IBM® DB2® ユニバーサル・データベース

データ回復と高可用性の手引きと解説書

バージョン 7

SC88-9019-00
(英文原典：DATA-RCVR-00)

IBM® DB2® ユニバーサル・データベース

データ回復と高可用性の手引きと解説書

バージョン 7

SC88-9019-00
(英文原典：DATA-RCVR-00)

ご注意!

本書の情報およびこの情報がサポートするプロダクトをご使用になる前に、 523ページの『付録K. 特記事項』に記載されている情報をお読みください。

本書には、IBM の専有情報が含まれています。その情報は、使用許諾条件に基づき提供され、著作権により保護されています。本書に記載される情報には、いかなる製品の保証も含まれていません。また、本書で提供されるいかなる記述も、製品保証として解釈すべきではありません。

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)

原 典:	DATA-RCVR-00 IBM® DB2® Universal Database Data Recovery and High Availability Guide and Reference Version 7
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 2001.8

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 ユニバーサル・データベース (UDB) のバックアップ、復元、および回復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、復元、および回復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 ユニバーサル・データベース、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。DB2 UDB についての一般情報については、*管理の手引き* を参照してください。 SQL については、*SQL 解説書* を参照してください。 DB2 UDB コマンド行プロセッサーの構成、起動、および使用については、 *コマンド解説書* を参照してください。 DB2 UDB アプリケーション・プログラミング・インターフェース (API) については、 *管理 API 解説書* を参照してください。 DB2 管理用 API を含むアプリケーションの作成についての一般情報は、 *アプリケーション構築の手引き* を参照してください。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。インストール情報は、 *概説およびインストール* マニュアル (各オペレーティング・システムに対応するもの) に記載されています。

本書の構成

以下のトピックについて説明します。

データ回復

第1章 適切なバックアップと回復の方針の開発
データベースおよび表スペースの回復方式 (データベースおよび表スペースのバックアップと回復、およびロールフォワード回復方式の使用を含む) 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 データベース復元

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 復元ユーティリティーについて説明します。

第4章 ロールフォワード回復

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回復する際に使用する、 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。

高可用性

第5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回復サポートの概要を説明します。

第6章 AIX での高可用性

AIX での高可用性回復機能の 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7章 Windows オペレーティング・システムでの高可用性

Windows NT での高可用性回復機能の 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8章 Solaris 実行環境での高可用性

Solaris 実行環境での高可用性回復機能の 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 2.x (SC2.x)、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

回復関連の 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

回復関連の API とそのデータ構造について説明します。

付録E. 回復サンプル・プログラム

回復関連の 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. 回復 CLP スクリプト

回復関連の CLP コマンドを含む DB2 コマンド・スクリプト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G. Tivoli Storage Manager

Tivoli Storage Manager (TSM、以前の ADSM) 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H. データベース回復用のユーザー出口

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I. ベンダー製品用のバックアップおよび復元 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 データ回復

第1章 適切なバックアップと回復の方針の開発

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、およびアプリケーション上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、障害が異なれば異なった回復処置が必要になります。十分にテストされた適切な回復方針を開発することにより、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。回復方針を開発する際には、以下の要素を考慮する必要があります。データベースは回復可能か。データベース回復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。

データベース回復方針では、データベース回復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、区分データベース・システムの場合はシステム規模の変更時 (データベース区画サーバーまたはノードの追加や除去による) 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・コピーの回復手順も戦略全体に組み込む必要があります。

以下に、各種の回復方式について説明し、業務環境に最適な回復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*回復* といいます。*バージョン回復* は、以前のバージョンのデータベースの回復であり、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。*ロールフォワード回復* では、データベースまたは表スペースのバックアップ・イメージが復元された後で、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。

破損回復 では、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、データベースが自動的に回復されます。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。

これらの回復方式に関する詳細は、 22ページの『バージョン回復』、23ページの『ロールフォワード回復』、または 9ページの『破損回復』を参照してください。

データベースを作成すると、ログ・ファイルとリカバリー・ヒストリー・ファイルが自動的に作成されます (図1)。リカバリー・ログ・ファイルやリカバリー・ヒストリー・ファイルは直接変更できません。これらは、消失または損傷したデータを回復するためにデータベース・バックアップ・イメージを使用する必要があるときに重要になります。

データベース・オブジェクト / 概念	等価の物理オブジェクト

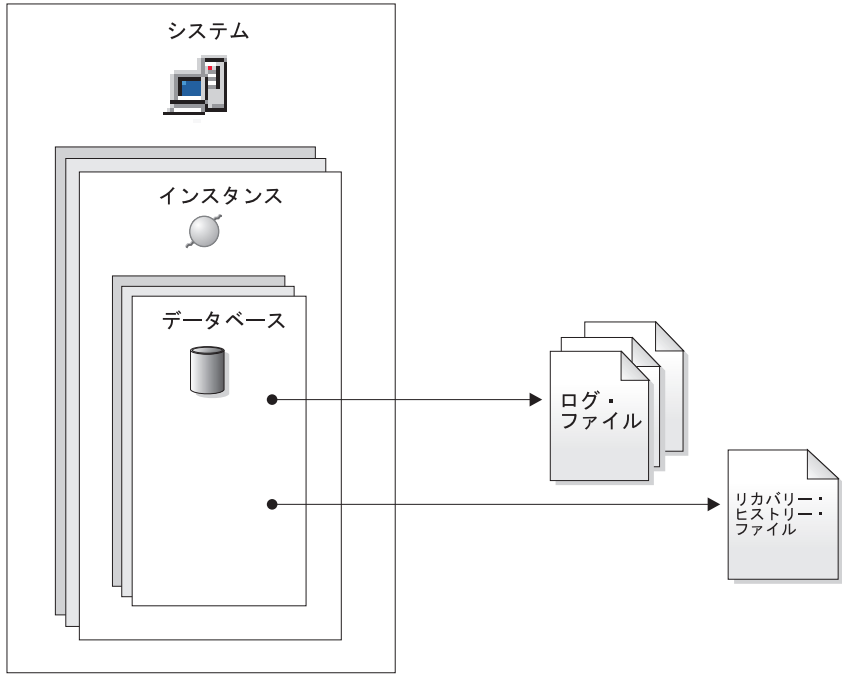


図 1. リカバリー・ログ・ファイルおよびリカバリー・ヒストリー・ファイル

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・エラーから回復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回復するために使用されます。

リカバリー・ヒストリー・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回復する必要のある場合に使用できるバックアップ情報の要約が含まれています。これは特に、バックアップ操作や復元操作などの回復関連のイベントを追跡するために使用します。

容易に再作成できるデータは、回復不能データベースに保管できます。このデータベースには、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復元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。*回復不能データベース* では、 *logretain* と *userexit* の両方のデータベース構成パラメーターが使用不能になっています。これは、保管されているログのみが破損回復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・バックアップを使ってバージョン回復を行うことは、基本的には回復不能データベースの回復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、オフラインでのみ復元できます。復元すると、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。

容易に再作成できない データは、回復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。*回復可能データベース* では、 *logretain* データベース構成パラメーターが 『RECOVERY』 に設定されているか、 *userexit* データベース構成パラメーターが使用可能になっています。または、これらの両方が行われています。アクティブ・ログを破損回復で使用できますが、*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、オフラインでのみ復元できます。復元すると、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、ロールフォワード回復では、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。

回復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)。データベースの復元とロールフォワード操作は、常にオフラインで実行しなければなりません。バックアップ操作中にロールフォワード回復は、すべての 表の変更がキャプチャーされ、バックアップが復元された時に再適用されることを保証します。

回復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、復元、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復元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。

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、回復計画では、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 (26ページの『増分バックアップおよび回復』を参照) 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、ロールフォワード回復が可能になる)、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。表スペースのバックアップ・イメージの集合からデータベースを再構築するより、全データベースのバックアップ・イメージからデータベースを回復する方が容易です。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・エラーから回復する場合に有効です。

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。

頻繁に使用されるデータベースの回復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。ロールフォワード回復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。ロールフォワード回復を使用でき、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、回復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オンライン・バックアップ操作より高速です。

データベースの中に長形式フィールドとラージ・オブジェクト (LOB) のデータが大量に含まれている場合には、データベースのバックアップ作業に非常に長い時間がかかり

ます。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。長形式フィールドと LOB データを別個の表スペースに格納してあるなら、長形式フィールドと LOB データが入っている表スペースのバックアップはとらないように選択することによって、バックアップ操作の完了にかかる時間を短くすることができます。長形式フィールドおよび LOB データがビジネスにとって重要である場合には、これらの表スペースのバックアップの作成と、これらの表スペースの復元操作の完了にかかる時間との関係を考慮してください。 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の特別な考慮事項を以下に示します。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時刻指定ロールフォワード回復は実行できません。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、表スペースを復元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。

データベースの回復に必要な時間は、次の 2 つの要素で構成されます。つまり、バックアップの復元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回復について使用可能である場合) です。回復計画を公式化するときは、これらの回復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。全般的な回復計画をテストすることで、データベースを回復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。回復方針の一部としてロールフォワード回復を実行する場合は、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、復元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。

ストレージについての考慮事項

どの回復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。

バージョン回復方式の場合は、データベースと復元後のデータベースを入れるスペースが必要になります。ロールフォワード回復方式の場合は、データベースまたは表スペースのバックアップ・コピー、復元後のデータベース、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、回復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回復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。

LOB が入っている SMS 表スペースのバックアップは、元の表スペースのサイズより大きくなっている可能性があります。表スペースに含まれる LOB データ・サイズによっては、バックアップは最大 40 % 大きくなる可能性があります。たとえば、1 GB の SMS 表スペース (LOB が入っている) のバックアップを取る場合には、復元する際に 1 GB より大きいディスク容量が必要になります。このことが起こるのは、予備割り振りをサポートするファイル・システム (たとえば、 UNIX ベースのオペレーティング・システム) だけです。

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。 (このパラメーターも含め、ロギング関連の構成パラメーターについては、 37ページの『データベース・ロギングの構成パラメーター』で説明します。)

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。ロールフォワード回復方法を使用する場合は、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。

- データベース・ログ・パス・ディレクトリーの中に、ログを入れるためのスペースを十分にとる。
- 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。たとえば OS/2 の場合は、標準および非標準装置に存在するデータベースおよびデータベース・ログの両方のデータベース・バックアップ・イメージのストレージを処理するためのユーザー出口プログラムが、 DB2 でサポートされています。 (詳細については、457ページの『付録H. データベース回復用のユーザー出口』を参照してください。)

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。回復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、すべてのデータの一貫性が保証されている時点まで、回復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回復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

複数のオペレーティング・システムのある環境で作業する際には、ほとんどの場合バックアップおよび回復の計画を統合できないことを考慮しなければなりません。つまり、一方のオペレーティング・システムでデータベースのバックアップを取り、別のオペレーティング・システムでそのデータベースを復元することは普通はできません。このような場合、それぞれのオペレーティング・システムの回復計画は、別個に独立して保持しなければなりません。

しかしながら、 Sun Solaris と HP の間のプラットフォーム間バックアップ操作および復元操作はサポートされています。システム間でバックアップ・イメージを転送する際には、バイナリー・モードで転送しなければなりません。ターゲット・システムでデータベースを作成する際のコード・ページとテリトリーは、オリジナルのデータベースが作成されたシステムのものと同じでなければなりません。

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。これらのユーティリティーの詳細については、*データ移動ユーティリティー 手引きおよび解説書* を参照してください。

破損回復

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。*破損回復* とは、データベースを一貫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、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 (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 ″一貫性ポイント″ にあることになります。

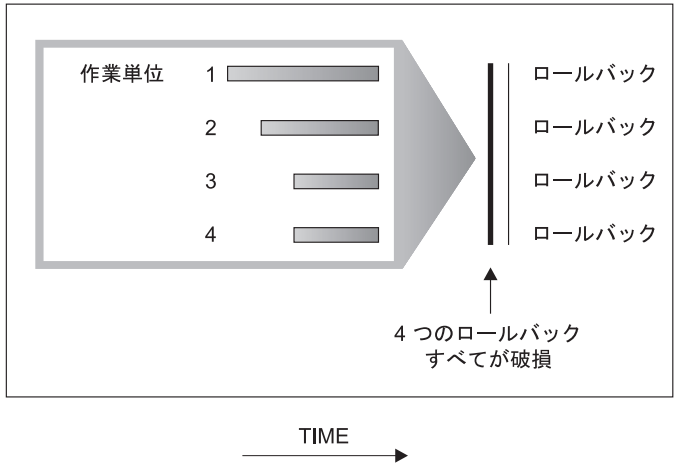


図 2. 作業単位のロールバック (破損回復)

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、または使用不能な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、データベースを回復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区画がダウンします。
- 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。このパラメーターに関する詳細については、*管理の手引き: パフォーマンス* を参照してください。) 自動再始動を動作したくない場合、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。データベースが再始動操作を開始すると、 db2diag.log ファイルに記録されます。

順方向回復が使用可能になっている (つまり、 *logretain* 構成パラメーターが RECOVERY にセットされているか、または *userexit* 構成パラメーターが使用可能になっている) データベースで破損回復を実行する場合に、個別の表スペースが原因で破損回復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らなければならず、修復されるまで

アクセスできなくなります。破損回復は続行します。破損回復が完了した時点で、データベースに入っている他の表スペースは通常は使用可能であり、データベースへの接続は確立できます。

損傷を受けた表スペースの回復

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。

- データベースを復元する
- カタログ表スペースを復元する。 (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、表スペースの復元は回復可能なデータベースについてのみ有効です。)

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一時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一時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、新しい一時表スペースが使用できるようになり、一時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、任意でオフライン一時表スペースを除去します。システム一時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。
- 損傷を受けた一時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。

回復可能データベースの表スペース回復

破損回復が必要であるため、損傷を受けた表スペースは、アクセス不能状態でオフラインに入れられ、さらにロールフォワード保留状態になります。他に問題がなければ再始動操作は成功します。損傷を受けた表スペースは、次の場合に再び使用できます。

- 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、それから表スペースのロールフォワード操作を完了した。 (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。)

- 損傷のあるコンテナーを修正した (元のデータが失われる場合も失われない場合もある) 後に、表スペースの復元を実行し、それからロールフォワード操作を実行した。

回復不能データベースの表スペース回復

破損回復は必要であり、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除去するのをいとわない場合にのみ、再始動操作を成功させることができます。 (正常な回復とは、損傷を受けた表スペースを回復して整合性のある状態に戻す必要のあるログ・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、これらを除去することだけです。)

これを行うには、修飾されていないデータベース再始動操作を呼び出します。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、`db2diag.log` 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できます。

- データベースの再始動操作が完了したときにこれらの表をすべて除去したい場合は、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できます。損傷を受けた表スペースが DROP PENDING TABLESPACES リストにある場合、表スペースは除去保留状態に入れられているため、回復後の唯一のオプションは、表スペースの除去になります。再始動操作は、この表スペースを回復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。
- これらの表スペースを除去したくない (そこにあるデータを失いたくない) 場合は、以下のいずれかを実行します。
  - 待機して、損傷を受けているコンテナーを修正し (元のデータを失わないようにする)、それからデータベース再始動の操作を行います。
  - データベースの復元操作を実行します。

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。これは、表スペースが再始動操作中に損傷を受けた場合にのみ起こることです。再始動操作が成功したら、TABLESPACE ステートメントを出して除去保留状態にある各表スペースを除去する必要があります (LIST TABLESPACES コマンドを使用して、どの表スペースがこの状態であるかを調べてください)。このようにして、スペースをレクラメーション処理するか、または表スペースを再作成できます。

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回復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写を行う。

- RAID (Redundant Array of Independent Disks) 構成、たとえば RAID レベル 5 などを使用する。 RAID について詳しくは、『ディスク障害に対する保護』を参照してください。
- 区分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 を使用することで行われます。ディスク・アレイはいくつかのディスク・ドライブをまとめたもので、アプリケーションからは 1 つの大きなディスクのように見えます。

ディスク・アレイには、ディスク・ストライピング (複数ディスク間でファイルを分散すること)、ディスクのミラーリング、およびデータ・パリティー検査が関係してきます。

ディスク・アレイは、単に RAID (Redundant Array of Independent Disks) 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・コントローラーに接続されています (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、データ保護で妥協することなく、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、これを実装すると通常の 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ユーザーはアレイ上のデータにアクセスできます。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 (『ソフトウェア・ディスク・アレイ』を参照)。これは、障害点までの回復可能性があり、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。 (ディスク障害発生後にただちにデータを回復できるように) 高信頼性が必要であるが、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、以下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 (13ページの『ハードウェア・ディスク・アレイ』を参照)、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 (13ページの『ハードウェア・ディスク・アレイ』を参照)。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。

- 中断されない電源供給
- データベース・ログに十分なディスク・スペース
- 区分データベース環境においては、データベース区画サーバー間の高信頼性通信リンク
- 区分データベース環境では、システム・クロックの同期 (148ページの『区分データベース・システムにおいてクロックを同期化する』を参照)

区分データベース環境におけるトランザクション障害回復

区分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区画サーバーと、トランザクションに参加していた他のデータベース区画サーバーとの両方で、データベース回復処理を実行する必要があります。

- 破損回復は、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区画サーバーで実行されます。
- 他の (アクティブのままの) データベース区画サーバーにおける*データベース区画回復処理* は、障害が検出された直後に行われます。

区分データベース環境では、アプリケーションが実行依頼されているデータベース区画サーバーは調整プログラム・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調整エージェントです。調整エージェントは他のデータベース区画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、調整エージェントは 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、調整プログラム・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区画サーバーに対し 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、調整プログラム・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、調整プログラム・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区画サーバーがコミットを完了すると、それらのサーバーは調整プログラム・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 肯定応答を調整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、調整エージェントは FORGET ログ・レコードを書き出します。

2 フェーズ・コミットの詳細については、*管理の手引き: 計画* を参照してください。

アクティブ・データベース区画サーバーにおける障害回復

データベース区画サーバーが他のサーバーのダウンを検出すると、障害データベース区画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・データベース区画サーバーがアプリケーションの調整プログラム・ノードで、障害データベース区画サーバー (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、調整エージェントは障害回復を実行するための割り込みが行われます。調整エージェントが 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、調整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。
- 障害データベース区画サーバーがアプリケーションの調整プログラム・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、障害回復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・データベース区画サーバーでトランザクションが未確定のままとされ、調整プログラム・ノードにはこのことが通知されません (調整プログラム・ノードが利用不能のため)。

  未確定トランザクションの解決に関する詳細な説明については、 *管理の手引き: 計画* を参照してください。

- 障害データベース区画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・データベース区画サーバーも障害データベース区画サーバーも調整プログラム・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。調整プログラム・ノードは、 ROLLBACK または切断メッセージを他のデータベース区画サーバーに送信します。調整プログラム・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区画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。

障害データベース区画サーバーにおけるトランザクション障害回復

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、プロセッサー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。プロセッサー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。詳細については、*コマンド解説書* を参照してください。

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区画は矛盾状態になることがあります。データベース区画を使用可能にするために、破損回復を、データベース区画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。

破損回復ではアクティブ・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。区分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。

- 調整プログラム・ノードではないデータベース区画サーバーでは、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。
- 調整プログラム・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、調整エージェントが、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。

破損回復では、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、データベース区画サーバーがアプリケーションの調整プログラム・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調整プログラム・ノードでない場合は、そのサーバーは調整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調整プログラム・ノードである 場合、そのサーバーは調整エージェントが 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。

破損回復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区画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、破損回復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区画サーバーを、できるだけ早期に回復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回復が間に合わない場合に、他のサーバーのデータベース区画にアクセスしなければならないときは、発見的手法の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 (*コマンド解説書* を参照) 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 ユニバーサル・データベース エンタープライズ拡張エディション。これは、区分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。

分散環境についての詳細は、*管理の手引き: 計画* を参照してください。

障害のあるデータベース区画サーバーの識別

データベース区画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区画サーバーが 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区画サーバーがトランザクションの調整プログラム・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区画サーバーがトランザクションの調整プログラム・ノードでないときに受け取られます。

どのデータベース区画サーバーが失敗したかの判別は、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区画サーバーでは、障害サーバーのノード番号を示すメッセージが `db2diag.log` ファイルに書き込まれます。

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。

データベース区画サーバーの障害から回復するためには、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区画サーバーから、 **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区画サーバー (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。

ホスト上の未確定トランザクションの回復

トランザクションでアプリケーションがホストまたは AS/400 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回復される方法は多少異なります。

