます。

印刷資料の注文方法:

一部の国または地域では、印刷された資料を注文することもできます。お客様がお住まいの国または地域でこのサービスが利用可能かどうかを確認するには、お住まいの国または地域の IBM Publications Web サイトをご覧ください。資料のご注文が可能な場合、以下のようにすることができます。

- 正規の IBM 製品販売業者または営業担当員に連絡してください。お客様がお住まいの地域の IBM 担当員の情報については、お手数ですが IBM の Web サイト (www.ibm.com/planetwide) の IBM Worldwide Directory of Contacts で確認してください。
- IBM Publications Center (http://www.ibm.com/shop/publications/order) にアクセスしてください。なお、IBM Publications Center から資料を注文できない国もあります。

DB2 製品がご利用可能になった時点で、印刷された資料は *DB2 PDF* ドキュメンテーション *CD* にある PDF 形式の資料と同じものです。さらに、 *DB2* インフォメーション・センター *CD* に収録されている印刷された資料の内容もまた、これらと同じです。ただし、DB2 インフォメーション・センター CD には、 PDF 資料にない追加情報も含まれます (たとえば、SQL 管理作業や HTML サンプル)。 DB2 PDF ドキュメンテーション CD に収録されている資料の中には、ハードコピーとしてご注文できない資料もあります。

注: DB2 インフォメーション・センターは、 PDF またはハードコピー の資料よりも頻繁に更新されます。ドキュメンテーションの更新が入手可能になった時点でインストールするか、 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) を参照して最新の情報を入手してください。

関連タスク:

- 435 ページの『PDF ファイルからの DB2 資料の印刷方法』

関連資料:

- 429 ページの『DB2 PDF 資料および印刷された資料』

DB2 ツールからコンテキスト・ヘルプを呼び出す

コンテキスト・ヘルプは、特定のウィンドウ、ノートブック、ウィザード、またはアドバイザーに関連したタスクまたはコントロールの情報を提供します。コンテキスト・ヘルプは、グラフィカル・ユーザー・インターフェースのある DB2 管理ツールおよび開発ツールから利用できます。コンテキスト・ヘルプには、以下の 2 種類があります。

- それぞれのウィンドウまたはノートブックにある**「ヘルプ」**ボタンからアクセス可能なヘルプ
- infopop (ポップアップ情報ウィンドウ)。これは、マウス・カーソルを特定のフィールドまたはコントロール上に置いたとき、またはウィンドウ、ノートブック、ウィザード、アドバイザー内でフィールドまたはコントロールを選択して F1 を押すと表示されます。

「ヘルプ」ボタンを押すと、概説、前提条件、およびタスク情報が表示されます。infopop は、それぞれのフィールドおよびコントロールについて説明します。

手順:

コンテキスト・ヘルプを呼び出すには、以下のようにします。

- ウィンドウおよびノートブックのヘルプを表示するには、いずれかの DB2 ツールを開始して、任意のウィンドウまたはノートブックを開きます。ウィンドウまたはノートブックの右下隅にある**「ヘルプ」**ボタンをクリックして、コンテキスト・ヘルプを呼び出します。

  また、それぞれの DB2 ツール・センターの上部にある**「ヘルプ」**メニュー項目からコンテキスト・ヘルプにアクセスすることもできます。

  ウィザードおよびアドバイザーでは、最初のページの「タスクの概要」リンクをクリックすると、コンテキスト・ヘルプを表示できます。
- ウィンドウまたはノートブック上の各コントロールの infopop ヘルプを表示するには、コントロールをクリックしてから、**F1** を押します。コントロールの詳細情報を示すポップアップ情報が、黄色いウィンドウに表示されます。

  注: フィールドまたはコントロールにマウス・カーソルを置いておくだけで infopops が表示されるようにするには、「ツール設定」ノートブックの**「文書 (Documentation)」**ページの**「infopops の自動表示」**チェック・ボックスを選択します。

  infopop に似た別のコンテキスト・ヘルプに、診断ポップアップ情報があります。これにはデータ入力規則が示されます。診断ポップアップ情報は、無効または不十分なデータが入力されたとき、紫色のウィンドウに表示されます。診断ポップアップ情報は、以下に関して表示されます。
  - 必須フィールド。
  - 日付フィールドのように、正確なフォーマットを必要とするデータのフィールド。

関連タスク:

- 426 ページの『DB2 インフォメーション・センターの呼び出し』
- 438 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』

- 438 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 439 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』
- 『DB2 UDB ヘルプの使用法: Common GUI help』