ホストまたは AS/400 データベース・サーバーにアクセスするのに、DB2 コネクト が使用されます。 DB2 コネクトに DB2 Syncpoint Manager が構成されている場合、回復ステップは異なります。

DB2 コネクトに DB2 Syncpoint Manager が構成されている場合の回復

ホストまたは AS/400 サーバーにおける未確定トランザクションの回復は、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 AS/400 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 AS/400 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。発見的手法の決定を行う必要があるとホストまたは AS/400 の管理者が判断すると、ホストまたは AS/400 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 コネクト・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 AS/400 サーバーのローカル別名です。 partner_lu がホストまたは AS/400 サーバーの完全修飾 LU 名です。これはホストまたは AS/400 サーバーを最もよく識別できるもので、ホストまたは AS/400 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固有 ID を提供するもので、すべてのホストまたは AS/400 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。詳細については、コマンド*解説書* を参照してください。

```
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。

生じたとは考えにくいが可能性はある、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 コネクト・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 コネクトが発見的手法の決定に基づいて、ホストまたは AS/400 サーバーとの再同期を行えるようにします。

DB2 コネクトが DB2 Syncpoint Manager を使用しない場合の回復

DB2 コネクト パーソナル・エディションまたは DB2 コネクト エンタープライズ・エディションのいずれかからのマルチサイト更新で、 DB2 (OS/390 版) を更新するために TCP/IP 接続性が使用されていて、 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回復は、 DB2 Syncpoint Manager が関係する未確定トランザクションの回復とは異なります。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。アラートについての詳細は、*問題判別の手引き* を参照してください。

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・サーバーで発見的手法による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. OS/390 ホストで、DISPLAY THREAD TYPE(INDOUBT) コマンドを出します。

   このリストから、発見的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 (OS/390 版) コマンド解説書* 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 コマンドの詳細については、 *DB2 (OS/390 版) コマンド解説書* を参照してください。

災害時回復

災害時回復 という用語は、火災、地震、暴力行為、または他の災害事象が発生した場合にデータベースを復元するために、実行しなければならない活動を記述するために使用されます。災害時回復の計画には、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
- データベースを回復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

災害時回復計画が別のマシンでのデータベース全体の回復を意味する場合は、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、災害発生後にだけ復元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復元することは通常は不可能です。

災害時回復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、災害時回復ではデータベース全体を復元する必要があるので、待機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、データベースを復元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回復に使用できます。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復元できます。詳細については、113ページの『復元操作 (リダイレクト復元) 時の表スペース・コンテナーの再定義』を参照してください。

表スペースのバックアップと完全なデータベースのバックアップの両方があれば、どのような災害時回復計画でも役に立ちます。バックアップ、復元、およびデータのロールフォワードのために用意されている DB2 機能は、災害時回復計画の基本となります。業務を保護するために、準備した回復手順をテストしておくようにしてください。

バージョン回復

バージョン回復 は、以前のバージョンのデータベースの回復であり、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。この回復方式は、回復不能データベース (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、回復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。データベース復元操作では、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復元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 (23ページの図3 を参照)。

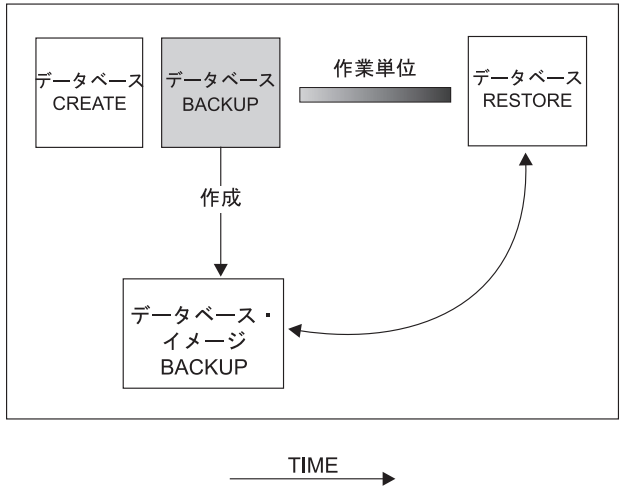


図 3. バージョン回復. データベースは最新のバックアップ・イメージから復元されますが、バックアップを取ってから障害が起きるまでの間に処理された作業単位はすべて失われます。

バージョン回復方式を使用して、データベースの完全な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。

区分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区画サーバー (またはノード) にまたがって存在します。すべての区画を復元することと、データベース復元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。(各データベース区画は、別々にバックアップされ、復元されます。) 同時に作成される各データベース区画のバックアップは、バージョン・バックアップ と呼ばれます。

ロールフォワード回復

ロールフォワード回復 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、 *logretain* または *userexit* データベース構成パラメーターのいずれか (またはその両方) を使用可能にすることで実行できます。使用するロギング手順について詳しくは、30ページの『リカバリー・ログについて』を参照してください。) データベースを復元して、 WITHOUT ROLLING FORWARD オプションを指定することは、バージョン回復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復元されます。データベースを復元する際、データベース復元操作で 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復元操作が終了するまでロールフォワード保留状態になります。これで、ロールフォワード回復を行なえるようになります。

考慮する 2 つのロールフォワード回復は次のとおりです。

- *データベースのロールフォワード回復*。このタイプのロールフォワード回復では、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復元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) の状態に回復されます。

  区分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区画にまたがって存在します。時刻指定ロールフォワード回復を実行する場合は、すべてのデータベース区画をロールフォワードし、すべての区画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区画を復元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回復を実行して、その区画をデータベース内の他の区画と同じレベルにすることができます。 1 つのデータベース区画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回復だけを実行できます。時刻指定回復の適用対象は、すべての データベース区画です。

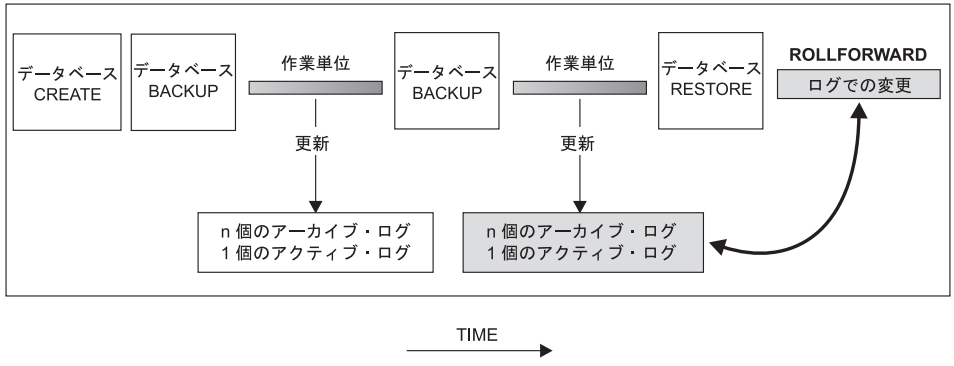


図 4. データベースのロールフォワード回復. 長時間実行しているトランザクションの場合には、複数のアクティブ・ログが生じる可能性があります。

- *表スペースのロールフォワード回復*。データベースの順方向回復が可能であれば、表スペースのバックアップ、復元、およびロールフォワードも可能です (25ページの図5 を参照)。表スペースの復元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、回復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 (*時刻指定* 回復と呼ばれる)。

表スペース・ロールフォワード回復は、以下の 2 つの場合に使用されます。

- 表スペース復元操作の後、表スペースは常にロールフォワード保留状態で、ロールフォワードする必要があります。 ROLLFORWARD DATABASE コマンド (150ページの『ROLLFORWARD DATABASE コマンド』 を参照) 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。

- 破損回復の後で 1 つまたは複数の表スペースが*ロールフォワード保留* 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復元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保留状態になります。この場合にはデータベース復元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。破損回復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破損回復で十分です。

  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 表スペースを復元した後、ログの終わりまでロールフォワード回復を実行する必要があります。

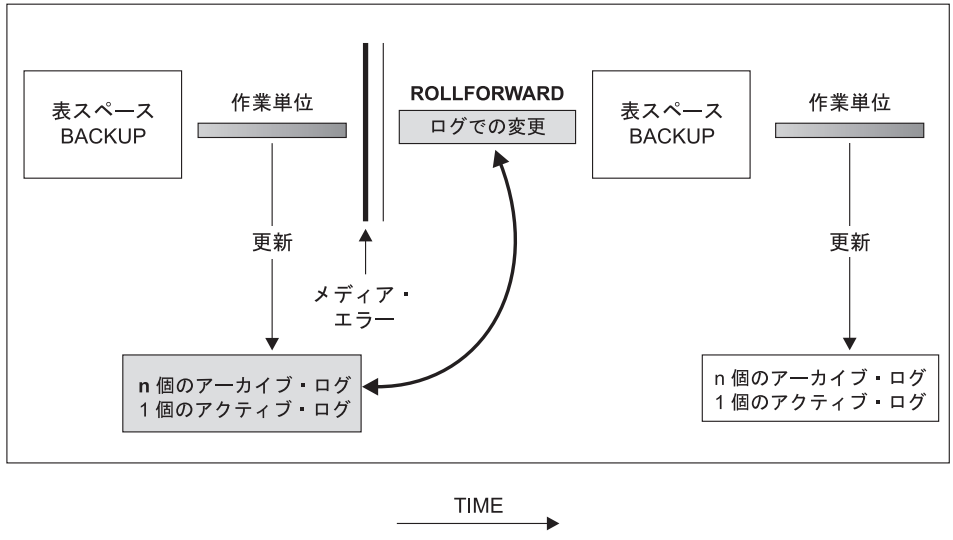


図 5. 表スペースのロールフォワード回復. 長時間実行しているトランザクションの場合には、複数のアクティブ・ログが生じる可能性があります。

区分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 (データベース区画) のリストを指定する必要はありません。 DB2 は、すべての区画にロールフォワード要求を実行依頼します。これは、表スペースが常駐するすべてのデータベース区画で表スペースを復元しなければならないことを意味します。

区分データベース・システムでは、*ログの最後に向けて* 表スペースをロールフォワードする場合で、すべての区画で表スペースをロールフォワードしたくない 場合は、データベース区画のリストを提供する必要があります。 (すべての区画上にある) ロールフォワード保留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、データベース区画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区画に送信されます。

増分バックアップおよび回復

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回復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。

DB2 で増分バックアップおよび回復がサポートされるようになりました (ただし、長形式フィールドやラージ・オブジェクトのデータは含まれません)。*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、データベースのヒストリーなど) も含まれます。

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。
- *デルタ (差分)*。デルタ、または増分デルタのバックアップ・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、またはデルタ) 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。デルタ・バックアップ・イメージの先行イメージは、そのデルタ・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。

増分とデルタのバックアップ・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。デルタ・バックアップ・イメージには、前回のバックアップ作成以降に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固有の集合になる可能性も生じるので、バックアップ戦略を計画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一貫性のある状態に作成し直す場合は、復元対象のオブジェクト (データベースまたは表スペース) 全体の一貫性のあるイメージを使用して回復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 (『増分バックアップ・イメージからの復元』を参照)。

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。
- 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、変更内容はインスタンス中の任意の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。増分バックアップを取るには、その前に全データベースのバックアップが必要になります。

既存のデータベースの場合は、デフォルトの *trackmod* の設定値は `NO` です。新しいデータベースの場合は `YES` です。

SMS 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。 DMS 表スペースの場合は、データ・ページや索引ページの場合はエクステント・レベルになり、それ以外のタイプのページの場合は表スペース・レベルになります。

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。

増分バックアップ・イメージからの復元

増分バックアップ・イメージからの復元操作は、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。

   DBA は、まず復元する最終イメージを決定し、 DB2 復元ユーティリティーによる増分復元操作を要求しなければなりません。このイメージは、復元される最終イメージになるので、増分復元のターゲット・イメージとして認識されます。このイメージに対して増分復元コマンドを実行すると、このターゲット・イメージ中の構成と表ス

ベースの定義を使用して、新しいデータベースの作成が開始されます。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復元して、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 で復元したベースライン・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、作成順に復元します。
4. ステップ 1 のターゲット・イメージが 2 度めに読み取られるまで、ステップ 3 を繰り返します。増分復元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。

   復元操作時に作成されるデータベースが、確実に正しいヒストリー、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、増分復元操作のターゲット・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除去された場合、増分復元処理の際に、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・イメージの集合を復元するには、 RESTORE DATABASE コマンドに TAKEN AT *timestamp* オプションを指定してください。復元したい最終イメージのタイム・スタンプを指定してください。たとえば、次のようにします。

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

この場合、DB2 復元ユーティリティーによって、上記の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、復元ユーティリティーによってデータベースとそのヒストリー、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・ヒストリーが照会され、要求された復元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、復元操作は終了し、エラー・メッセージが戻されます。このような場合は、自動復元は行えないので、手動復元の手順を使用して処理を続ける必要があります。

注: PRUNE HISTORY コマンドに 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・イメージから回復する際に必要になることがあるヒストリー項目は削除されませんが、 FORCE オプションを指定すると、自動復元操作に必要な項目を削除できるようになります。

データベース・ヒストリーが使用できない場合は、この項の最初で概説されているステップに従って、手動で増分復元操作を実行できます。たとえば、次のようにします。

1. db2 により、<ts> からデータベース・サンプル増分が復元されます。

   ただし、
   <ts> は復元する最終増分バックアップ・イメージを指します。

2. db2 により、<ts1> からデータベース・サンプル増分が復元されます。

   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 により、<tsX> からデータベース・サンプル増分が復元されます。

   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. ステップ 3 を繰り返して、
   イメージ <ts> まで個々の増分バックアップ・イメージを復元します。

表スペースの増分バックアップ・イメージが作成されている場合に、データベースの復元操作を試行するには、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復元しなければなりません。

自動増分復元の制限

1. 復元元にするバックアップを取った後で表スペースの名前を変更し、新しい名前を使用して表スペース・レベルの復元を実行すると、データベース・ヒストリーを使用した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。

   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>
   ```

   推奨されている対処策: 手動増分復元を使用してください。

2. データベースを除去すると、データベース・ヒストリーは削除されます。除去したデータベースを復元すると、データベース・ヒストリーは、復元元のバックアップが取られたときの状態に復元され、それ以降のヒストリー項目はすべて失われます。その後、失われたヒストリー項目を使用する必要のある自動増分復元を試行すると、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復元が試行され、 ″out of sequence″ エラーが戻されます。

   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:

- 手動増分復元を使用してください。
- 最初にイメージ <ts4> からヒストリー・ファイルを復元してから、自動増分復元を実行してください。

リカバリー・ログについて

すべてのデータベースには、それに伴うログがあります。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復元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 ログには 3 つのタイプがあります。*循環*、キャプチャー、およびアーカイブ であり、それぞれは、異なるレベルの回復機能を提供します。

- *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。(*logretain* データベース構成パラメーターの設定値は NO です。) 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有効です。名前から分かるとおり、循環ログはオンライン・ログの 『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回復を提供します。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。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。このタイプの復元操作では、全バックアップが取られた時点までデータが回復されるため、これは、*バージョン回復* と呼ばれています。

  31ページの図6 は、循環ロギングが活動状態のときにアクティブ・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。

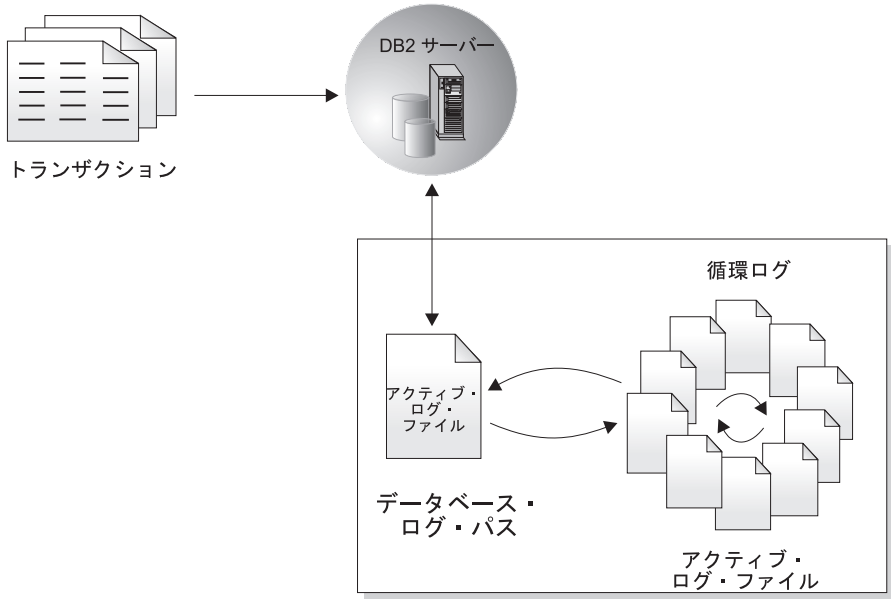


図 6. 循環ロギング

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一貫性のない状態になるのを防ぐために、破損回復処理中に使用されます。 RESTART DATABASE コマンドは必要に応じてアクティブ・ログを使用し、データベースを、一貫性のある使用可能な状態にします。破損回復の際、まだコミットされていないログ内の変更内容は、ロールバックされます。コミットされてはいるもののメモリー (バッファー・プール) からディスク (データベース・コンテナー) へまだ書き出されていない変更は、再実行されます。こうした処置によって、データベースの保全性が保たれます。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。

- キャプチャー・ログは、 *logretain* データベース構成パラメーターを CAPTURE に設定して構成できます。キャプチャー・ログは複製処理に使用されます。ログ・ファイルは、複製処理が完了するまで保存され、完了すると自動的に削除されます。このロギング・モードは、DB2 ユーティリティーにより循環ログと同じ方法で処理されます。つまり、オンライン・バックアップ操作、表スペースのバックアップおよび復元操作、およびロールフォワード操作のいずれも使用できません。また、RECOVERY NO オプションを指定してロード操作を行うと、表スペースはバックアップ保留状態になりません。
- アーカイブ・ログ はロールフォワード回復に使用されます。このログは、 *logretain* データベース構成パラメーターを RECOVERY に設定して構成できます。アーカイブ・ログには次のものがあります。

オンライン・アーカイブ・ログ

アクティブ・ログに入っている変更内容が通常の処理で必要なくなると、そのログはクローズされて、アーカイブ・ログになります。アーカイブ・ログがデータベース・ログ・パス・ディレクトリーに入っている場合、そのログはオンライン であるといいます (図7 を参照)。

オフライン・アーカイブ・ログ

アーカイブ・ログがデータベース・ログ・パス・ディレクトリーに入っていない場合、そのログはオフライン であるといいます (33ページの図8 を参照)。ユーザー出口プログラムを使用して、アーカイブ・ログをデータベース・ログ・パス・ディレクトリー以外の位置に保管することもできます。 (追加情報については、457ページの『付録H. データベース回復用のユーザー出口』を参照してください。)

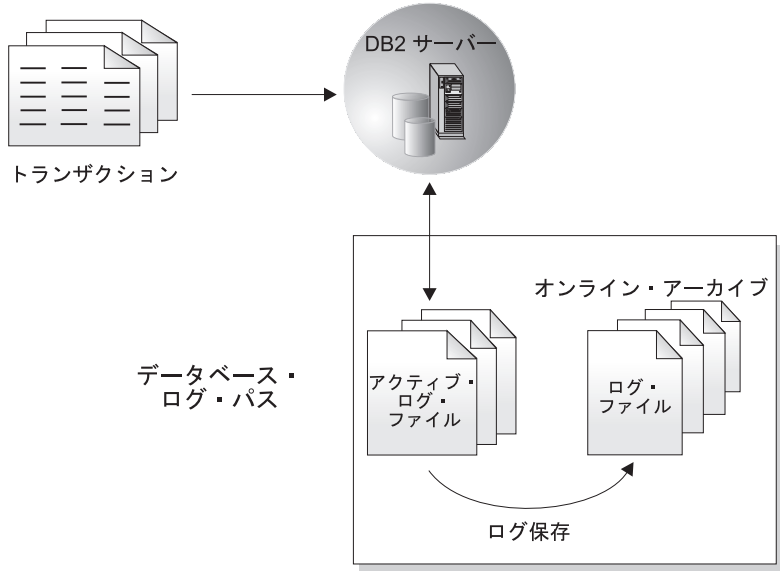


図7. オンライン・アーカイブ・ログ

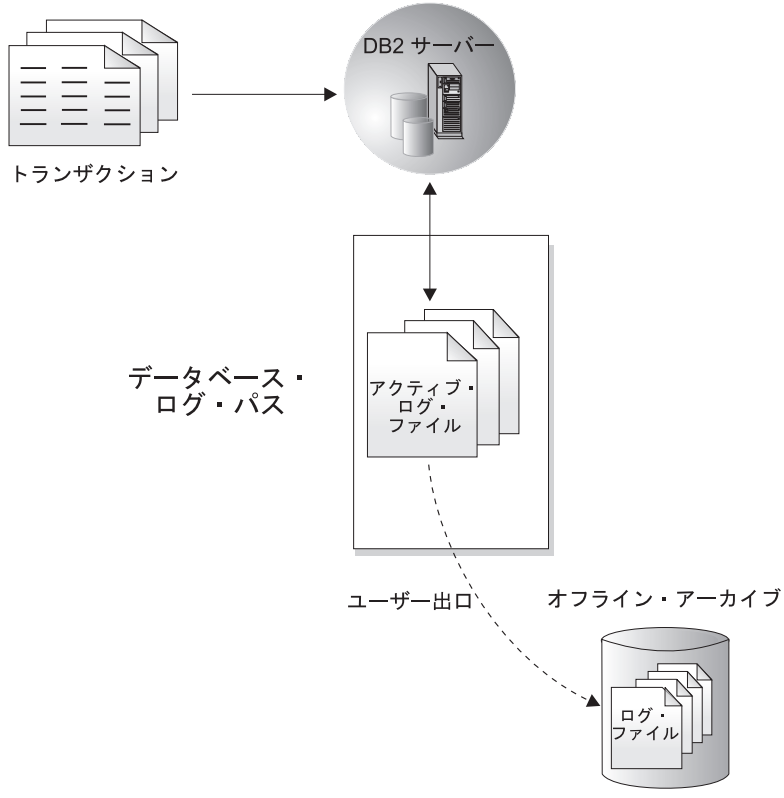


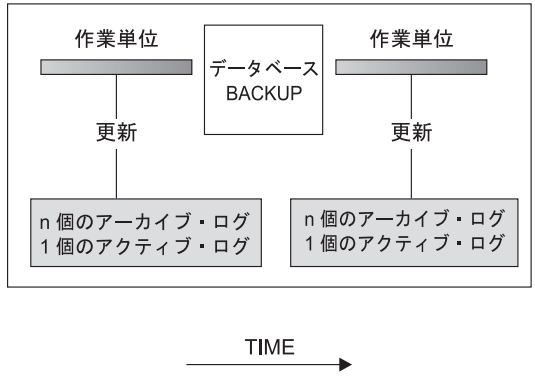
図 8. オフライン・アーカイブ・ログ

ロールフォワード回復では、アーカイブ・ログとアクティブ・ログの両方を使用して、ログの終わりまで、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ます。ロールフォワード・ユーティリティーは、アーカイブ・ログとアクティブ・ログで検出されるコミットされた変更項目を、復元されるデータベースに再適用することで、この処理を実行します。

ロールフォワード回復は、アーカイブ・ログとアクティブ・ログの両方にあるコミットされた更新を再適用することにより、ログを使用して表スペースを再構築することもできます。表スペースは、ログの最後まで、または特定の時点まで回復可能です。

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・イメージが復元されるとき、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、データベースが復元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、回復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 9. ロールフォワード回復でのアクティブ・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* と *userexit* の 2 つ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。これらの構成パラメーターについての詳細は、*管理の手引き: パフォーマンス* を参照してください。

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* よりも小さい順序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検査できます。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、*管理の手引き: パフォーマンス* を参照してください。

注:

1. アクティブ・ログを消去すると、データベースは使用不能になってしまい、再び使用できるようにするには復元することが必要になります。また、消去された最初のログまでしかロールフォワードできません。
2. 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、ログに使用するディスクをミラーリングしてオペレーティング・システム・レベルのミラーリングを行うか、 NEWLOGPATH2 レジストリー変数を使用して DB2 レベルのミラーリングを行うことを考慮してください。

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、新しい DB2 レジストリー変数 NEWLOGPATH2 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。

NEWLOGPATH2 レジストリー変数を使用すると、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・コントローラーが単一の障害点になることはありません。

注: Windows NT では、任意のパス名で装置を「マウントする」ことができないので、(このプラットフォーム上で) 別個の装置上に 2 次パスを指定することはできません。

NEWLOGPATH2 は、使用可能にしたり (1 に設定)、使用不可にしたり (0 に設定) できます。デフォルト値はゼロです。この変数を 1 に設定すると、2 次パス名は、LOGPATH 変数の現行値に文字 2 を連結した値になります。たとえば、SMP 環境で、LOGPATH が /u/dbuser/sqllogdir/logpath の場合は、 2 次ログ・パスは /u/dbuser/sqllogdir/logpath2 になります。 MPP 環境で、LOGPATH が /u/dbuser/sqllogdir/logpath の場合は、 DB2 ではこのパスにノード標識が付加され、 1 次ログ・パスとして /u/dbuser/sqllogdir/logpath/NODE0000 が使用されます。この場合、2 次ログ・パスは /u/dbuser/sqllogdir/logpath2/NODE0000 になります。

NEWLOGPATH2 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に現行のログ・ファイルが完了するまで使用されません。これは、LOGPATH や NEWLOGPATH の現在の使用方法と同様です。

パス 1 または 2 への書き込みエラーが生じると、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、 db2diag.log ファイル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、データベースは異常終了します。

作業表におけるロギングの低減

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回復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。 (そのためには、NOT LOGGED INITIALLY オプションを指定して表を作成していなければなりません。)

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 パラメーター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。
- 表が作成される作業単位の操作についてエラーが戻されると、作業単位全体がロールバックされます (SQLCODE -1476、SQLSTATE 40506)。
- ロールフォワード時には、これらの表を回復できません。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された表が検出されると、この表には使用不能のマークが付けられます。データベースの回復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、並行操作を意味します。詳細については、*管理の手引き: パフォーマンス* の『並行性』の項を参照してください。

表の作成についての詳細は、*SQL 解説書* を参照してください。

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除去されるときに、宣言済み一時表は暗黙的に除去されます。
- 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。

宣言済み一時表とその制限についての詳細は、 *SQL 解説書* で DECLARE GLOBAL TEMPORARY TABLE ステートメントを参照してください。

データベース・ロギングの構成パラメーター

データベース構成ファイルには、ロールフォワード回復に関連するパラメーターが含まれています。これらのパラメーターのデフォルト値ではこのタイプの回復がサポートされていないので、この方法を使用する予定の場合は、デフォルト値をいくつか変更しなければなりません。 DB2 ユニバーサル・データベースの構成についての詳細は、*管理の手引き: パフォーマンス* を参照してください。

1 次ログ (logprimary)

このパラメーターには、作成する 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。

既存データベースでロールフォワード回復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。ロールフォワード回復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。

ログ・ファイルの合計サイズの限界は、32 GB です。つまり、ログ・ファイルの数 (*logprimary* + *logsecond*) に、各ログ・ファイルのサイズ (バイト単位) (*logfilsiz* * 4096) を掛けた値が、 32 GB より小さくなければなりません。

この構成パラメーターについての詳細は、*管理の手引き: パフォーマンス* を参照してください。

2 次ログ (logsecond)

このパラメーターは、必要に応じて回復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。構成された数より多くの 2 次ログ・ファイルが必要になると、エラーが戻され、データベースに対するアクティビティーは停止されます。

この構成パラメーターについての詳細は、*管理の手引き: パフォーマンス* を参照してください。

ログ・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 32 GB です。この値は、 *logfilsiz* の上限 (65535) と *logprimary* + *logsecond* の上限 (128) の計算結果です。つまり、 ((*logprimary* + *logsecond*) * *logfilsiz*) < (32 GB / 4096) 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。データベースの構成がログを保持するようになっている場合は、ログが満杯になるたびに、新規ログの割り振りと初期化を求める要求が出されます。ログのサイズを大きくすると、新規ログの割り振りと初期化に必要な要求の回数は少なくなります。 (ただし、ログ・サイズを大きくすれば、新規ログの形式化処理にかかる時間が長くなります。) 新規ログの形式化処理はデータベースに接続しているアプリケーションにとって透過的なので、形式化処理はデータベースのパフォーマンスに影響しません。

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 (55ページの『回復パフォーマンスの向上』を参照) 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。

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、ディスク・スペースの空き容量によっては、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。そうなると、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。 (ログを保管するために異なるメディアを使用する場合については、 457ページの『付録H. データベース回復用のユーザー出口』を参照してください。) 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、作業負荷がピークの時の 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。

ログ・ファイルの損失を最小限にくい止めることも、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・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、データベース共有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。

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。

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回復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ーカイブ・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、ロールフォワード回復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、データベースが一貫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についての詳細は、*管理の手引き: パフォーマンス* を参照してください。データベースが一貫した状態になっていない場合にデータベース・ログが果たす役割については、 41ページの『ログ・ファイル管理』を参照してください。

ログ保持 (logretain)

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、データベースは回復可能と見なされます。つまり、ロールフォワード回復を使用できるようになります。

logretain を CAPTURE に設定すると、複製収集プログラムによって PRUNE LOGFILE コマンドが呼び出され、収集プログラムの完了時にログ・ファイルが削除されます。データベースのロールフォワード回復を実行したい場合は、logretain を CAPTURE に設定しないでください。

ユーザー出口 (userexit)

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、ロールフォワード回復を使用できるようになります。ユーザー出口プログラムについては、457ページの『付録H. データベース回復用のユーザー出口』を参照してください。

ログ・ファイル管理

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回復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回復不可 に変更された場合
  - S9999999.LOG が使用された場合

  DB2 では、データベース復元後、ロールフォワード回復を実行してもしなくても、ログ名を再使用します。データベース・マネージャーでは、ロールフォワード回復時に誤ったログが適用されないようになっていますが、必要なログの位置を検出することはできません。ロールフォワード回復時には、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、ロールフォワード回復に使用された最後のログより順序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。ユーザー出口プログラムについては、457ページの『付録H. データベース回復用のユーザー出口』を参照してください。)
- 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 (42ページの図10を参照)。たとえば、Backup 2 を回復したい場合、使用できるログ・シーケンスは 2 通りあります。全データベースの回復時に、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。

回復後に新規のログ・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。データベースの復元操作の直後に表スペースの復元操作を行うと、復元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復元操作を行うと、正常に完了することもありますが、表スペースのロールフォワード操作は失敗し、表スペースはロールフォワード保留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 (図10 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 を使用して復元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復元とロールフォワード回復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 を復元してから、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 の復元が終了すると、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。

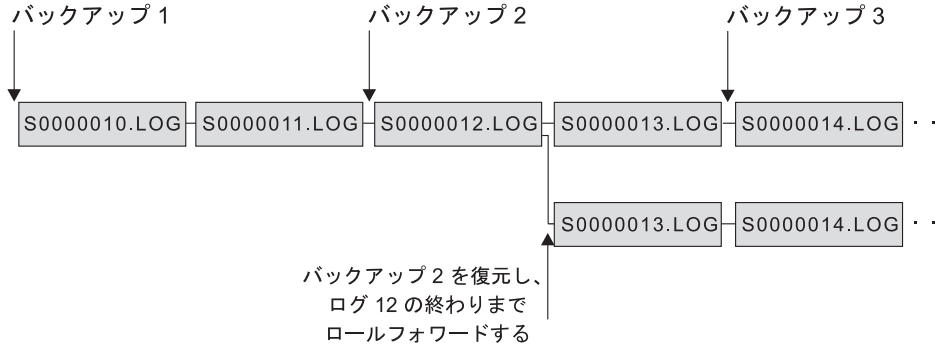


図 10. ログ・ファイル名の再使用

ユーザー出口プログラムを使用したログ・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。

- データベース構成ファイル・パラメーター *userexit* は、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回復時にログ・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 NT 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、破損回復に必要なので、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て非アクティブになると 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優っています。
- DB2 は、破損回復またはロールバックの際に、ログ・ファイルの検索をするために、ユーザー出口プログラムを呼び出しません。
- ユーザー出口プログラムは障害の時点までロールフォワード回復することを保証しませんが、障害ウィンドウを小さくしようとすることだけはします。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、ユーザー出口に直接影響します。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、比較的大型のログ・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。キューが満杯になると、アーカイブと検索の要求は処理されません。キューに余地があれば、処理が再開します。処理されない要求が自動的に再びキューに入れられることはありません。
- ユーザー出口プログラムに対するアーカイブ要求は、 *userexit* が構成されたときに限り、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、ロールフォワード回復でオフライン・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。
- 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検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、特定の時点で回復した後で、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、正しい回復パスと関連付けるように作成する必要があります。(41ページの『ログ・ファイル管理』 を参照)
- 2 つ以上のデータベースが同時に 1 つの装置を使用しており、その操作の 1 つがロールフォワード操作に関係している場合、ロールフォワード回復に必要なログ・ファイルは、現在ドライブにあるメディアに入っている必要はありません。次の 2 つの条件が起きる可能性があります。
  - ユーザー出口プログラムがゼロ (正常実行) の戻りコードをデータベース・マネージャーに戻し、要求されたログ・ファイルが検索されていない場合、データベー

ス・マネージャーは、ロールフォワード操作がログの末尾まで完了し、停止したものと見なします。しかし、ロールフォワード処理は、ログの末尾に達していないこともあります。

– ゼロ以外の戻りコードが戻される場合、データベースはロールフォワード操作保留状態となり、ユーザーはロールフォワード処理の再開または停止のいずれかを行う必要があります。

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。

ログ・ディレクトリーが満杯の場合のトランザクションのブロック化

新しい DB2 レジストリー変数 DB2_BLOCK_ON_LOG_DISK_FULL を設定して、 DB2 で新しいログ・ファイルをアクティブ・ログ・パス中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。 *userexit* パラメーターを ON に設定してデータベースが構成されている場合でも、DB2 によりログ・ファイルのアーカイブが完了したか検査されます。非アクティブのログ・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。試行されるたびに、メッセージが db2diag.log ファイルに書き込まれます。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、 db2diag.log ファイル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。

オンデマンドのログのアーカイブ

DB2 では、いつでも回復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る (さらに、ユーザー出口オプションを使用できる場合はアーカイブもできる) ようになりました。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待機データベースを更新できるようになりました。

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。これらについては、それぞれ 328ページの『ARCHIVE LOG』および 342ページの『db2ArchiveLog - アクティブ・ログのアーカイブ API』 で説明されています。

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接続できる。
  - 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。作業負荷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。ロー・デバイスにより使用される構文の例については、 *管理の手引き: インプリメンテーション* の『ロー I/O』を参照してください。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。
- 循環ロ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。
- ユーザー出口プログラムをもたないログ保存 (*logretain*) を使用する場合、*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、回復性を保全します。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、データベースの復元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、データベースを復元してロールフォワードできます。
- ユーザー出口プログラムをもつログ保存 (*logretain*) を使用する場合、ログ・レコードを入れられるログ・エクステントごとにユーザー出口プログラムが呼び出されます。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しなければなりません。 DB2 は、ロー・デバイスに対してログ・ファイルを検索するためのユーザー出口プログラムを呼び出しません。しかし、ロールフォワード回復中にエクステント・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、ユーザー出口プログラムを呼び出し、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。ロールフォワード操作にオーバーフロー・ログ・パスを指定しなければ、ログ・ファイルを検索するために DB2 がユーザー出口プログラムを呼び出すことはありません。ユーザー出口プログラムの呼び出しについての詳細は、 460ページの『呼び出し形式』を参照してください。
- DPROP を使ってロー・デバイスにログを書き込まなければ、読み取りログ API はログ・ファイルを検索するためのユーザー出口プログラムを呼び出しません。しかし、

要求されたログ・レコードを装置上で使用できる場合、そのレコードは依然として戻されます。装置上で一番古いログよりも新しいログを要求しても、そのログは戻されません。 (これは、要求ログ・レコードが入ったログ・ファイルを見つけられない DB2 の動作に似ています)。

注:

1. ログ記録にロー・デバイスを使用しているときは、DPROP を使用しないことをお勧めします。
2. **sqlurlog** API を使用する場合は、ログ記録にロー・デバイスを使用してはなりません。

ログの消失

データベースを除去すると、現在のデータベース・ログ・パス・ディレクトリーに入っているすべてのログが消去されます。データベースを除去する前に、それらのログのコピーを作成することを考慮してください。

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、最後に使われたログと、その後のすべての既存ログが再利用されます。特定の時点を過ぎると、回復することはできなくなります。特定時点までの回復操作が始まる*前*に、現在のデータベース・ログ・パス・ディレクトリーに入っているすべてのログをコピーしてください。

ロールフォワード操作が完了すると、最後にコミットされたトランザクションを含むログ・ファイルは切り捨てられ、その次の順序のログからロギングが開始されます。切り捨てられたログとそれより大きい順序番号のログのコピーがないなら、データベースの指定時点以降の部分を回復することはできません。 (通常のデータベース・アクティビティーが再開されると、新規のログが作成され、以後の回復操作で使用できます。)

ログ・パス・ディレクトリーを変更した後で、サブディレクトリーを除去した場合、またはそのログ・パス上のサブディレクトリーのログをすべて消去した場合、データベース・マネージャーはデータベースのオープン時にデフォルトのログ・パス SQLOGDIR のログを検索します。ログが見つからないと、データベースはバックアップ保留状態になるので、データベースが使用可能状態になる前に、このデータベースのバックアップをとっておく必要があります。このバックアップは、サブディレクトリーに入っているログが空であっても必ず作成してください。

オンライン・バックアップ終了時点が記録されているログが失われた場合に、復元後の対応するイメージをロールフォワードすると、データベースは使用できなくなります。データベースを使用可能にするためには、異なったバックアップおよび関連するすべてのログからデータベースを復元する必要があります。

特定時点までの回復を全データベースに対して実行したいが、回復中にログを失う可能性がある場合があるかもしれません。 (このようなことは、バックアップ・データベース・イメージの最後のものと、データベースの回復目標時点との間にあるアーカイブ・ログ数が膨大である場合などに生じます。)

まず、適用可能なすべてのログを『安全な』位置にコピーする必要があります。その後、RESTORE コマンドで、ロールフォワード回復方式を使用して、データベースを特定の時点のものにすることができます。必要なログがこのプロセス中に壊れたり失われたりした場合は、他のところに移しておいたすべてのログのバックアップ・コピーを使います。

リカバリー・ヒストリー・ファイルについて

リカバリー・ヒストリー・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復元
- 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
- 表スペースの除去
- 表のロード
- 表の除去
- 表の再編成
- 表統計の更新

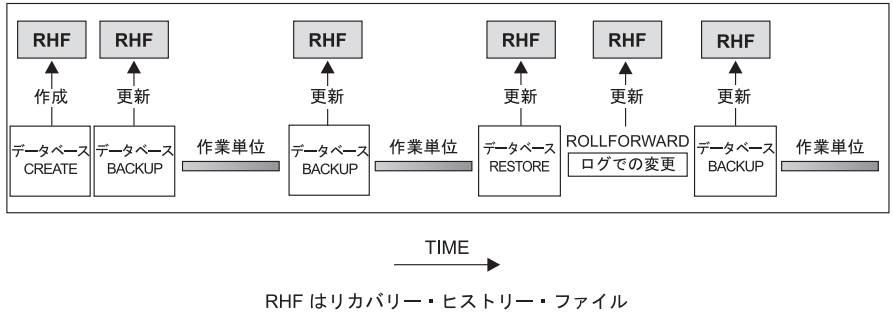


図 11. リカバリー・ヒストリー・ファイルの作成と更新

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回復できます。このファイルには、以下のような情報が含まれています。

- それぞれのエントリー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)
- 復元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回復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・ヒストリー・ファイル内の項目を参照するには、 LIST HISTORY コマンドを使用してください。このコマンドの詳細については、331ページの『LIST HISTORY』 を参照してください。

注: ログの末尾まで復元操作を実行してからロールフォワード操作を実行する場合、 LIST HISTORY コマンドの呼び出し後に示されるバックアップ ID は、最終時刻を表します。つまり、バックアップ ID 値は 99991231235959 です。ロールフォワードを実行する場合、バックアップ ID の変換方法はこれだけです。

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・ヒストリー・ファイルのコピーが含まれます。リカバリー・ヒストリー・ファイルは、データベースにリンクしています。データベースを削除すると、リカバリー・ヒストリー・ファイルも削除されます。データベースを新規の位置に復元すると、リカバリー・ヒストリー・ファイルも復元されます。復元の際に、既存のリカバリー・ヒストリー・ファイルは上書きされません。