コマンド行プロセッサーからメッセージ・ヘルプを呼び出す

メッセージ・ヘルプは、メッセージが出された原因と、エラーへの応答として実行すべきアクションを説明します。

手順:

メッセージ・ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? XXXnnnnn
```

ここで、*XXXnnnnn* は有効なメッセージ ID を表します。

たとえば、`? SQL30081` と入力すると、メッセージ SQL30081 に関するヘルプを表示します。

関連概念:

- 「*メッセージ・リファレンス 第 1 巻*」の『メッセージの概要』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーからコマンド・ヘルプを呼び出す

コマンド・ヘルプは、コマンド行プロセッサーでのコマンドの構文を説明します。

手順:

コマンド・ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? command
```

ここで *command* はキーワードまたはコマンド全体を表します。

たとえば、`? catalog` と入力すると、すべての CATALOG コマンドに関するヘルプが表示され、 `? catalog database` と入力すると、 CATALOG DATABASE コマンドのヘルプだけが表示されます。

関連タスク:

- 437 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 426 ページの『DB2 インフォメーション・センターの呼び出し』
- 438 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 439 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーから SQL 状態ヘルプを呼び出す

DB2 Univerrsal Database は、 SQL ステートメントの結果の原因となったと考えられる条件の SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順:

SQL 状態ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate または ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。

たとえば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

関連タスク:

- 426 ページの『DB2 インフォメーション・センターの呼び出し』
- 438 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 438 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』

DB2 チュートリアル

DB2® チュートリアルは、 DB2 Universal Database のさまざまな機能について学習するのを支援します。このチュートリアルでは、アプリケーションの開発、SQL 照会のパフォーマンス調整、データウェアハウスの処理、メタデータの管理、および DB2 を使用した Web サービスの開発の各分野で、段階的なレッスンが用意されています。

はじめに:

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) から、このチュートリアルの XHTML 版を表示できます。

チュートリアルの中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、それぞれのチュートリアルを参照してください。

DB2 Universal Database チュートリアル:

以下に示すチュートリアルのタイトルをクリックすると、そのチュートリアルを表示できます。

ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介
データウェアハウス・センターを使用して簡単なデータウェアハウジング・タスクを実行します。

ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド
データウェアハウス・センターを使用して高度なデータウェアハウジング・タスクを実行します。

インフォメーション・カタログ・センター・チュートリアル
インフォメーション・カタログを作成および管理して、インフォメーション・カタログ・センターを使用してメタデータを配置し使用します。

Visual Explain チュートリアル
Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

DB2 トラブルシューティング情報

DB2® 製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 ドキュメンテーション
トラブルシューティング情報は、DB2 インフォメーション・センター、および DB2 ライブラリーに含まれる PDF 資料の中でご利用いただけます。DB2 インフォメーション・センターで、 (ブラウザー・ウィンドウの左側の) ナビゲーション・ツリーの「サポートおよびトラブルシューティング (Support and troubleshooting)」ブランチを参照すると、 DB2 トラブルシューティング・ドキュメンテーションの詳細なリストが見つかります。

DB2 Technical Support の Web サイト
現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、 DB2 Technical Support の Web サイトを参照してください。Technical Support サイトには、最新の DB2 出版物、TechNotes、プログラム診断依頼書 (APAR)、フィックスパック、DB2 内部エラー・コードの最新リスト、その他のリソースが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

DB2 Technical Support の Web サイト
(http://www.ibm.com/software/data/db2/udb/winos2unix/support) にアクセスしてください。

DB2 Problem Determination Tutorial Series
DB2 製品で作業中に直面するかもしれない問題を素早く識別し、解決する方法に関する情報を見つけるには、 DB2 Problem Determination Tutorial Series の Web サイトを参照してください。あるチュートリアルでは、使用可能な DB2 問題判別機能およびツールを紹介し、それらをいつ使用すべきかを判断する助けを与えます。別のチュートリアルは、『データベース・エンジン問題判別 (Database Engine Problem Determination)』、『パフォーマンス問題判別 (Performance Problem Determination)』、『アプリケーション問題判別 (Application Problem Determination)』などの関連トピックを扱っています。

DB2 Technical Support
(http://www.ibm.com/software/data/support/pdm/db2tutorials.html) には、 DB2 問題判別チュートリアルがすべて揃っています。

関連概念:

- 416 ページの『DB2 インフォメーション・センター』
- 「*問題判別の手引き*」の『Introduction to Problem Determination - DB2 テクニカル・サポートのチュートリアル』

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。 DB2® バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 ユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、 442 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java™ Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、 442 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、 442 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、 DB2 ツールのユーザー・インターフェースをナビゲートできます。