現在のデータベースが使用不能で、関連するリカバリー・ヒストリー・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・ヒストリー・ファイルだけを復元できます。その後、そのリカバリー・ヒストリー・ファイルを調べて、データベースの復元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 (とその復元セット) 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プルーニングが必要になります。この構成パラメーターについての詳細は、*管理の手引き: パフォーマンス* を参照してください。

ガーベッジ・コレクション

PRUNE HISTORY コマンドを使用して (334ページの『PRUNE HISTORY/LOGFILE』を参照) ヒストリー・ファイルからエントリーを除去できますが、この種の枝取りは DB2 に任せることをお勧めします。リカバリー・ヒストリー・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。 DB2 ガーベッジ・コレクションは、データベースのバックアップ操作が正常に完了した後に呼び出されます。また、データベースの復元操作が正常に完了した後にも呼び出されます。構成パラメーター *num_db_backups* は、保持されるアクティブな全 (増分ではない) データベース・バックアップ・イメージの数を定義します。ヒストリー・ファイルを、最後のエントリーから始めて走査するために、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、ヒストリー・ファイルから有効期限切れのエントリーが枝取りされます。

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回復するように現行のログを使用して復元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、復元の際に、データベースを直前の状態に戻します。

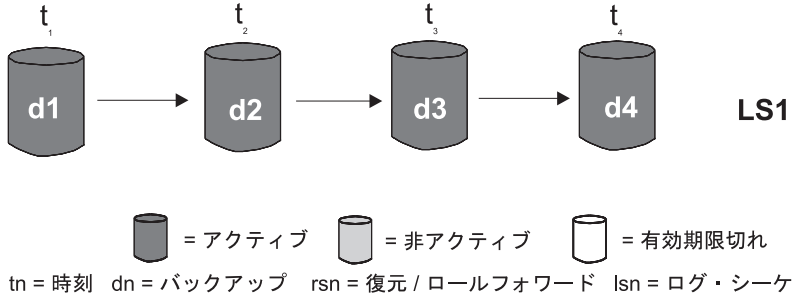


図 12. アクティブ・データベース・バックアップ. num_db_backups の値は 4 に設定されています。

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。

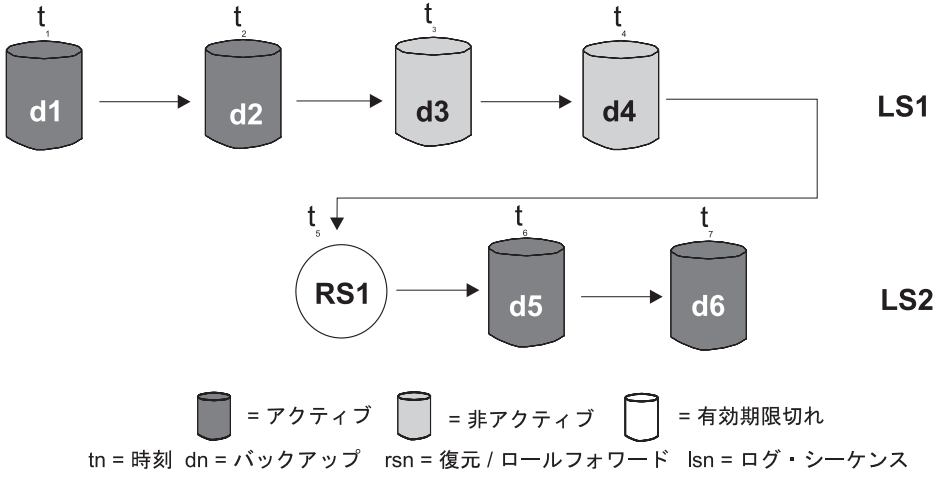


図 13. 非アクティブ・データベース・バックアップ

アクティブのデータベース・バックアップ・イメージが復元されるものの、ヒストリー・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・イメージが復元された場合、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。

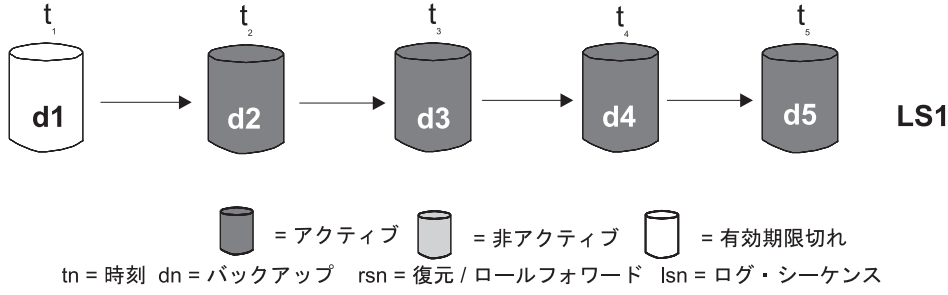


図 14. 有効期限が切れたデータベース・バックアップ

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・イメージのヒストリー・ファイル・エントリーに、『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、復元された 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・イメージが復元されると、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。復元されたイメージは、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復元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・イメージを復元する必要が生じることが考えられます。)

表スペースを復元した後で、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 データ・リンク・マネージャー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。

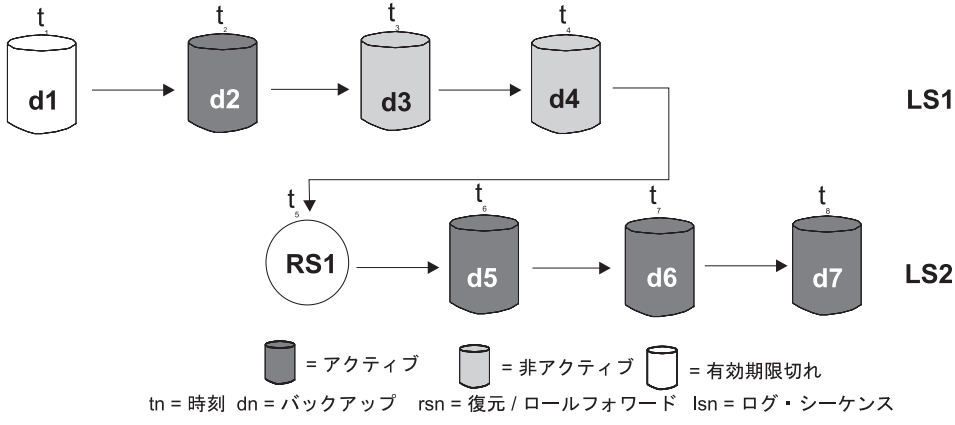


図 15. アクティブ、非アクティブ、有効期限切れのデータベース・バックアップの混合

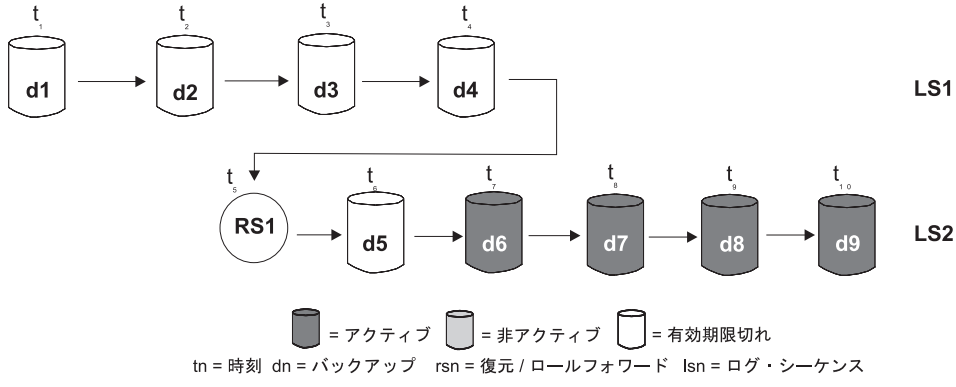


図 16. 有効期限が切れたログ・シーケンス

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。回復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。

- *ロールフォワード保留状態*。表スペースの復元後、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。表スペースの復元後の場合は、ログの最後まで、または特定の時刻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。
- *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。

- *復元保留状態*。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回復不能エラーが生じた場合 (この場合は、表スペースの復元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。
- *バックアップ保留状態*。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。)

回復パフォーマンスの向上

回復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。
- 復元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。
  - 復元バッファー・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。
  - バッファー数を増加させる。

    複数のバッファーと入出力チャネルを使用する場合、少なくともチャネルの 2 倍のバッファーを使用し、チャネル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、復元操作のパフォーマンスに影響を与えます。理想的な復元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、メディア装置またはコンテナーの数に、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。
  - 復元操作の 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、それらの復元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回復可能にしておくと、 RESTORE コマンドでは特定の表スペースを復元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復元するときは、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復元を選択しないことで、復元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復元しないが、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一貫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、その表スペースの時刻指定ロールフォワード回復は実行できません。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復元操作には、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。

並列回復

DB2 では、複数のエージェントを使用して、破損回復とデータベースのロールフォワード回復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。データベースを回復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。

今回の拡張で導入された新しい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、データベースの回復に使用するエージェントの数が選択されます。 SMP マシンの場合、使用されるエージェントの数は、CPU の数 + 1 です。 CPU が 1 つのマシンでは、ログの読み取り、ログ・レコードの処理、およびデータ・ページの事前取り出しの効率を高めるために複数のエージェントが使用されます。

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。

DB2 データ・リンク・マネージャーについての考慮事項

以下の節では、DATALINK 列を含む表が存在する場合に適用される情報が記載されています。 DATALINK 列の完全な説明については、 *SQL 解説書* の CREATE TABLE ステートメントを参照してください。

破損回復の考慮事項

アプリケーションが、 DB2 データ・リンク・マネージャーを実行しているデータ・リンク・サーバーに関連する SQL 要求を出すと (FILE LINK CONTROL 属性で DATALINK 列を使用)、データベース・マネージャーは、データ・リンク・サーバーに作業を分散します。また、どのデータ・リンク・サーバーがトランザクションに関係しているかの記録を保持します。アプリケーションがトランザクションの COMMIT を出すと、データベース・マネージャーは 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、データベース・マネージャーは PREPARE ログ・レコードを書き込み、すべてのデータ・リンク・サーバーに PREPARE 要求を配布します。それぞれのデータ・リンク・サーバーは次のいずれかで応答します。

- YES; データ・リンク・サーバーでコミットの準備ができていることを示す。
- NO; エラーが発生したため、データ・リンク・サーバーはコミットの準備ができていない。

最初のフェーズでは、すべてのデータ・リンク・サーバーが 『YES』と応答すると、成功したものと見なされます。

2 番目のフェーズでの処理は、1 番目のフェーズの結果によって異なります。少なくとも 1 つのデータ・リンク・サーバーが 『NO』と応答した場合、データベース・マネージャーは、関連するすべてのデータ・リンク・サーバーに ABORT 要求を配布します。トランザクションはロールバックされ、戻りコード 『03』 と共にエラー・メッセージ SQL0903N がアプリケーションに戻されます。そうでない場合には、データベース・マネージャーは、データ・リンク・サーバーの関与がない場合に通常行うとおりに、トランザクションのコミットを継続します。この処理の最後に、トランザクションに関係するすべてのデータ・リンク・サーバーに COMMIT 要求を配布します。

何らかのトランザクションを PREPARED 状態にしたままでデータ・リンク・サーバーに障害が起きた場合には、これらのトランザクションは*未確定トランザクション* と呼ばれます。データベース・マネージャーは、これらのトランザクションの成りゆきを追跡し、最終的にデータ・リンク・サーバーでこれらを随時解決します。障害によってデータ・リンク・サーバー上で未確定のトランザクションが作成された可能性があるとデータベース・マネージャーが判別する場合には、そのデータ・リンク・サーバーの状態

を、破損回復の必要なものとしてマークします。この状態であるうちは、どの SQL 要求もデータ・リンク・サーバーに関係することはできません。 SQL 要求を出したアプリケーションには、 SQL0357N が戻りコード 『03』 と共に戻されます。

RESTART、ACTIVATE DATABASE、または最初の CONNECT 処理では、データベース・マネージャーは、それぞれの構成されたデータ・リンク・サーバーに接続しようとし、それらを打ち切るか、コミットすることにより、この未確定トランザクションを解決しようとします。データベース・マネージャーでも未確定なトランザクションを除いて、すべての未確定トランザクションが解決されると、データ・リンク・サーバーの状態は、使用可能なものとしてマークされます。使用可能な状態では、データ・リンク・サーバーに関係する SQL 要求は許可されます。未確定トランザクションを解決しようと試みた後で、依然としてデータ・リンク・サーバーに解決の必要な未確定トランザクションがあるとデータベース・マネージャーが判別した場合は、そのデータ・リンク・サーバーの状態を、破損回復の必要なものとしてマークします。これは、たとえば、RESTART、ACTIVATE DATABASE、または最初の CONNECT 処理でデータ・リンク・サーバーが使用できない場合に起こります。あるいは、その処理の途中でデータ・リンク・サーバーに障害が発生しました。

データベースに構成されたデータ・リンク・サーバーが破損回復の必要な状態の場合には、データベース・マネージャーは、 SQL 要求がこのデータ・リンク・サーバーと関係することを許可しません。データベースの他のデータに関連する SQL 要求は依然として許可されます。データベース・マネージャーは、回復の必要なそれぞれのデータ・リンク・サーバーで非同期的に破損回復を実行しようとするプロセスを開始します。プロセスで破損回復が正常に完了すると、データ・リンク・サーバーの状態は使用可能としてマークされ、したがって、関連する SQL 要求を実行できるようになります。

バックアップ・ユーティリティーについての考慮事項

DB2 はバックアップ操作が完了するまでには、 DB2 データ・リンク・マネージャーを実行しているデータ・リンク・サーバーもバックアップされるようにします。 (バックアップ・ユーティリティーはオンラインでもオフラインでも実行でき、バックアップ・イメージにはデータベースまたは表スペースのいずれかを指定できます。) 後続する記述は、 DATALINK 列でリンクされていて RECOVERY パラメーターが YES に設定されているものだけに適用されます。 (DATALINK 列によって参照されているファイルで、 RECOVERY=NO が指定されていないものはバックアップされません。)

ファイルがリンクされると、データ・リンク・サーバーは、 TSM などのアーカイブ・サーバーまたはディスクに、それらが非同期でコピーされるように、スケジュールを組みます。バックアップ・ユーティリティーが実行されると、 DB2 はコピーされるようにスケジュールが組まれたすべてのファイルがコピーされるようにします。バックアップ処理の始めには、 DB2 は DB2 構成ファイルで指定されたすべてのデータ・リンク・サーバーに連絡します。データベース用に 1 つ以上のデータ・リンク・サーバーが構成されている場合、 1 つのデータ・リンク・サーバーが利用不能でもバックアップ操作は続けられます。データ・リンク・サーバーを再始動すると、データベースが再び使

用可能になる前に、そのデータ・リンク・サーバー上のバックアップ処理が完了します。データ・リンク・サーバーに 1 つ以上のリンク・ファイルが存在していて実行していない場合、またはバックアップ操作時に実行が停止している場合、バックアップ・イメージに完全な DATALINK が情報が含まれるようにはなりません。バックアップ操作は正常に完了します。バックアップ操作用のヒストリー・ファイル中のコメント欄では、障害が起こった DLM サーバーが識別されます。また、5 つ以上の DLM サーバーに障害が起きた場合は、障害が起きた DLM サーバーの数も識別されます。 DLM でバックアップが正常に実行されると、コメント欄は以前の値にリセットされます。

データ・リンク・サーバーがデータベースで使用可能としてマークが付けられるには、その前に、すべての未解決バックアップに関するバックアップ処理が正常に完了しなければなりません。 (この処理は非同期処理により自動的に行われます。) データ・リンク・サーバー上で、 *num_db_backups* 値 (下記参照) の 2 倍の数の未解決バックアップが完了待ちになっている時点でバックアップ操作が開始されると、バックアップ操作は失敗します。この場合、データ・リンク・サーバーを再始動しなければならず、未解決のバックアップが完了してからバックアップを追加できます。

ファイルのリンクが解除されると、 ON UNLINK パラメーターで指定された値に応じて、リンクは削除されるか、直前のファイルに戻されます。バックアップ操作が正常に実行されると、データ・リンク・サーバーは、アーカイブ・サーバー (ディスクまたは TSM のいずれか、 51ページの『ガーベッジ・コレクション』を参照) にあるファイルのアーカイブ版をクリーンアップすることができます。 *num_db_backups* データベース構成パラメーターは、ファイルのアーカイブ版 (リンク解除されたもの) が削除される前に実行される DB2 データベース・バックアップの数を指定します。この構成パラメーターについての詳細は、*管理の手引き: パフォーマンス* を参照してください。

リンクされていないファイルが削除されると、リンクされていないファイルについての情報もデータ・リンク・サーバー登録表から削除されます。

AIX 上で DB2 データ・リンク・マネージャーのバックアップ方式を選択する

ディスク・コピーや XBSA に加えて、 Tivoli Storage Manager (TSM) を使用してデータ・リンク・サーバー上のファイルのバックアップを取ることができます。

Tivoli Storage Manager をアーカイブ・サーバーとして使用するには、以下のようにします。

1. データ・リンク・サーバーに Tivoli Storage Manager をインストールします。詳しくは、Tivoli Storage Manager 製品の文書を参照してください。
2. データ・リンク・サーバー・クライアント・アプリケーションを Tivoli Storage Manager サーバーに登録します。詳しくは、Tivoli Storage Manager 製品の文書を参照してください。
3. 以下の環境変数を、データ・リンク・マネージャー管理者の `db2profile` または `db2cshrc` スクリプト・ファイルに追加します。

```
(Bash、Bourne、または Korn シェルの場合)
export DSMI_DIR=/usr/tivoli/tsm/client/api/bin
export DSMI_CONFIG=$HOME/tsm/dsm.opt
export DSMI_LOG=$HOME/dldump
export PATH=$PATH:$DSMI_DIR

(C シェルの場合)
setenv DSMI_DIR /usr/tivoli/tsm/client/api/bin
setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
setenv DSMI_LOG ${HOME}/dldump
setenv PATH=${PATH}:$DSMI_DIR
```

4. $DSMI_DIR ディレクトリー中に dsm.sys TSM システム・オプション・ファイルがあるか確認します。
5. *INSTHOME*/tsm ディレクトリー (*INSTHOME* はデータ・リンク・マネージャー管理者のホーム・ディレクトリー) に、 dsm.opt TSM ユーザー・オプション・ファイルがあるか確認します。
6. /usr/tivoli/tsm/client/api/bin/dsm.sys という Tivoli Storage Manager システム・オプション・ファイル中の *PASSWORDACCESS* オプションを generate に設定します。
7. 初めてデータ・リンク・ファイル・マネージャーを開始する*前*に、 TSM パスワードを generate オプションに登録します。この作業を行うと、データ・リンク・ファイル・マネージャーで TSM サーバーに対する接続が開始される際にパスワードを入力する必要がなくなります。詳しくは、TSM 製品の資料を参照してください。
8. DLFM_BACKUP_TARGET レジストリー変数を TSM に設定します。この場合は DLFM_BACKUP_DIR_NAME レジストリー変数の値は無視されます。これにより、Tivoli Storage Manager バックアップ・オプションがアクティブにされます。

   注:

   a. 実行時に DLFM_BACKUP_TARGET レジストリー変数の設定を TSM とディスクの間で切り替える場合は、アーカイブ済みのファイルは新しく指定したアーカイブ場所に移動されないことに注意する必要があります。たとえば、DLFM_BACKUP_TARGET レジストリー値を TSM に設定してデータ・リンク・ファイル・マネージャーを開始した場合に、このレジストリー値をディスク位置に変更すると、新しいアーカイブ・ファイルはすべてディスク上の新しい場所に保管されます。以前に TSM にアーカイブされたファイルは、新しいディスク位置に移動されません。

   b. デフォルトの TSM 管理クラスをオーバーライドするには、新しいレジストリー変数 DLFM_TSM_MGMTCLASS を使用します。このレジストリー変数を未設定にしておくと、デフォルトの TSM 管理クラスが使用されます。
9. **dlfm stop** コマンドを入力して、データ・リンク・ファイル・マネージャーを停止します。
10. **dlfm start** コマンドを入力して、データ・リンク・ファイル・マネージャーを開始します。

Solaris 実行環境で DB2 データ・リンク・マネージャーのバックアップ方式を選択する

ディスク・コピーや XBSA に加えて、 Tivoli Storage Manager (TSM) を使用してデータ・リンク・サーバー上のファイルのバックアップを取ることができます。

Tivoli Storage Manager をアーカイブ・サーバーとして使用するには、以下のようにします。

1. データ・リンク・サーバーに Tivoli Storage Manager をインストールします。詳しくは、Tivoli Storage Manager 製品の文書を参照してください。
2. データ・リンク・サーバー・クライアント・アプリケーションを Tivoli Storage Manager サーバーに登録します。詳しくは、Tivoli Storage Manager 製品の文書を参照してください。
3. 以下の環境変数を、データ・リンク・マネージャー管理者の `db2profile` または `db2cshrc` スクリプト・ファイルに追加します。

   ```
   (Bash、Bourne、または Korn シェルの場合)
   export DSMI_DIR=/opt/tivoli/tsm/client/api/bin
   export DSMI_CONFIG=$HOME/tsm/dsm.opt
   export DSMI_LOG=$HOME/dldump
   export PATH=$PATH:/opt/tivoli/tsm/client/api/bin

   (C シェルの場合)
   setenv DSMI_DIR /opt/tivoli/tsm/client/api/bin
   setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
   setenv DSMI_LOG ${HOME}/dldump
   setenv PATH=${PATH}:/opt/tivoli/tsm/client/api/bin
   ```

4. `/opt/tivoli/tsm/client/api/bin` ディレクトリー中に `dsm.sys` TSM システム・オプション・ファイルがあるか確認します。
5. *INSTHOME*`/tsm` ディレクトリー (*INSTHOME* はデータ・リンク・マネージャー管理者のホーム・ディレクトリー) に、 `dsm.opt` TSM ユーザー・オプション・ファイルがあるか確認します。
6. `/opt/tivoli/tsm/client/api/bin/dsm.sys` という Tivoli Storage Manager システム・オプション・ファイル中の *PASSWORDACCESS* オプションを `generate` に設定します。
7. 初めてデータ・リンク・ファイル・マネージャーを開始する*前*に、 TSM パスワードを `generate` オプションに登録します。この作業を行うと、データ・リンク・ファイル・マネージャーで TSM サーバーに対する接続が開始される際にパスワードを入力する必要がなくなります。詳しくは、TSM 製品の資料を参照してください。
8. DLFM_BACKUP_TARGET レジストリー変数を TSM に設定します。この場合は DLFM_BACKUP_DIR_NAME レジストリー変数の値は無視されます。これにより、Tivoli Storage Manager バックアップ・オプションがアクティブにされます。

注:

a. 実行時に DLFM_BACKUP_TARGET レジストリー変数の設定を TSM とディスクの間で切り替える場合は、アーカイブ済みのファイルは新しく指定したアーカイブ場所に移動されないことに注意する必要があります。たとえば、DLFM_BACKUP_TARGET レジストリー値を TSM に設定してデータ・リンク・ファイル・マネージャーを開始した場合に、このレジストリー値をディスク位置に変更すると、新しいアーカイブ・ファイルはすべてディスク上の新しい場所に保管されます。以前に TSM にアーカイブされたファイルは、新しいディスク位置に移動されません。

b. デフォルトの TSM 管理クラスをオーバーライドするには、新しいレジストリー変数 DLFM_TSM_MGMTCLASS を使用します。このレジストリー変数を未設定にしておくと、デフォルトの TSM 管理クラスが使用されます。

9. **dlfm stop** コマンドを入力して、データ・リンク・ファイル・マネージャーを停止します。
10. **dlfm start** コマンドを入力して、データ・リンク・ファイル・マネージャーを開始します。

Windows NT 上で DB2 データ・リンク・マネージャーのバックアップ方式を選択する

回復用に定義された DATALINK 列のある表に DATALINK 値を挿入すると、必ずデータ・リンク・サーバー上の対応する DATALINK ファイルは、アーカイブ・サーバーにバックアップされるようにスケジュールされます。現時点では、アーカイブ・サーバーへのファイル・バックアップ用に、ディスク・コピー (デフォルトの方式) と Tivoli Storage Manager という 2 つのオプションがサポートされています。今後のリリースの DB2 Data Links Manager for Windows NT では、他のベンダーのバックアップ・メディアやソフトウェアがサポートされる予定です。

ディスク・コピー (デフォルトの方式)

DB2 サーバー上でバックアップ・ユーティリティーを呼び出すと、データベース中のリンク・ファイルが、データ・リンク・サーバー上の DLFM_BACKUP_DIR_NAME 環境変数で指定したディレクトリーにバックアップされるか確認されます。この変数のデフォルト値は、 c:¥dlfmbackup (c:¥ はデータ・リンク・マネージャーのバックアップ・インストール・ドライブ) です。

この変数を c:¥dlfmbackup に設定するには、以下のコマンドを入力してください。

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

DLFM_BACKUP_DIR_NAME 環境変数で、データ・リンク・ファイルシステム・フィルターを使用するファイル・システム上の場所を指定しないでください。これに従うと、バックアップ・ファイル用に指定したディレクトリー中で必須スペースが使用可能になります。

さらに、以下のコマンドを入力して、 DLFM_BACKUP_TARGET 変数が LOCAL に設定されていることを確認してください。

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

これらの変数を設定したり変更したりしたら、 **dlfm stop** コマンドと **dlfm start** コマンドを使用して、データ・リンク・ファイル・マネージャーを停止してから再始動してください。

Tivoli Storage Manager

Tivoli Storage Manager をアーカイブ・サーバーとして使用するには、以下のようにします。

1. データ・リンク・サーバーに Tivoli Storage Manager をインストールします。詳しくは、Tivoli Storage Manager 製品の文書を参照してください。
2. データ・リンク・サーバー・クライアント・アプリケーションを Tivoli Storage Manager サーバーに登録します。詳しくは、Tivoli Storage Manager 製品の文書を参照してください。
3. **「スタート」**をクリックして、**「設定」 ―> 「コントロール パネル」 ―> 「システム」**を選択します。「システムのプロパティ」ウィンドウがオープンします。**「環境」**タブを選択して、以下の環境変数と対応する値を入力します。

変数	値
DSMI_DIR	c:¥tsm¥baclient
DSMI_CONFIG	c:¥tsm¥baclient¥dsm.opt
DSMI_LOG	c:¥tsm¥dldump

4. `c:¥tsm¥baclient` ディレクトリー中に `dsm.sys` TSM システム・オプション・ファイルがあるか確認します。
5. `c:¥tsm¥baclient` ディレクトリー中に `dsm.opt` TSM ユーザー・オプション・ファイルがあるか確認します。
6. `c:¥tsm¥baclient¥dsm.sys` という Tivoli Storage Manager システム・オプション・ファイル中の *PASSWORDACCESS* オプションを `generate` に設定します。
7. 初めてデータ・リンク・ファイル・マネージャーを開始する*前*に、 TSM パスワードを `generate` オプションに登録します。この作業を行うと、データ・リンク・ファイル・マネージャーで TSM サーバーに対する接続が開始される際にパスワードを入力する必要がなくなります。詳しくは、TSM 製品の資料を参照してください。
8. 以下のコマンドを使用して、DLFM_BACKUP_TARGET 環境変数を TSM に設定します。

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

この場合は DLFM_BACKUP_DIR_NAME 環境変数の値は無視されます。これにより、Tivoli Storage Manager バックアップ・オプションがアクティブにされます。

注:

a. 実行時に DLFM_BACKUP_TARGET 環境変数の設定を TSM と LOCAL の間で切り替える場合は、アーカイブ済みのファイルは新しく指定したアーカイブ場所に移動されないことに注意する必要があります。たとえば、 DLFM_BACKUP_TARGET 環境変数を TSM に設定してデータ・リンク・ファイル・マネージャーを開始した場合に、この値を LOCAL に変更すると、新しいアーカイブ・ファイルはすべてディスク上の新しい場所に保管されます。以前に TSM にアーカイブされたファイルは、新しいディスク位置に移動されません。

b. デフォルトの TSM 管理クラスをオーバーライドするには、新しい環境変数 DLFM_TSM_MGMTCLASS を使用します。この変数を未設定にしておくと、デフォルトの TSM 管理クラスが使用されます。

9. **dlfm stop** コマンドを入力して、データ・リンク・ファイル・マネージャーを停止します。
10. **dlfm start** コマンドを入力して、データ・リンク・ファイル・マネージャーを開始します。

AIX でのジャーナル・ファイル・システムのバックアップ

AIX 上で、データ・リンク・マネージャーを停止した後にジャーナル・ファイル・システムのオフライン・バックアップを実行できます。以下の方法はデータ・リンク・マネージャーを停止する必要がないので、高可用性が必要なユーザーにお勧めします。

1. CLI ソース・ファイル quiesce.c およびシェル・スクリプト online.sh にアクセスします。これらのファイルは、 /samples/dlfm ディレクトリーにあります。
2. 以下のように quiesce.c をコンパイルします。

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

3. root として、DLFS ファイル・システムのあるノード上でスクリプトを実行します。

シェル・スクリプト online.sh は、データ・リンク・マネージャーに登録されているデータベースごとのカタログ・エントリーがデータ・リンク・マネージャー・ノード上にあることを前提にしています。また、 /etc/filesystems に DLFS ファイル・システムの完全なエントリーがあることも前提にしています。シェル・スクリプトにより以下の処理が実行されます。

- データ・リンク・マネージャーに登録されているデータベース中の表をすべて静止する。これにより、新しいアクティビティーがすべて停止します。

- ファイル・システムをアンマウントして、読み取り専用ファイル・システムとして再マウントする。
- ファイル・システムのバックアップを実行する。
- ファイル・システムをアンマウントして、読み取り / 書き込みファイル・システムとして再マウントする。
- DB2 表をリセットする。すなわち、静止状態を解除します。

以下のように、ご使用の環境に合わせてスクリプトに変更を加えなければなりません。

1. バックアップ・コマンドを選択して、スクリプトの do_backup 関数を挿入します。
2. スクリプト中に以下の環境変数を設定します。
   - DLFM_INST: DLFM インスタンス名に設定します。
   - PATH_OF_EXEC: ″quiesce″ 実行可能ファイルがあるパスに設定します。

以下のように、スクリプトを呼び出します。

```
online.sh <filesystem_name>
```

復元およびロールフォワードについての考慮事項

下記の情報は、表に対して RECOVERY=YES オプションが定義された DATALINK 列 (複数も可能) が存在する場合当てはまります。表で RECOVERY=NO オプションが定義されている DATALINK 列が定義されている場合、表は復元処理終了後にデータ・リンク調整保留 (DRP) 状態になります。詳しくは、79ページの『データ・リンクの調整』を参照してください。

復元操作中には、DATALINK 列をもつ表は、次の 2 つのうちいずれかの状態になります。

- *データ・リンク調整不能 (DRNP)*

  表がデータ・リンク調整不能 (DRNP) 状態にある場合、 DATALINK 列以外のすべての列に対して無制限のアクションを行えます。 SELECT ステートメントに DATALINK 列が関係している場合、警告が戻されます。しかし、DATALINK 列に対して UPDATE 呼び出しは発行できます (いくつかの制限付きで、詳細については、78ページの『データ・リンク調整不能 (DRNP) 状態から表を取り除く』を参照)。INSERT および DELETE ステートメントには DATALINK 列が関係するので発行できません。

- *データ・リンク調整保留 (DRP)*

  表がデータ・リンク調整保留 (DRP) 状態にある場合、 DATALINK 列以外のすべての列に対して無制限のアクションを行えます。 SELECT ステートメントに DATALINK 列が関係している場合、警告が戻されます。 UPDATE、INSERT、または DELETE などの DML ステートメントは、どれも発行できません。

これらの状態は、復元またはロールフォワード・ユーティリティーが実行されるときに、 db2diag.log ファイルに報告されます。この情報を得るために、 **db2dart** コマンドを使用することもできます。

データベースや表スペースを復元しても、復元操作が正常に実行されるには、次の条件が満たされなければなりません。

- バックアップ・ファイルに記録されているデータ・リンク・サーバーが実行中でない場合でも、復元操作は正常に完了する。

  欠落しているデータ・リンク・サーバーの影響を受ける DATALINK 列情報のある表は、復元操作 (または、使用されている場合にはロールフォワード操作) の完了後に、データ・リンク調整保留 (DRP) 状態にされます。データ・リンク・サーバーがデータベースで使用可能としてマークが付けられるには、その前に、この復元処理が正常に完了しなければなりません。 DLM が使用可能になった時点でバックアップ処理を完了する非同期処理 (58ページの『バックアップ・ユーティリティーについての考慮事項』を参照) によっても、復元処理は完了します。
- バックアップ・ファイルに記録されているデータ・リンク・サーバーが、復元操作中に実行を停止する場合でも、復元操作は正常に完了する。データ・リンク・サーバーがダウンしていても、復元を再始動できます (上記参照)。
- データ・リンク・サーバー上での以前のデータベース復元操作がまだ完了していない場合は、以後のデータベースや表スペースの復元操作は、それらのデータ・リンク・サーバーが再始動されて未完了の復元が完了しない限り失敗する。
- バックアップ・ファイルに記録される DATALINK 列すべてについての情報が、適切なデータ・リンク・サーバーの登録表に存在している。

  DATALINK 列についての情報がすべて登録表に記録されていない場合、 DATALINK 列についての情報が欠落している表は、復元操作 (もし使用している場合にはロールフォワード操作) が終了すると、データ・リンク調整不能 (DRNP) 状態になります。

  バックアップが登録表に記録されていない場合には、提供されるバックアップ・ファイルが num_db_backups の値より早く、すでにガーベッジ・コレクションされたことを意味する場合があります。これは、この早期バックアップからアーカイブされたファイルが削除されていて、復元できないことを意味します。 DATALINK 列のある表はすべて、データ・リンク調整保留 (DRP) 状態になります。

  バックアップが登録表に記録されていない場合には、データ・リンク・サーバーが実行中でないためバックアップ処理がまだ完了していないことを意味する場合があります。 DATALINK 列のある表はすべて、データ・リンク調整保留 (DRP) 状態になります。データ・リンク・サーバーを再始動すると、バックアップ処理が完了してから、復元処理が行われます。

  表は、ユーザーによって使用可能なままです。しかし、DATALINK 列の値はファイルを正確に参照しないかもしれません (たとえば、 DATALINK 列の値に一致するファイルが見つからない場合があります)。この動作が必要でない場合、「SET CONSTRAINTS for tablename TO DATALINK RECONCILE PENDING」ステートメントを発行して、表を検査保留状態にすることができます。

復元操作を行った後にデータ・リンク調整不能 (DRNP) 状態の表がある場合は、 78ページの『データ・リンク調整不能 (DRNP) 状態から表を取り除く』で推奨されている方法の 1 つを使って DATALINK 列データを修正できます。

注: ファイルをリンクされていない状態からリンクされた状態へとマーキングする過程で、そのファイルをアーカイブ・サーバーから取り出してからファイル・システムに入れなければならないかもしれません。この処理を行う間にエラーが起きた場合 (たとえば、ファイル名が重複しているためにファイル・システムへコピーできないときなど)、それに対応している表はデータ・リンク調整保留 (DRP) 状態になります。

datalink.cfg ファイルをアーカイブして、特殊な回復に備えることを強くお勧めします。なぜなら、データベース・バックアップ・イメージ中の datalink.cfg ファイルには、バックアップ時の datalink.cfg しか反映されないからです。あらゆる回復のケースを網羅するには、最新の datalink.cfg ファイルが必要です。したがって、 ADD DATALINKS MANAGER コマンドや DROP DATALINKS MANAGER コマンドを呼び出すたびに、その後で datalink.cfg ファイルのバックアップを取らなければなりません。こうすると、ディスク上の最新の datalink.cfg ファイルが利用不能な場合に、最新の datalink.cfg ファイルを取り出すのに役立ちます。

ディスク上の最新の datalink.cfg ファイルが利用不能な場合は、既存の datalink.cfg ファイル (バックアップ・イメージから復元したもの) を、ロールフォワード操作の実行前にアーカイブした最新の datalink.cfg ファイルに取り替えてください。この作業は、データベースの復元後に行ってください。

ロールフォワードせずにデータベースをオフライン・バックアップから復元する

ロールフォワードをせずに復元できるのはデータベース・レベルだけで、表スペース・レベルでは不可能です。ロールフォワードせずにデータベースを復元するには、回復不能のデータベース (循環ログを使用するデータベース) を復元するか、 RESTORE DATABASE コマンドで WITHOUT ROLLING FORWARD パラメーターを指定します。

WITHOUT DATALINK オプションを指定した復元ユーティリティーを使用する場合、DATALINK 列をもつすべての表はデータ・リンク調整保留 (DRP) 状態になり、復元操作中にはデータ・リンク・サーバーによる調整は一切行われません。

WITHOUT DATALINK オプションを使用していない場合に、バックアップ・ファイルに記録されているデータ・リンク・サーバーがデータベースに定義されなくなる (つまり、 DROP DATALINKS MANAGER コマンドを使用して除去する) と、除去されたデータ・リンク・サーバーを参照する DATALINK データを含む表は、復元ユーティリティーによって DRP 状態にされます。

WITHOUT DATALINK オプションを使用していないか、またはデータ・リンク・サーバーが使用可能で、 DATALINK 列についての情報がすべて登録表に記録されている場合、バックアップ・ファイルに記録されているすべてのデータ・リンク・サーバーで次のことが起きます。

- バックアップ・イメージ (データベースの復元操作に使用されたもの) の後にリンクされたファイルは、すべてリンクされていないものとしてマーク付けられます (バックアップ・イメージではリンクされたものとして記録されていないため)。
- バックアップ・イメージの後にリンク解除されたファイルで、バックアップ・イメージが取られる前はリンクされていたものはすべて、リンクされているものとしてマーク付けられます (バックアップ・イメージではリンクされたものとして記録されている)。ファイルが後に別のデータベースの表にリンクされる場合、復元された表はデータ・リンク調整保留 (DRP) 状態になります。

注: 少なくとも 1 つのデータ・リンク・サーバーが実行中でなかったときに取られたバックアップ・イメージがデータベースの復元操作に使用された場合は、バックアップ中の DATALINK 情報が未完了状態なので、上記の処理を行えません。 (ロールフォワード回復を行うかどうかに関係なく) データベースの復元操作後に取られたバックアップ・イメージが、データベースの復元操作に使用された場合も、上記の処理は行われません。どちらの場合も、DATALINK 列をもつすべての表はデータ・リンク調整保留 (DRP) 状態になり、復元操作中にはデータ・リンク・サーバーによる調整は一切行われません。

データベースおよび表スペースを復元してからログの終わりまでロールフォワードする

データベースまたは表スペースを復元してからログの終わりまでロールフォワードすると (すべてのログが提供されている)、バックアップ・ファイル中に記録されている少なくとも 1 つのデータ・リンク・サーバーが復元操作時に実行中でなくても、調整検査を行う必要はありません。ロールフォワード操作ですべてのログが提供されたかどうか分からない場合、または DATALINK 値を調整する必要があると思える場合には、次のことを行います。

1. 関係している表 (または複数の表) に対して、以下の SQL ステートメントを発行します。

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

   このステートメントは、表をデータ・リンク調整保留 (DRP) 状態および検査保留状態にします。
2. 表を検査保留状態にしたくない場合は、次の SQL ステートメントを発行します。

   ```
   SET CONSTRAINTS FOR tablename IMMEDIATE CHECKED
   ```

これは、表を検査保留状態から解放し、データ・リンク調整保留 (DRP) 状態のままにします。表をこの状態から解放するには、調整ユーティリティーを使用する必要があります。

データベースから除去された DB2 データ・リンク・マネージャーを参照する DATALINK データが、バックアップ・ファイルに含まれている (つまり、バックアップが取られたときに DB2 データ・リンク・マネージャーがデータベースに登録された) ことがあります。除去された DB2 データ・リンク・マネージャーを参照する DATALINK データの含む表が少なくとも 1 つある表スペースをロールフォワードするたびに、ロールフォワード・ユーティリティーによって、DATALINK 列のある表はすべて DRP 状態にされます。

データベースおよび表スペースを復元してからある時点までロールフォワードする

データ・リンク表を使用して作業する際、ログの最後まで、または指定時刻までのロールフォワードを実行できます。

指定時刻までロールフォワードされる表スペースの表は、ロールフォワード操作の終了時にはデータ・リンク調整保留 (DRP) 状態になります。この状態から解放するには、調整 (reconcile) ユーティリティーを使用する必要があります。詳しくは、79ページの『データ・リンクの調整』を参照してください。