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

アクセスしやすい表示

DB2 ツールには、視力の弱いユーザー、その他の視力障害をもつユーザーのためにアクセシビリティーを向上させる機能が備わっています。これらのアクセシビリティー拡張機能には、フォント・プロパティーのカスタマイズを可能にする機能も含まれています。

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、メニューおよびテキストのフォントを変更する: Common GUI help を参照してください。

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

DB2 形式は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。 XHTML により、ご使用のブラウザーに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン・ドキュメンテーションにアクセスする場合にのみ使用できます。

関連概念:

• 442 ページの『ドット 10 進シンタックス・ダイアグラム』

ドット 10 進シンタックス・ダイアグラム

スクリーン・リーダーを使用してインフォメーション・センターを利用するユーザーのために、シンタックス・ダイアグラムがドット 10 進形式で提供されます。

ドット 10 進形式では、各シンタックス・エレメントは別々の行に書き込まれます。複数のシンタックス・エレメントが常に同時に存在する (または常に同時に不在の) 場合、単一のコンパウンド・シンタックス・エレメントとみなせるので同一行に表示できます。

各行は、ドット 10 進数で開始します。たとえば、3 または 3.1 ないしは 3.1.1 です。こうした数を適切に聞き取るには、スクリーン・リーダーが句読点を読み取るように設定されていることを確認してください。同じドット 10 進数を持つすべてのシンタックス・エレメント (たとえば、 3.1 という数値を持つすべてのシンタックス・エレメント) は、相互に排他的な代替エレメントです。 3.1 USERID および 3.1 SYSTEMID という行を聞き取る場合、シンタックスには両方ではなく USERID または SYSTEMID のどちらかが含まれることが分かります。

ドット 10 進レベルは、ネストのレベルを表示します。たとえば、ドット 10 進数 3 のシンタックス・エレメントの後に、一連のドット 10 進数 3.1 のシンタックス・エレメントが続きます。 3.1 の番号が付されたシンタックス・エレメントすべては、番号 3 の付されたシンタックス・エレメントに従属します。

シンタックス・エレメントに関する情報を追加するため、ドット 10 進数の次に特定のワードおよびシンボルが使用されます。 時折、こうしたワードおよびシンボルはエレメントの最初に表示される場合もあります。 簡単に識別するため、ワードやシンボルがシンタックス・エレメントの一部である場合には、円記号 (¥) 文字が先頭に付きます。 * シンボルはドット 10 進数の次に使用でき、シンタックス・エレメントが反復することを示します。 たとえば、ドット 10 進数 3 のシンタックス・エレメント *FILE は、3 ¥* FILE という形式になります。 3* FILE という形式は、シンタックス・エレメント FILE が反復されることを示します。 3* ¥* FILE という形式は、シンタックス・エレメント * FILE が反復されることを示します。

シンタックス・エレメントのストリングを分離するのに使用されるコンマなどの文字は、シンタックス内の分離する項目の直前に表示されます。 こうした文字は、それぞれの項目と同一行に表示するか、同じドット 10 進数を持つ関連する項目のある別の行に表示できます。 またその行には、シンタックス・エレメントに関する情報を提供する別のシンボルを表示することも可能です。 たとえば、複数の LASTRUN および DELETE シンタックス・エレメントを使用している場合には、5.1*、5.1 LASTRUN、および 5.1 DELETE という行は、エレメントをコンマで区切る必要があります。区切り文字が指定されないと、各シンタックス・エレメントを区切るのにブランクが使用されると想定されます。

シンタックス・エレメントの前に % シンボルが付く場合、他の箇所で定義されている参照であることを示します。 % シンボルの後のストリングは、リテラルではなくシンタックス・フラグメントの名前です。 たとえば、2.1 %OP1 という行は別のシンタックス・フラグメント OP1 を参照すべきことを意味します。

以下のワードおよびシンボルが、ドット 10 進数の次に使用されます。

- ? は、オプションのシンタックス・エレメントであることを表します。? シンボルが後に続くドット 10 進数は、対応するドット 10 進数のシンタックス・エレメント、および任意の従属のシンタックス・エレメントがオプションであることを示します。ドット 10 進数の付いたシンタックス・エレメントが 1 つしかない場合、 ? シンボルはそのシンタックス・エレメントと同じ行に表示されます (たとえば、5? NOTIFY)。ドット 10 進数の付いたシンタックス・エレメントが複数