時刻指定ロールフォワードの例

バックアップおよび回復処理をするために必要なファイルを示す簡単なシナリオは、次のとおりです。この例は、 DATALINK タイプの列の単一行に加える変更と回復をサポートするために DB2 データ・リンク・マネージャーが保持する必要のあるファイルを示します。この例では、最後のバックアップよりも以前のバックアップを使って、ファイルの時刻指定回復は行わないことが前提になっています。 DB2 データ・リンク・マネージャーを実行しているデータ・リンク・サーバーには、このような制約事項はありません。この例のデータベース管理方針 (つまり、 *num_db_backups* データベース構成パラメーターは 1 に設定される) では、次のバックアップが実行されるまで、リンク解除されたファイルが保持されます。たとえば、fileA に注目してください。このファイルのリンクは time 2 で解除されたため、time 3 まで保持されてから削除されます。

Time	1	2	3	4	5	6	7
アクティビティー	作成	更新	バックアップ	更新	更新	削除	5 に復元
列値	valueA	valueB	valueB	valueC	valueD	-	valueD
リンクされたファイル	fileA	fileB	fileB	fileC	fileD	-	fileD

Time	1	2	3	4	5	6	7
データ・リンク・ファイル・マネージャーにより保持されるエクストラ・ファイル		fileA		fileB	fileB、fileC	fileB、fileC、fileD	fileB、fileC

注: リンク・ファイルの回復は、データベースの他の部分と同じときに実行されます。

DB2 データ・リンク・マネージャーと回復の対話

次の表は、実行可能な様々な種類の回復、復元およびロールフォワード処理時に発生する DB2 データ・リンク・マネージャー処理、回復終了後に調整ユーティリティーを呼び出す必要があるかどうかを示します。

回復の種類	復元時の DB2 データ・リンク・マネージャーの処理	ロールフォワード時の DB2 データ・リンク・マネージャーの処理	調整
回復不能データベース			
データ・リンク・サーバーがすべて起動している場合の、完全バックアップによるデータベース復元	高速な調整が実行される	なし	ファイル・リンクで問題があると思われる場合には、オプションで実行できる
WITHOUT DATALINK オプションを使ったデータベースの復元	データ・リンク調整保留 (DRP) 状態の表	なし	必須

回復の種類	復元時の DB2 データ・リンク・マネージャーの処理	ロールフォワード時の DB2 データ・リンク・マネージャーの処理	調整
少なくとも 1 つのデータ・リンク・サーバーがダウンしている場合の、完全バックアップによるデータベース復元	ダウンしているデータ・リンク・サーバーに対するリンクのない表スペース中の表に限り高速な調整が実行され、それ以外の表はデータ・リンク調整保留 (DRP) 状態にされる	なし	ダウンしているデータ・リンク・サーバーに対するリンクのある表スペース中の表について必須
データ・リンク・サーバーがすべて起動している場合の、不完全バックアップによるデータベース復元	高速な調整は実行されず、DATALINK 列のある表はすべてデータ・リンク調整保留 (DRP) 状態にされる	なし	必須
回復可能データベース			
データ・リンク・サーバーがすべて起動している場合の、WITHOUT ROLLING FORWARD オプションを使用した、完全バックアップによるデータベース復元	高速な調整が実行される	なし	オプション
データ・リンク・サーバーが起動またはダウンしている場合の、 WITHOUT ROLLING FORWARD および WITHOUT DATALINK オプションを使用した、完全または不完全バックアップによるデータベース復元	データ・リンク調整保留 (DRP) 状態の表	なし	必須

回復の種類	復元時の DB2 データ・リンク・マネージャーの処理	ロールフォワード時の DB2 データ・リンク・マネージャーの処理	調整
少なくとも 1 つのデータ・リンク・サーバーがダウンしている場合の、WITHOUT ROLLING FORWARD オプションを使用した、完全バックアップによるデータベース復元	ダウンしているデータ・リンク・サーバーに対するリンクのない表スペース中の表に限り高速な調整が実行され、それ以外の表はデータ・リンク調整保留 (DRP) 状態にされる	なし	ダウンしているデータ・リンク・サーバーに対するリンクのある表スペース中の表について必須
データ・リンク・サーバーが起動またはダウンしている場合の、 WITHOUT ROLLING FORWARD オプションを使用した、不完全バックアップによるデータベース復元	高速な調整は実行されず、DATALINK 列のある表はすべてデータ・リンク調整保留 (DRP) 状態にされる	なし	必須
データ・リンク・サーバーがすべて起動している場合の、完全バックアップによる、データベース復元およびログの終わりまでのロールフォワード	処置なし	処置なし	オプション

回復の種類	復元時の DB2 データ・リンク・マネージャーの処理	ロールフォワード時の DB2 データ・リンク・マネージャーの処理	調整
ロールフォワード処理中に少なくとも 1 つのデータ・リンク・サーバーがダウンしている場合の、完全バックアップによる、データベース復元およびログの終わりまでのロールフォワード	処置なし	処置なし	オプション
復元中にデータ・リンク・サーバーがダウンしている場合の、完全または不完全バックアップによる、データベース復元およびログの終わりまでのロールフォワード	処置なし	DATALINK 列のある表はすべてデータ・リンク調整保留 (DRP) 状態にされる	DATALINK 列のあるすべての表について必須
復元中にデータ・リンク・サーバーがすべて起動している場合の、不完全バックアップによる、データベース復元およびログの終わりまでのロールフォワード	処置なし	処置なし	オプション

回復の種類	復元時の DB2 データ・リンク・マネージャーの処理	ロールフォワード時の DB2 データ・リンク・マネージャーの処理	調整
データ・リンク・サーバーがすべて起動しており、バックアップがいずれかのデータ・リンク・サーバーで不明な場合の、完全または不完全バックアップによる、データベース復元およびログの終わりまでのロールフォワード	処置なし	バックアップが不明なデータ・リンク・サーバーに対するリンクのある表スペース中の表はすべてデータ・リンク調整保留 (DRP) 状態にされる	必須
データ・リンク・サーバーがすべて起動している場合の、完全バックアップによる、表スペース復元およびログの終わりまでのロールフォワード	処置なし	処置なし	オプション
ロールフォワード処理中に少なくとも 1 つのデータ・リンク・サーバーがダウンしている場合の、完全バックアップによる、表スペース復元およびログの終わりまでのロールフォワード	処置なし	処置なし	オプション

回復の種類	復元時の DB2 データ・リンク・マネージャーの処理	ロールフォワード時の DB2 データ・リンク・マネージャーの処理	調整
復元処理中にデータ・リンク・サーバーがダウンしている場合の、完全または不完全バックアップによる、表スペース復元およびログの終わりまでのロールフォワード	処置なし	ダウンしているデータ・リンク・サーバーに対するリンクのある表スペース中の表はすべてデータ・リンク調整保留 (DRP) 状態にされる	ダウンしているデータ・リンク・サーバーに対するリンクのある表スペース中の表について必須
データ・リンク・サーバーがすべて起動している場合の、不完全バックアップによる、表スペース復元およびログの終わりまでのロールフォワード	処置なし	処置なし	オプション
復元、ロールフォワード、またはその両方の処理中にデータ・リンク・サーバーが起動またはダウンしている場合の、完全または不完全バックアップによる、データベース復元および時刻指定のロールフォワード	処置なし	データ・リンク調整保留 (DRP) 状態の表	必須

回復の種類	復元時の DB2 データ・リンク・マネージャーの処理	ロールフォワード時の DB2 データ・リンク・マネージャーの処理	調整
復元、ロールフォワード、またはその両方の処理中にデータ・リンク・サーバーが起動またはダウンしている場合の、完全または不完全バックアップによる、表スペース復元および時刻指定のロールフォワード	処置なし	データ・リンク調整保留 (DRP) 状態の表	必須
別のデータベース名、別名、ホスト名、またはロールフォワードのないインスタンスへのデータベースの復元 (78ページの1 ページの注を参照のこと)	データ・リンク調整不能 (DRNP) 状態の表	なし	オプション、しかしデータ・リンク調整不能 (DRNP) 状態の表は手作業で修正する
別のデータベース名、別名、ホストまたはインスタンスへのデータベースの復元、およびロールフォワード	処置なし	データ・リンク調整不能 (DRNP) 状態の表	オプション、しかしデータ・リンク調整不能 (DRNP) 状態の表は手作業で修正する

回復の種類	復元時の DB2 データ・リンク・マネージャーの処理	ロールフォワード時の DB2 データ・リンク・マネージャーの処理	調整
WITHOUT DATALINK オプションが指定されている場合とされていない場合の、使用できないバックアップからのデータベースの復元 (データ・リンク・サーバーでイメージがガーベッジ・コレクションされた)。ロールフォワードは行われない (78ページの1ページの注を参照のこと)。	データ・リンク調整保留 (DRP) 状態の表	処置なし	必須
WITHOUT DATALINK オプションが指定されている場合とされていない場合の、使用できないバックアップからのデータベースの復元 (データ・リンク・サーバーでイメージがガーベッジ・コレクションされた)、そしてロールフォワードを行う。	処置なし	データ・リンク調整保留 (DRP) 状態の表	必須

回復の種類	復元時の DB2 データ・リンク・マネージャーの処理	ロールフォワード時の DB2 データ・リンク・マネージャーの処理	調整
使用できないバックアップからの表スペースの復元 (データ・リンク・サーバーでイメージがガーベッジ・コレクションされた)、そしてロールフォワードを行う。	処置なし	データ・リンク調整保留 (DRP) 状態の表	必須

注:

1. WITHOUT ROLLING FORWARD オプションを指定した、オンライン・バックアップ・イメージを使用した復元操作、またはオフライン・バックアップ・イメージを使用した復元操作。
2. *完全 バックアップ*とは、必須のデータ・リンク・サーバーがすべて実行中のときに取ったバックアップのことです。*不完全 バックアップ*とは、必須のデータ・リンク・サーバーのうち 1 つ以上が実行中でなかったときに取ったバックアップのことです。
3. (ロールフォワードを行うかどうかに関係なく) データベースの復元後に取られたバックアップ・イメージが、データベースの復元操作に使用された場合には、高速な調整処理は実行されません。この場合には、DATALINK 列のある表はすべてデータ・リンク調整保留 (DRP) 状態にされます。

データ・リンク調整不能 (DRNP) 状態から表を取り除く

num_db_backups データベース構成パラメーターで指定された値よりも前のバックアップから、表スペースが復元された場合、 DATALINK 列のある復元された表はデータ・リンク調整不能 (DRNP) 状態にされます。この構成パラメーターについての詳細は、*管理の手引き: パフォーマンス* を参照してください。

DATALINK 列の値が有効でないとしても、DB2 で表をアクセスすることはまだ可能です。矛盾しているかもしれない DATALINK 列の値をもつ表へのアクセスを禁止したい場合には、 SET CONSTRAINTS for *tablename* TO DATALINK RECONCILE PENDING コマンドを発行します。次のように DATALINK 値を更新できます。

- SQL UPDATE ステートメントを使用して、 DATALINK 列値のデータ位置部分をゼロ長の URL に設定する (列がヌル不能な場合)、列がヌル可能な場合には NULL に設定します。

- 適切なデータ・リンク・サーバーでファイルを復元します。それから、 SELECT ステートメントを発行するアプリケーションを実行して DATALINK 列の値を読み取ります。そして、UPDATE ステートメントを発行して DATALINK 列を同じ値で更新します。 DATALINK 列の値が更新されている間は、データ・リンク調整不能 (DRNP) 状態でなければならないことに注意してください。更新操作が完了すると、適切なデータ・リンク・サーバーでファイルがリンクされたものとしてマークされます。

それから、次のコマンドを発行して、データ・リンク調整不能 (DRNP) 状態をリセットします。

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

データ・リンクの調整

データ・リンクを調整するには、調整 (reconcile) ユーティリティーを使用します。このユーティリティーは DB2 から開始され、 DATALINK 列値で参照される DB2 データ・リンク・サーバーを実行しているすべてのデータ・リンク・サーバーが関係しています。参照ファイルがデータ・リンク・サーバーに存在していること、またはリンクを再確立できることを検査する妥当性検査を実行します。以下の節では、データ・リンクを調整する必要を DB2 が見極める方法と、調整方法を説明します。

データ・リンク・サーバーのファイル参照が存在しない場合、または再確立できない場合、調整ユーティリティーはエラーが起きた行のコピーと各エラーの理由を、例外表 (指定した場合) に置きます。それから、エラーが起きた行を修正します。例外表が指定されていない場合には、ファイル参照が再確立できなかった DATALINK 列の値が例外報告ファイルにコピーされ、これに列 ID と理由も入れられます。この例外表 (指定した場合) の情報またはレポートを使用すると、行を更新して必要な訂正を行えます。調整ユーティリティーで使用される例外表は、ロード・ユーティリティーで使用される例外表と同じです。ロード・ユーティリティーの詳細については、*データ移動ユーティリティー 手引きおよび解説書* を参照してください。レポートでは、 *report*.exp (.exp 拡張子は、調整ユーティリティーが提供します) という命名規則が使用されます。たとえば、次のステートメントを使用して調整ユーティリティーを呼び出せます。

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

このコマンドは、dept と呼ばれる表を調整し、ユーザーにより作成された例外表 excptab に例外を書き込みます。調整中にリンク解除されたファイルについての情報は、ファイル report.ulk に書き込まれます。このファイルは、ディレクトリー /u/scottba に作成されます。 *FOR EXCEPTION excptab* が指定されていない場合には、例外情報がファイル report.exp に書き込まれます。このファイルは、ディレクトリー /u/scottba に作成されます。調整ユーティリティーについての詳細は、*コマンド解説書* を参照してください。

調整が必要な状態の検出

次のような状態では、調整ユーティリティーを実行する必要があるかもしれません。

- データベース全体が復元され、時刻指定でロールフォワードされます。コミット済みのトランザクションまでデータベース全体がロールフォワードされるので、検査保留状態にある表はありません (参照制約または検査制約のため)。データベースのすべてのデータは、一貫した状態になります。しかし、DATALINK 列は DB2 データ・リンク・マネージャーのメタデータとは同期化できず、調整が必要です。

  この状態では、DATALINK データ付きの表はすでに DRP 状態にあります。これらのそれぞれの表ごとに調整ユーティリティーを呼び出す必要があります。
- DB2 データ・リンク・マネージャーを実行している特定のデータ・リンク・サーバーでは、メタデータの位置が分からなくなります。これには、いくつかの理由が挙げられます。たとえば、次のようなものがあります。
  - データ・リンク・サーバーがコールド・スタートした。
  - データ・リンク・メタデータがバックレベル状態に復元された。

  ある場合 (SQL UPDATE および DELETE 中など)、 DB2 はデータ・リンク・サーバーでメタデータに関する問題を検出するかもしれません。このようなときは、SQL ステートメントは失敗します。 SET CONSTRAINTS ステートメントを使用して、表を DRP 状態にしてから、その表で調整ユーティリティーを実行します。
- ファイル・システムは使用不能で (たとえば、ディスク破損のため)、現行の状態には復元されません。この状態では、ファイルが欠落している可能性があります。
- DB2 データ・リンク・マネージャーがデータベースから除去されていて、 DB2 データ・リンク・マネージャーを参照する DATALINK FILE LINK CONTROL 値がある場合。このような表で調整ユーティリティーを実行する必要があります。

調整手順の要約

時刻指定回復のために、または DB2 データ・リンク・マネージャーを実行しているデータ・リンク・サーバーと DB2 制御情報が一致しないために、データ・リンクを調整する必要が生じる場合、以下のことを行ってください。

1. SET CONSTRAINTS ステートメントを発行して、表をデータ・リンク調整保留 (DRP) 状態にします。 (ある場合、DB2 がこれを行います。)
2. 調整ユーティリティーを使用してリンクを解決してから、例外表または例外報告の例外に対して適切な処置を実行します。

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 NT/2000 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
```

バックアップ・ユーティリティーを起動する際にオプションを指定すると、その宛先にバックアップ・イメージが作成されます。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 (451ページの『付録G. Tivoli Storage Manager』 を参照)
- 他のベンダーのサーバー
- ユーザー出口プログラムを使用して指定した場所 (OS/2 のみ)

全データベースのバックアップ操作を起動すると、ヒストリー・ファイルが要約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。リカバリー・ヒストリー・ファイルについては、49ページの『リカバリー・ヒストリー・ファイルについて』を参照してください。

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、次のようになります。

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

その他のプラットフォームでは、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、次のようになります (Windows NT/2000)。

```
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 環境変数からとられる 1〜8 文字の現行インスタンス名。

ノード番号
ノード番号。非区分データベース・システムでは、この値は常に NODE0000 です。区分データベース・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。非区分データベース・システムでは、この値は常に CATN0000 です。区分データベース・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)

順序番号
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、大規模な区分データベース・システムでは、テープ装置を各データベース区画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、各データベース区画サーバーから可能にするためです。 TSM の詳細については、451ページの『付録G. Tivoli Storage Manager』を参照してください。
- 区分データベース・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 (データベース区画サーバー) 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。

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法

バックアップを使用する前に

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。

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。

区分データベース・システムでは、データベース区画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区画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区画サーバーから **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区画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。 LIST NODES コマンドについては、*コマンド解説書* を参照してください。) 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とり、次に、他のデータベース区画のバックアップをとる必要があります。 Command Center を使用してデータベース区画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回復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 (表および視点) 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マイグレーションしていない場合は、データベースをマイグレーション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。データベースのマイグレーションについては、*管理の手引き: 計画* を参照してください。

バックアップの起動

バックアップ・ユーティリティーは、以下の方法で起動できます。

- コマンド行プロセッサー (CLP)。

  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
  ```

- コントロール・センターの「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。「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。

  コントロール・センターについての一般情報は、*管理の手引き* を参照してください。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。

- アプリケーション・プログラミング・インターフェース (API) としての **sqlubkp**。この API については、93ページの『Backup Database API』を参照してください。 DB2 管理用 API を含むアプリケーションの作成についての一般情報は、 *アプリケーション構築の手引き* を参照してください。

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、および復元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。

このユーティリティーの詳細については、319ページの『db2ckbkp - バックアップの検査』を参照してください。

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、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、バックアップ操作に時間がかかります。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回復できる手順を、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。

変数長ブロック・サイズのバックアップ・イメージから復元すると、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。

```
tcl -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	接続	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復元する) ためのサポートを使用できるようになりました。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。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. 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。

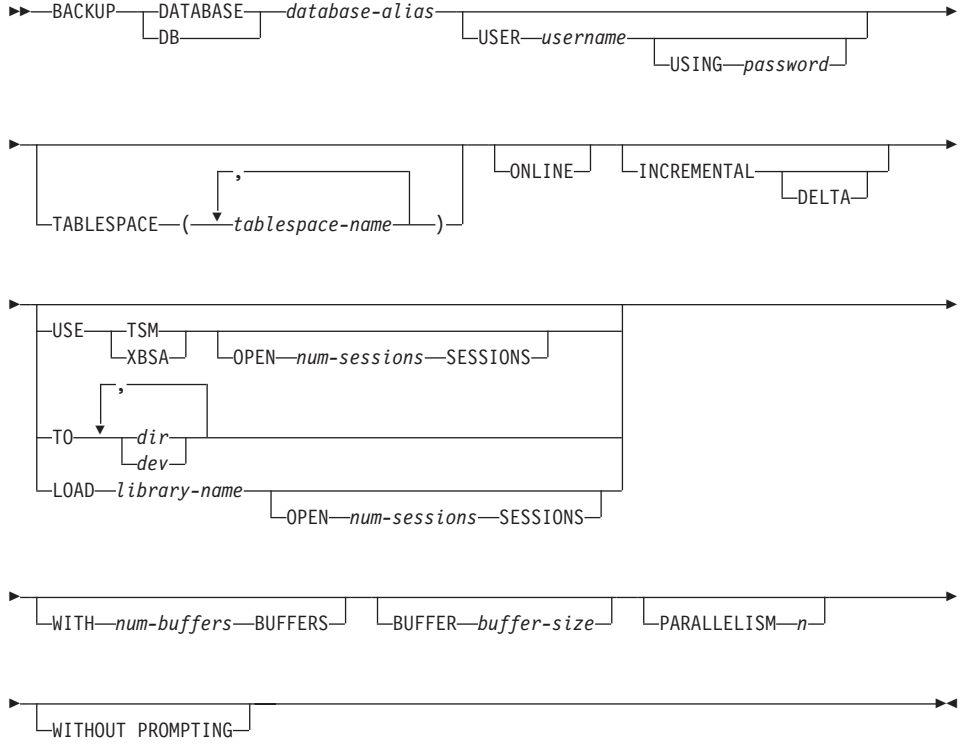
   ```
   db2 backup db sample to /u/dmcinnis/mypipe
   ```

3. 復元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* 復元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
   ```

BACKUP DATABASE コマンド

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
ユーザー名を承認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。デフォルトはオンライン・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。

注: オンライン・バックアップ操作は、 sysibm.systables に IX ロックがある場合、タイムアウトになる可能性があります。それは DB2 バックアップ・ユーティリティーが、 LOB を含むオブジェクト上に S ロックを必要とするためです。

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。

DELTA 非累積 (デルタ) バックアップ・イメージを指定します。デルタ・バックアップ・イメージは、最新の正常実行された任意のタイプの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。

USE TSM
バックアップに Tivoli Storage Manager (以前は ADSM と呼ばれた) 出力を使用することを指定します。

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。

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。 Legato NetWorker は、現在 XBSA インターフェースをサポートしているストレージ・マネージャーです。

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完全パス名を指定しなければなりません。宛先は、データベース・サーバーに常駐していなければなりません。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、およびヒストリー・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。 OS/2 または 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固有のデバイス・ドライバーが必要です。 OS/2 および 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。

テープは OS/2 ではサポートされていません。 OS/2 の場合、`0` または `0:` を指定すると、バックアップ操作はユーザー出口プログラムを呼び出します (457 ページの『付録H. データベース回復用のユーザー出口』 を参照)。ユーザー出口プログラムを使用したオンライン・データベース・バックアップ操作が開始される前に、データベースが静止します。バックアップ・ユーティリティーは、すべてのトランザクションがコミットされるかロールバックされるまで待機します。ユーティリティーの実行中は、すべての新しいトランザクションはバックアップ操作が完了するまで待機します。

テープ・システムでバックアップ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復元 I/O 関数を含む共用ライブラリー (OS/2 または 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。デフォルトは 2 です。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

バックアップ・イメージを構築するときに使用する、 4 KB ページのバッファーのサイズです。このパラメーターの最小値は 8 ページです。デフォルトは 1024 ページです。バッファー・サイズがゼロに指定されている場合、データベース・マネージャー構成パラメーターの値 *backbufsz* が、バッファー割り振りサイズとして使用されます。

可変長のブロック・サイズを使用する場合は、バッファー・サイズを小さくして、テープ装置でサポートされている範囲にしてください。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回復不能になります。

SCO UnixWare 7 上でテープ装置を使用する際には、バッファー・サイズを 16 に指定してください。

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、以下の式を使用してください。

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

bufferpages は *backbufsz* または *restbufsz* のいずれかの値で、 ST_MAX_BUFFERS および 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。デフォルト値は 1 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。

Backup Database API

C API 構文

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

汎用 API 構文

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

API パラメーター

DbAliasLen
入力。データベース別名の長さを示す 2 バイトの無符号整数 (バイト単位) です。

UserNameLen
入力。ユーザー名の長さを示す 2 バイトの無符号整数 (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。

PasswordLen
入力。パスワードの長さを示す 2 バイトの無符号整数 (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。

pReserved1
将来の利用のために予約済み。

pDbAlias

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。

BufferSize

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。デフォルトは 1024 単位です。

BackupMode

入力。バックアップ・モードを指定します。有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLUB_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。

SQLUB_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: DB2 バックアップ・ユーティリティーでは、 SMS LOB オブジェクトに S ロックを掛け、それ以外のすべてのオブジェクトに IN ロックを掛けるため、 sysibm.systables に IX ロックが掛かっていると、オンライン・バックアップ操作がタイムアウトになる場合があります。

BackupType

入力。実行するバックアップのタイプを指定します。有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLUB_FULL

全 (非累積) データベース・バックアップ操作を指定します。この値は、将来サポートされなくなる予定です。全データベース・バックアップ操作を指定するには、SQLUB_DB を使用してください。

SQLUB_DB

データベース内のすべての表スペースのバックアップを指定します。

SQLUB_TABLESPACE

表スペース・レベルのバックアップ操作を指定します。
pTablespaceList パラメーターで、バックアップを取る表スペースのリストを指定してください。

SQLUB_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。

SQLUB_DELTA
非累積 (デルタ) バックアップ・イメージを指定します。デルタ・バックアップ・イメージは、最新の正常実行された任意のタイプの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。

CallerAction
入力。実行するアクションを指定します。有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLUB_BACKUP
バックアップ操作を開始します。

SQLUB_NOINTERRUPT
バックアップ操作を開始します。バックアップ操作を無人で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、バックアップ操作に必要なメディアがすべて装てん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。

SQLUB_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を実行した後で、バックアップ操作を継続します (たとえば、新しいテープの装てん後に継続します)。

SQLUB_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、バックアップ操作を終了します。

SQLUB_DEVICE_TERMINATE
バックアップ・ユーティリティー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。特定のメディアが満杯になると、バックアップ・ユーティリティー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・ユーティリティーを再び起動することによって、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。

SQLUB_PARM_CHECK
バックアップ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 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。

SQLUB_PARM_CHECK_ONLY
バックアップ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。

pApplicationId
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・モニターを使用して、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。

pTimestamp
出力。長さ SQLU_TIME_STAMP_LEN+1 (sqlutil 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。

NumBuffers
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。

pTablespaceList
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操作の場合にのみ必要です。 106ページの『データ構造: SQLU-TABLESPACE-BKRST-LIST』を参照してください。

pMediaTargetList
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、*media_type* フィールドの値によって異なります。 *media_type* についての有効な値 (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が可能です)。 *sqlu_media_entry* 構造のリストを提供してください。 OS/2 または Windows オペレーティング・システムでは、項目をテープ装置名ではなく、ディレクトリー・パスだけにすることができます。

SQLU_TSM_MEDIA
TSM。 *sqlu_media_entry* 構造によってバックアップ・イメージのパスを指定していない場合は、 *sqlu_media_list_targets* 構造の *media* ポインターをヌルに初期設定してください。 DB2 に付属の 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。 *sqlu_vendor* 構造に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 (OS/2 でのみ使用可能です)。

102ページの『データ構造: SQLU-MEDIA-LIST』を参照してください。

pUserName
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。

pPassword
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。

pReserved2
将来の利用のために予約済み。

VendorOptionsSize
入力。 *pVendorOptions* 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。

pVendorOptions
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、バイト逆転が行われず、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。

Parallelism
入力。並列処理度 (バッファー・マニピュレーターの数) を指定します。

pBackupSize
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。ヌルに設定することができます。

pReserved4
将来の利用のために予約済み。

pReserved3
将来の利用のために予約済み。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

REXX API 構文

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

[TABLESPACE :tablespacenames] [ONLINE]

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

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

REXX API パラメーター

dbalias
バックアップされるデータベースの別名。

value データベースのバックアップ情報が戻される複合 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 変数内のエレメントの数 (常に 2)。

XXX.1 バックアップ・イメージのタイム・スタンプ。

XXX.2 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アプリケーション ID。

username
データベースのバックアップに使用されるユーザー名を識別します。

password
ユーザー名の認証に使用されるパスワード。

tablespacenames
バックアップされる表スペースのリストを含む複合 REXX ホスト変数。以下の項目において、XXX はホスト変数の名前を表しています。

XXX.0 バックアップされる表スペースの数。

XXX.1 最初の表スペース名。

XXX.2 2 番目の表スペース名。

XXX.3 3 番目の名前。以下同様に続く。

vendor-library
使用されるベンダーの入出力バックアップ / 復元関数を含む共用ライブラリー (Windows オペレーティング・システムまたは OS/2 では DLL) の名前。絶対

パスで指定することができます。全パスが指定されない場合は、デフォルトとして、ユーザー出口プログラムが存在するパスが使用されます。

vendor-options
ベンダー関数によって必要とされる情報。

num-sessions
TSM またはベンダー製品と共に使用する入出力セッションの数。

target-area
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が可能です。 102ページの『データ構造: SQLU-MEDIA-LIST』のリストを提供してください。 OS/2 または Windows オペレーティング・システムでは、項目をテープ装置名ではなく、ディレクトリー・パスだけにすることができます。

caller-action
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。

SQLUB_BACKUP
バックアップ操作を開始します。

SQLUB_NOINTERRUPT
バックアップ操作を開始します。バックアップ操作を無人で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、バックアップ操作に必要なメディアがすべて装てん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。

SQLUB_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を実行した後で、バックアップ操作を継続します (たとえば、新しいテープの装てん後に継続します)。

SQLUB_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、バックアップ操作を終了します。

SQLUB_DEVICE_TERMINATE
バックアップ・ユーティリティー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。特定のメディアが満杯になると、バックアップ・ユーティリティー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・ユーティリティーを再び起動することによって、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。

SQLUB_PARM_CHECK
バックアップ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。

num-buffers
使用されるバックアップ・バッファーの数。

buffer-size
4 KB の割り振り単位のバックアップ・バッファー・サイズ。最小値は 8 単位です。

parallelism-degree
並列処理度 (バッファー・マニピュレーターの数) を指定します。

データ構造: SQLU-MEDIA-LIST

この構造は、以下の目的で使用されます。

- バックアップ・イメージの宛先 メディアのリストを保持する (93ページの『Backup Database API』を参照)。
- バックアップ・イメージのソース・メディアのリストを保持する (122ページの『Restore Database API』を参照)。
- DB2 ロード・ユーティリティーに情報を渡す。

表 1. SQLU-MEDIA-LIST 構造のフィールド

フィールド名	データ・タイプ	説明
MEDIA_TYPE	CHAR(1)	メディアのタイプを示す文字。
SESSIONS	INTEGER	この構造の *target* フィールドにより示される配列中のエレメントの数を示します。
TARGET	Union	このフィールドは、3 つの構造タイプのうちの 1 つを指すポインターです。示される構造のタイプは、*media_type* フィールドの値によって判別されます。このフィールドに提供する値の詳細については、適切な API を参照してください。

表 2. SQLU-MEDIA-LIST-TARGETS 構造のフィールド

フィールド名	データ・タイプ	説明
MEDIA	ポインター	*sqlu_media_entry* 構造を指すポインター。
VENDOR	ポインター	*sqlu_vendor* 構造を指すポインター。
LOCATION	ポインター	*sqlu_location_entry* 構造を指すポインター。

表 3. SQLU-MEDIA-ENTRY 構造のフィールド

フィールド名	データ・タイプ	説明
RESERVE_LEN	INTEGER	*media_entry* フィールドの長さ。 C 以外の言語用。
MEDIA_ENTRY	CHAR(215)	バックアップおよび復元ユーティリティーが使用するバックアップ・イメージのパス。

表 4. SQLU-VENDOR 構造のフィールド

フィールド名	データ・タイプ	説明
RESERVE_LEN1	INTEGER	*shr_lib* フィールドの長さ。 C 以外の言語用。
SHR_LIB	CHAR(255)	ベンダーにより提供される、データの保管または検索用の共用ライブラリーの名前。
RESERVE_LEN2	INTEGER	*filename* フィールドの長さ。 C 以外の言語用。
FILENAME	CHAR(255)	共用ライブラリーの使用時にロード入力ソースを識別するファイル名。

表 5. SQLU-LOCATION-ENTRY 構造のフィールド

フィールド名	データ・タイプ	説明
RESERVE_LEN	INTEGER	*location_entry* フィールドの長さ。 C 以外の言語用。
LOCATION_ENTRY	CHAR(256)	ロード・ユーティリティー用の入力データ・ファイルの名前。

MEDIA_TYPE 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)

SQLU_SERVER_LOCATION
サーバー装置 (テープ、ディスク、またはディスケット。ロード専用)。 *pDataFileList* パラメーターにのみ指定できます。

SQLU_TSM_MEDIA
TSM

SQLU_OTHER_MEDIA
ベンダー・ライブラリー

SQLU_USER_EXIT
ユーザー出口 (OS/2 のみ)

SQLU_PIPE_MEDIA
名前付きパイプ (ベンダー API のみ)

SQLU_DISK_MEDIA
ディスク (ベンダー API のみ)

SQLU_DISKETTE_MEDIA
ディスケット (ベンダー API のみ)

SQLU_TAPE_MEDIA
テープ (ベンダー API のみ)

言語構文

C 構造

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


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


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


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

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

COBOL 構造

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



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



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



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

データ構造: SQLU-TABLESPACE-BKRST-LIST

この構造は、表スペース名のリストを提供するのに使用されます。

表 6. SQLU-TABLESPACE-BKRST-LIST 構造のフィールド

フィールド名	データ・タイプ	説明
NUM_ENTRY	INTEGER	*tablespace* フィールドによって示されるリストにある項目の数。
TABLESPACE	ポインター	*sqlu_tablespace_entry* 構造を指すポインター。

表 7. SQLU-TABLESPACE-ENTRY 構造のフィールド

フィールド名	データ・タイプ	説明
RESERVE_LEN	INTEGER	*tablespace_entry* フィールドで提供される文字ストリングの長さ。 C 以外の言語用。
TABLESPACE_ENTRY	CHAR(19)	表スペース名。

言語構文

C 構造

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


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

COBOL 構造

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


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

バックアップ・セッションの例

CLP の例

```
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
```

回復可能データベースの場合の、週単位の増分バックアップ戦略の例を以下に示します。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 (デルタ) 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
```

DB2 コマンド・スクリプトの例と、その使用法に関する情報は、443ページの『付録F. 回復 CLP スクリプト』を参照してください。

API の例

DB2 API と組み込み SQL 呼び出しを含むサンプル・プログラムと、その使用法に関する情報は、377ページの『付録E. 回復サンプル・プログラム』を参照してください。

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。

- 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、後で回復することも可能です。この作業を行うと、テーブル・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際に開始される処理またはスレッドの数を定義します。それぞれの処理またはスレッドが、特定の表スペースを割り当てられます。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。したがって、負荷の大きいシステムでは PARALLELISM パラメーターをデフォルト値の 1 のままにしておいてください。

- 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。

- バッファー数を増やします。

  複数のバッファーと入出力チャネルを使用する場合、少なくともチャネルの 2 倍のバッファーを使用し、チャネル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。

バックアップに関する制約事項

バックアップ・ユーティリティーには、以下の制約事項が適用されます。

- 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復元操作とを同時に実行することはできません。
- 区分データベース環境でロールフォワード回復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区画サーバーで、データベース区画のバックアップ・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区画サーバーを追加し、このデータベース区画サーバーについて順方向回復を実行する必要がある場合。
  - 特定時点の回復を使用する場合。この場合は、システム内のすべてのデータベース区画がロールフォワード保留状態でなければなりません。

バックアップに関するトラブルシューティング

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 (バックアップ保留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保留状態でない限り、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。

復元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復元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復元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一時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。

第3章 データベース復元

以下に、DB2 UDB 復元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。

復元の概要

DB2 RESTORE DATABASE コマンドの最も単純な形式の場合、必要なのは、復元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、次のようになります。

```
db2 restore db sample
```

この例では、SAMPLE データベースが存在するので、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、SAMPLE データベースのバックアップ・イメージがある場合は、復元操作は正常に完了するはずです。

データベースを復元操作する際には、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、

復元ユーティリティーを実行すると、復元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、表スペースの復元操作は、オンラインで行うことができます。

(ロールフォワード回復後の) 復元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復元する必要があります。

部分またはサブセット復元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。

復元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* データベースに復元するときには、SYSADM、SYSCTRL、または SYSMAINT の権限が必要です。*新規の* データベースに復元するには、 SYSADM または SYSCTRL 権限が必要です。

復元の使用法

復元を使用する前に

既存の データベースに復元する場合は、復元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復元ユーティリティーにより自動的に確立され、この接続は復元操作が完了すると終了します。*新規の* データベースに復元する場合は、データベースを作成するには、インスタンス接続が必要です。*新規のリモート*・データベースに復元する場合は、まず新規データベースを置くインスタンスに接続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。

復元の起動

復元ユーティリティーは、以下の方法で起動できます。

- コマンド行プロセッサー (CLP)。

  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
  ```

- コントロール・センターの「データベースの復元 (Restore Database)」ノートブックまたはウィザード。「データベースの復元 (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「データベースの復元 (Restore Database)」または「ウィザードを使用したデータベースの復元 (Restore Database Using Wizard)」を選択します。「データベースの復元 (Restore Database)」ノートブックまたは「データベースの復元 (Restore Database)」ウィザードがオープンします。

  コントロール・センターについての一般情報は、*管理の手引き* を参照してください。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。

- アプリケーション・プログラミング・インターフェース (API) としての **sqlrestore**。この API については、122ページの『Restore Database API』を参照してください。 DB2 管理用 API を含むアプリケーションの作成についての一般情報は、 *アプリケーション構築の手引き* を参照してください。

復元操作 (リダイレクト復元) 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。復元操作時に、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検査が行われ、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、復元操作は失敗します。このような場合でも復元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「データベースの復元 (Restore Database)」ノートブックの「コンテナー (Containers)」ページを使用します。コマンド行プロセッサー、コマンド・スクリプト、またはアプリケーショ

ン・プログラミング・インターフェースを使用したリダイレクト復元の例については、133ページの『復元セッションの例』を参照してください。

リダイレクト復元操作中に、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、リダイレクト復元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。同様に、DMS 表スペースをファイル・コンテナーから装置コンテナーに移動することもできます。

既存データベースへの復元

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復元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。バックアップ・イメージが異なったデータベース・シードを持っている場合でも、シードは復元操作後も変わりません。 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復元する場合は、復元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。
- 復元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・ヒストリー・ファイルを保持する (損傷している場合を除く)。リカバリー・ヒストリー・ファイルが損傷を受けている場合は、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。

– イメージが回復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復元

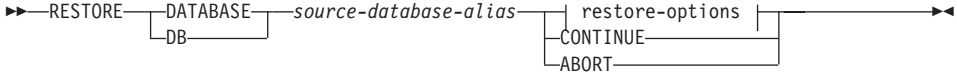
新規のデータベースを作成して、全データベースのバックアップ・イメージをそのデータベースに復元することもできます。バックアップ・イメージとターゲット・データベースのコード・ページは一致していなければなりません。

新規データベースに復元する場合は、復元ユーティリティーにより以下の機能が実行されます。

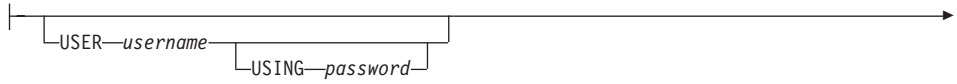
- ターゲット・データベースの別名パラメーターで指定されたデータベース別名を使用して、新規データベースを作成する。 (ターゲット・データベース別名が指定されていない場合は、復元ユーティリティーで、元のデータベース別名パラメーターで指定されているものと同じ別名のデータベースが作成されます。)
- データベース構成ファイルをバックアップ・イメージから復元する。
- RESTORE DATABASE コマンドで NEWLOGPATH が指定されている場合、NEWLOGPATH を *logpath* データベース構成パラメーターの値に設定する。ログ・パスの妥当性検査が行われ、データベースがそのログ・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復元する。
- 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復元する。
- データベースのリカバリー・ヒストリー・ファイルを復元する。

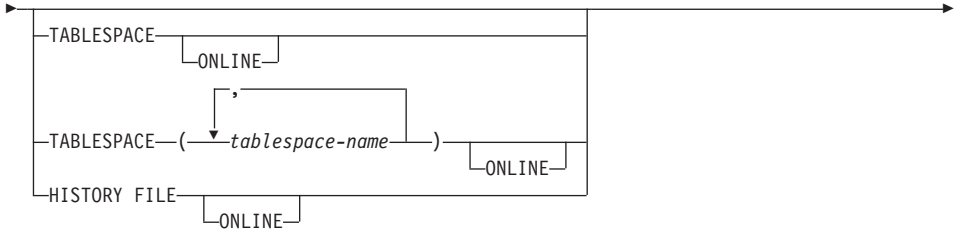
RESTORE DATABASE コマンド

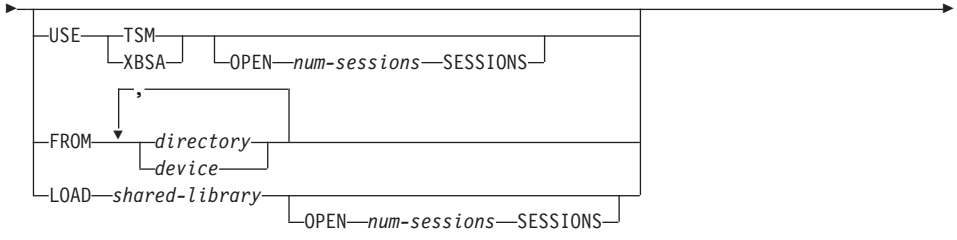
コマンド構文

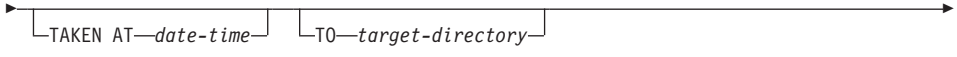


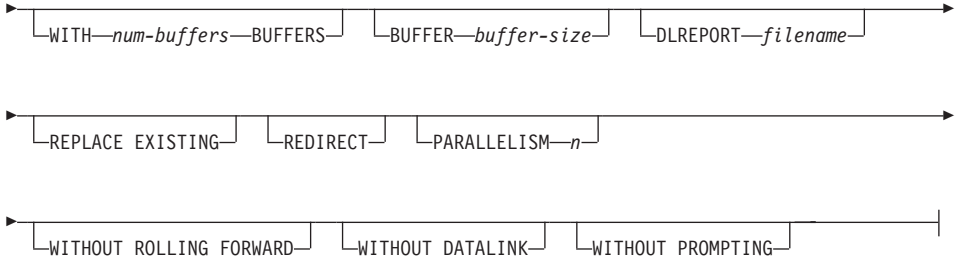
restore-options:











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復元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。

- リダイレクトした復元操作を停止します。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、リダイレクトした復元操作の各ステップを繰り返す必要があります。
- 完了する前に増分復元操作を終了します。

USER username
データベースが復元される際のユーザー名を識別します。

USING password
ユーザー名の認証に使用するパスワードです。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されます。

TABLESPACE tablespace-name
復元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・レベルの復元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・イメージが復元できます。これは、他のエージェントが、バックアップ・イメージの復元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復元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・イメージからヒストリー・ファイルのみを復元するのに指定されます。

INCREMENTAL
手動の累積復元操作を指定します。この場合、ユーザーがそれぞれの復元コマンドを手動で発行する必要があります。

AUTOMATIC/AUTO
自動累積 (増分) 復元操作を指定します。

USE TSM
データベースが TSM 管理の出力から復元されるように指定します。

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。 Legato NetWorker は、現在 XBSA インターフェースをサポートしているストレージ・マネージャーです。

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。 USE TSM、FROM、および LOAD が省略される場合には、デフォルト値は現行ディレクトリーです。

Windows オペレーティング・システムまたは OS/2 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
```