ある場合、 ? シンボルだけで行に表示され、その後にオプションのシンタックス・エレメントが続きます。たとえば、「5 ?、5 NOTIFY、および 5 UPDATE」という行を聞き取る場合、シンタックス・エレメント NOTIFY および UPDATE がオプションである、つまりそのいずれかを選択でき、どちらも選択しないこともできることが分かります。 ? シンボルは、線路型ダイアグラムのバイパス線に相当します。

- ! は、デフォルトのシンタックス・エレメントであることを表します。! シンボルおよびシンタックス・エレメントが後に続くドット 10 進数は、そのシンタックス・エレメントが、同じドット 10 進数を共用するシンタックス・エレメントすべてのデフォルト・オプションであることを示します。同じドット 10 進数を共用するシンタックス・エレメントのうち 1 つだけに、 ! シンボルを指定できます。たとえば、「2? FILE、2.1! (KEEP)、および 2.1 (DELETE)」という行を聞き取る場合、 FILE キーワードのデフォルト・オプションは (KEEP) になります。この例では、FILE キーワードを含めてもオプションを指定しない場合には、デフォルト・オプション KEEP が適用されます。デフォルト・オプションは、次に高位のドット 10 進数にも適用されます。 この例の場合、FILE キーワードが省略されると、デフォルトの FILE(KEEP) が使用されます。しかし、「2? FILE、2.1、2.1.1! (KEEP)、および 2.1.1 (DELETE)」という行を聞き取る場合、デフォルト・オプション KEEP は次に高位のドット 10 進数 2.1 (関連キーワードを持っていない) にのみ適用され、 2? FILE には適用されません。キーワード FILE が省略されると、どれも使用されません。
- * は、0 回以上反復できるシンタックス・エレメントを示します。 * シンボルが後に続くドット 10 進数は、このシンタックス・エレメントが 0 回以上使用できること、つまりオプションであり、なおかつ反復できることを表します。たとえば、5.1* データ域という行を聞き取る場合、 1 つまたは複数のデータ域を含めるか、またはデータ域を全く含めないことが可能です。「3*、3 HOST、および 3 STATE」という行を聞き取る場合、 HOST、STATE をどちらか一方または両方同時に含めるか、どちらも含めないことができます。

  注:

  1. ドット 10 進数の後にアスタリスク (*) が付き、ドット 10 進数の付いた項目が 1 つしかない場合には、同じ項目を複数回反復できます。
  2. ドット 10 進数の後にアスタリスクが付き、ドット 10 進数の付いた項目が複数ある場合、リストから複数の項目を使用できますが、各項目を複数回使用することはできません。 前述の例では、HOST STATE と書くことはできますが、HOST HOST とは書けません。
  3. * シンボルは、線路型シンタックス・ダイアグラムのループバック線に相当します。
- + は、1 回以上含める必要のあるシンタックス・エレメントであることを示します。 + シンボルが後に続くドット 10 進数は、このシンタックス・エレメントを 1 回以上含める必要があること、つまり少なくとも 1 回は含める必要があり、反復できることを表します。たとえば、「6.1+ データ域」という行を聞き取る場合、データ域を少なくとも 1 回は含めなければなりません。「2+、2 HOST、および 2 STATE」という行を聞き取る場合には、 HOST、STATE、またはその両方を含める必要があります。 * シンボルと同様に、+ シンボルは、ドット 10 進

数の付いた項目が 1 つしかない場合に限り、その特定の項目のみを反復できます。 * シンボルと同様、 + シンボルは線路型シンタックス・ダイアグラムのループバック線に相当します。

関連概念:

- 441 ページの『アクセス支援』

関連タスク:

- 『目次』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『構文図の見方』

DB2 Universal Database 製品の共通基準認証

DB2 Universal Database は、 Common Criteria の評価検定レベル 4 (EAL4) で認証の評価を受けています。 Common Criteria の詳細については、以下の Common Criteria の Web サイトを参照してください。 http://niap.nist.gov/cc-scheme/

付録 J. 特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム(本プログラムを含む)との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

商標

以下は、IBM Corporation の商標です。

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

以下は、それぞれ各社の商標または登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Pentium は、Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[数字]

A

B

D

E

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

IBM と連絡をとる

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

DB2 Universal Database 製品に関する情報は、
http://www.ibm.com/software/data/db2/udb から入手できます。

このサイトには、技術ライブラリー、資料の注文方法、製品のダウンロード、ニュースグループ、フィックスパック、ニュース、および Web リソースへのリンクに関する最新情報が掲載されています。

米国以外の国で IBM に連絡する方法については、 IBM Worldwide ページ (www.ibm.com/planetwide) にアクセスしてください。

Printed in Japan

SC88-9143-01

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12