DB2 は c:¥backup ディレクトリーにサブディレクトリーを生成しますが、これらは無視されます。復元するバックアップ・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以下のとおりです。

c 続行。警告メッセージを生成した装置をそのまま使用し続けます (たとえば、新しいテープをマウントした場合に継続するなど)。

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、復元操作を異常終了します。

テープは OS/2 ではサポートされていません。 OS/2 の場合、`0` または `0:` を指定すると、復元ユーティリティーがユーザー出口プログラムを呼び出します。 (これは、ユーザー出口プログラムがデータベースをバックアップするのに使用されていた場合にのみ生じます。) ユーザー出口プログラムから復元を実行するときは、データベースへのパスだけが、コンテナーを探すために使用する唯一の参照パスとなります。したがって、そのデータベースのすべてのコンテナーが復元されます。

ユーザー出口プログラムが使用されているときは、リダイレクト復元はできません。

LOAD shared-library
使用するベンダー・バックアップおよび復元 I/O 関数を含む共用ライブラリー (Windows オペレーティング・システムまたは OS/2 では DLL) の名前です。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 19971001010101 および 19971002010101 で指定されるバックアップ・イメージが存在する場合、 `19971002` を指定することで、タイム・スタンプ `19971002010101` 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復元している場合には、このパラメーターは無視されます。

注: Windows オペレーティング・システムまたは OS/2 上では、このパラメーターを使用する場合はドライブ文字だけを指定してください。それより長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。

NEWLOGPATH directory

復元操作の後にアクティブ・ログ・ファイルに使用されるディレクトリーの完全修飾名です。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、これを指定した復元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、復元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。デフォルト値は 2 です。ただし、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

復元操作に使用するバッファーのサイズ (ページ数)。このパラメーターの最小値は 8 ページです。デフォルトは 1024 ページです。バッファー・サイズがゼロに指定されている場合、データベース・マネージャー構成パラメーターの値 *restbufsz* が、バッファー割り振りサイズとして使用されます。

復元バッファー・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。

磁気テープ装置を SCO UnixWare 7 上で使用するときは、バッファー・サイズを 16 に指定してください。

DLREPORT filename

ファイル名を指定する場合は、完全修飾にしなければなりません。復元操作中に (高速調整の結果として) リンク解除されるファイルが報告されます。このオプションが使用されるのは、復元する表に 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 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、復元ユーティリティーが既存のデータベースを復元したデータベースに置換するように指定します。これは復元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

リダイレクトした復元操作を指定します。リダイレクトした復元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復元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、正常に復元された後ロールフォワード保留状態にしないように指定します。

正常な復元に続いて、データベースがロールフォワード保留状態にある場合には、 150ページの『ROLLFORWARD DATABASE コマンド』 を起動してからでなければデータベースを使用できません。

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
復元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。デフォルト値は 1 です。

WITHOUT PROMPTING
復元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。

Restore Database API

C API 構文

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

汎用 API 構文

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

API パラメーター

SourceDbAliasLen

入力。ソース・データベースの別名の長さを示す 2 バイトの無符号整数 (バイト単位)。

TargetDbAliasLen
入力。ターゲット・データベースの別名の長さを示す 2 バイトの無符号整数 (バイト単位)。ターゲット・データベースの別名が指定されていない場合は、ゼロに設定してください。

TimestampLen
入力。タイム・スタンプの長さを示す 2 バイトの無符号整数 (バイト単位)。タイム・スタンプが提供されていない場合は、ゼロに設定してください。

TargetPathLen
入力。ターゲット・ディレクトリーの長さを示す 2 バイトの無符号整数 (バイト単位)。ターゲット・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。

UserNameLen
入力。ユーザー名の長さを示す 2 バイトの無符号整数 (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。

PasswordLen
入力。パスワードの長さを示す 2 バイトの無符号整数 (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。

ReportFileLen
入力。レポート・ファイル名の長さを示す 2 バイトの無符号整数 (バイト単位)。レポート・ファイル名が提供されていない場合は、ゼロに設定してください。

Reserved2Len
入力。予約済み領域の長さを示す 2 バイトの無符号整数 (バイト単位)。ゼロに設定してください。

pSourceDbAlias
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。

pTargetDbAlias
入力。ターゲット・データベースの別名を含むストリング。このパラメーターの値が NULL の場合、値 *pSourceDbAlias* が使用されます。

BufferSize
入力。バックアップ・バッファー・サイズ (4 KB の割り振り単位 (ページ))。最小値は 8 単位です。デフォルト値は 1024 単位です。

指定するバッファー・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。

RollforwardMode

入力。復元の終了時に、データベースをロールフォワード保留状態にするかどうかを指定します。有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLUD_ROLLFWD
正常に復元された後で、データベースをロールフォワード保留状態にします。

SQLUD_NOROLLFWD
正常に復元された後で、データベースがロールフォワード保留状態にならないようにします。

正常な復元に続いて、データベースがロールフォワード保留状態にある場合には、 156ページの『Rollforward Database API』 を起動してからでなければデータベースを再び使用することはできません。

DatalinkMode

入力。 DATALINK 列を持つ任意の表を DataLink_Reconcile_Pending (DRP) 状態にするかどうか、またリンクされたファイルの調整を実行するかどうかを指定します。有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLUD_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。

SQLUD_NODATALINK
調整操作を実行しません。 DATALINK 列を含む表は DRP 状態になります。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。

RestoreType

入力。復元操作のタイプを指定します。有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLUD_FULL
バックアップ・イメージからすべてのものを復元します。これはオフラインで実行されます。この値は、将来サポートされなくなる予定です。 SQLUD_DB を使用して、完全データベース復元操作を指定してください。

SQLUD_DB
データベースにあるすべての表スペースを復元します。これはオフラインで実行されます。

SQLUD_ONLINE_TABLESPACE
表スペース・レベルのバックアップ・イメージのみを復元します。これはオンラインで実行されます。この値は、将来サポートされなくな

る予定です。 SQLUD_TABLESPACE | SQLUD_ONLINE を使用して、オンラインの表スペース・レベル復元操作を指定してください。

SQLUD_TABLESPACE
表スペース・レベルのバックアップ・イメージのみを復元します。これはオンラインでも、オフラインでも実行されます。

SQLUD_HISTORY
リカバリー・ヒストリー・ファイルだけを復元します。

SQLUD_INCREMENTAL
手動の累積復元操作を実行します。

SQLUD_AUTOMATIC
自動累積 (増分) 復元操作を実行します。 SQLUD_INCREMENTAL で指定する必要があります。つまり、SQLUD_INCREMENTAL | SQLUD_AUTOMATIC とします。

RestoreMode
入力。復元操作がオフラインとオンラインのどちらで実行されるかを指定します。有効な値 (sqlutil で定義される) は、以下のとおりです。

SQLUD_OFFLINE
オフライン復元操作を実行します。

SQLUD_ONLINE
オンライン復元操作を実行します。

CallerAction
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 (sqlutil で定義される) は、以下のとおりです。

SQLUD_RESTORE
復元操作を開始します。

SQLUD_TERMINATE_INCREMENTAL
完了する前に増分復元操作を終了します。

SQLUD_NOINTERRUPT
復元操作を開始します。復元操作を無人で実行するように指定します。通常ユーザー介入を必要とするシナリオは、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、復元操作に必要なメディアがすべて装てん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。

SQLUD_CONTINUE
警告メッセージを生成した装置をそのまま使用し続けます (たとえば、新しいテープをマウントした場合に継続するなど)。

SQLUD_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を実行しなかった場合、復元操作を異常終了します。

SQLUD_DEVICE_TERMINATE
復元ユーティリティーによって使用される装置のリストから装置を除外します。ある装置が入力でいっぱいになると、復元ユーティリティーは呼び出し側に警告を戻します。この呼び出し側アクションで復元ユーティリティーを再び起動します。警告を生成した装置が、使用されている装置のリストから除去されます。

SQLUD_PARM_CHECK
復元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。

SQLUD_RESTORE_STORDEF
初期呼び出し。表スペース・コンテナーの再定義が要求されます。

復元 API の最初の呼び出しでは、 *CallerAction* を必ず SQLUD_RESTORE、SQLUD_NOINTERRUPT、 SQLUD_RESTORE_STORDEF、または SQLUD_PARM_CHECK に設定してください。

pApplicationId
出力。長さ SQLU_APPLID_LEN+1 (sqlutil で定義) のバッファーを提供します。復元ユーティリティーは、アプリケーションにサービスしているエージェントを識別するストリングを戻します。データベース・システム・モニター API と共に使用して、アプリケーションをモニターすることができます。

pTimestamp
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。

pTargetPath
入力。ターゲット・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。復元操作中に新規データベースが作成される場合に使用します。

NumBuffers
入力。復元操作に使用するバッファーの数。

pReportFile
ファイル名を指定する場合は、完全修飾にしなければなりません。復元操作中に (高速調整の結果として) リンク解除される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、復元する表に 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。

pTablespaceList
復元される 1 つまたは複数の表スペースを指定します。表スペース・レベル・

バックアップ・イメージから、バックアップ・イメージのサブセットまたは表スペースを復元する場合に使用されます。

以下の制限が適用されます。

- データベースは回復可能である必要があります。回復可能データベースでは、 *logretain* データベース構成パラメーターが 『RECOVERY』 に設定されているか、 *userexit* データベース構成パラメーターが使用可能になっています。または、この両方が当てはまっています。
- 復元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、表スペース復元機能を使用して、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。
- この機能は、OS/2 上のユーザー出口からの復元時には利用できません。
- ロールフォワード・ユーティリティーは、MPP 環境で復元される表スペースが、同じ表スペースを含む他のノードと必ず同期化されるようにします。

注: バックアップを実行した後に名前変更した表スペースを復元する場合、復元ユーティリティーを起動するときには変更後の新しい表スペース名を使用する必要があります。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。

pMediaSourceList

入力。バックアップ・イメージのソース・メディア。 102ページの『データ構造: SQLU-MEDIA-LIST』を参照してください。呼び出し側がこの構造に提供する必要のある情報は、 *media_type* フィールドの値によって異なります。このフィールドの有効な値 (`sqlutil` で定義される) 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_media_entry* 構造のリストを提供してください。 Windows オペレーティング・システムまたは OS/2 では、項目はディレクトリー・パスのみです。テープ装置名ではありません。

SQLU_TSM_MEDIA

TSM。他に入力は必要ありません。 DB2 に付属の 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 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。 *sqlu_vendor* 構造に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 (OS/2 でのみ使用可能です)。

pUserName
入力。接続に使用されるユーザー名を含むストリング。

pPassword
入力。ユーザー名の認証に使用するパスワードを含むストリング。

pReserved2
将来の利用のために予約されています。

VendorOptionsSize
入力。ベンダーのオプション・フィールドの長さ。 65535 バイトを超えることはできません。

pVendorOptions
入力。情報をアプリケーションからベンダー関数に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、間接参照のレベルはサポートされません。このデータについては、バイト逆転が行われず、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。

Parallelism
入力。区画内並列処理度 (バッファー・マニピュレーターの数)。

RestoreInfoSize
将来の利用のために予約されています。

pRestoreInfo
将来の利用のために予約されています。

ContainerPageListSize
将来の利用のために予約されています。

pContainerPageList
将来の利用のために予約されています。

pReserved3
将来の利用のために予約されています。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

REXX API 構文

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

REXX API パラメーター

source-database-alias
データベースのバックアップ・イメージが取られたソース・データベースの別名。

value データベースの復元情報が戻される複合 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 変数内のエレメント数 (常に 1)

XXX.1 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アプリケーション ID。

username
接続に使用されるユーザー名を識別します。

password
ユーザー名の認証に使用するパスワード。

tablespacenames
復元される表スペースのリストを含む複合 REXX ホスト変数。以下の項目において、XXX はホスト変数の名前を表しています。

XXX.0 復元される表スペースの数

XXX.1 最初の表スペース名

XXX.2 2 番目の表スペース名

XXX.3 3 番目の名前。以下同様に続く。

HISTORY FILE
バックアップ・イメージからヒストリー・ファイルのみを復元するように指定します。

shared-library
使用するベンダーの復元 I/O 関数を含む共用ライブラリー (Windows オペレーティング・システムまたは OS/2 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。絶対パスが指定されない場合は、デフォルトとして、ユーザー出口プログラムが存在するパスが使用されます。

vendor-options
ベンダー関数によって必要とされる情報。

num-sessions
TSM またはベンダー製品と共に使用する入出力セッションの数。

source-area
バックアップ・イメージが存在するディレクトリーまたは装置を示す複合

REXX ホスト変数。デフォルト値は現行ディレクトリーです。 Windows オペレーティング・システムまたは OS/2 では、項目はディレクトリー・パスのみです。テープ装置名ではありません。

timestamp
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。

target-directory
ターゲット・データベース・ディレクトリーです。

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。

caller-action
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。

SQLUD_RESTORE
復元操作を開始します。

SQLUD_NOINTERRUPT
復元操作を開始します。復元操作を無人で実行するように指定します。通常ユーザー介入を必要とするシナリオは、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、復元操作に必要なメディアがすべて装てん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。

SQLUD_CONTINUE
警告メッセージを生成した装置をそのまま使用し続けます (たとえば、新しいテープをマウントした場合に継続するなど)。

SQLUD_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を実行しなかった場合、復元操作を異常終了します。

SQLUD_DEVICE_TERMINATE
復元ユーティリティーによって使用される装置のリストから装置を除外します。ある装置が入力でいっぱいになると、復元ユーティリティーは呼び出し側に警告を戻します。この呼び出し側アクションで復元ユーティリティーを再び起動します。警告を生成した装置が、使用されている装置のリストから除去されます。

SQLUD_PARM_CHECK
復元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。

SQLUD_RESTORE_STORDEF
初期呼び出し。表スペース・コンテナーの再定義が要求されます。

num-buffers
使用されるバックアップ・バッファーの数。

buffer-size
バックアップ・バッファー・サイズ (4KB の割り振り単位)。最小値は 16 単位です。

parallelism-degree
区画内並列処理度 (バッファー・マニピュレーターの数)。

復元セッションの例

CLP の例

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復元のシナリオです。

1. 次のように、REDIRECT オプションを指定して RESTORE DATABASE コマンドを実行する。

   ```
   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、次を発行して復元操作を打ち切ることができる。

   ```
   db2 restore db mydb abort
   ```

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 コマンドを実行する。たとえば、OS/2 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)
   ```

   復元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 コマンドを実行する。

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復元操作の最終ステップです。

4. ステップ 3 が失敗した場合、または復元操作を打ち切った場合、リダイレクト復元はステップ 1 から再始動できる。

回復可能データベースの週ごとの増分バックアップ・ストラテジーの例を以下に示します。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 (デルタ) 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 kdr use tsm
(Mon) backup db kdr online incremental delta use tsm
(Tue) backup db kdr online incremental delta use tsm
(Wed) backup db kdr online incremental use tsm
(Thu) backup db kdr online incremental delta use tsm
(Fri) backup db kdr online incremental delta use tsm
(Sat) backup db kdr 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復元するには、次のようにします。

```
restore db kdr incremental automatic taken at (Thu)
```

復元セッションの例

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復元するには、次のようにします。

```
restore db kdr incremental taken at (Thu)
restore db kdr incremental taken at (Sun)
restore db kdr incremental taken at (Wed)
restore db kdr incremental taken at (Thu)
```

DB2 コマンド・スクリプトの例と、その使用法に関する情報は、443ページの『付録F. 回復 CLP スクリプト』を参照してください。

API の例

DB2 API と組み込み SQL 呼び出しを含むサンプル・プログラムと、その使用法に関する情報は、377ページの『付録E. 回復サンプル・プログラム』を参照してください。

復元のパフォーマンスの最適化

復元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。

- 復元バッファー・サイズを大きくする。

  復元バッファー・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 16 です。

復元に関する制約事項

復元ユーティリティーには、以下の制約事項が適用されます。

- 復元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。
- DB2 コントロール・センターを使用している場合は、以前のバージョンの DB2 を使用して作成したバックアップ・イメージは復元できません。
- ロールフォワード処理の実行中にデータベースの復元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除去して再作成していないこと) に限り、表スペースの復元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復元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・レベルの復元操作をオンラインで実行することはできません。
- OS/2 では、ユーザー出口プログラムを呼び出すときは、部分またはサブセット復元操作は実行できません。
- 復元するデータベースのコード・ページが、アプリケーションで使用可能なコード・ページと一致しない場合、またはデータベース・コード・ページからアプリケーションで使用可能なコード・ページへのコード・ページ変換が、データベース・マネージャーでサポートされていない場合、復元後のデータベースは使用できなくなります。

復元に関するトラブルシューティング

別のシステム上の新しいデータベースにバックアップ・イメージを復元しようとすると SQL0970N が戻される場合は、絶対パスを使用して定義された表スペースが元のデータベース中にあれば、リダイレクト復元操作を行って、新しいシステム上に表スペース・コンテナーを定義してください。復元ユーティリティーは、新しいシステム上に存在しない絶対パスとコンテナーを使用するよう試みます。

データベースの復元操作中にシステム障害が発生したなら、復元ユーティリティーをもう一度起動して、復元操作を正常完了しない限り、データベースに接続することはできません。表スペースの復元操作中にシステム障害が発生した場合、使用できなくなるのは復元中のその表スペースだけです。データベースの他の表スペースは使用できます。

第4章 ロールフォワード回復

ここでは、DB2 UDB エクスポート・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回復します。

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、ロールフォワード回復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 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.
```

これらのフィールドの説明については、150ページの『ROLLFORWARD DATABASE コマンド』 を参照してください。

ロールフォワード回復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、戻された時刻が予定より早いと、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。

   時刻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時刻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復元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保留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復元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保留状態の場合、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保留状態にある場合、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保留状態にない が、表スペースはロールフォワード保留状態にある 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、ロールフォワード保留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、データベースは利用不能です。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、データベースは利用不能です。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、ロールフォワード回復は不可能です。破損回復またはバージョン回復だけが行えます (30ページの『リカバリー・ログについて』 を参照してください)。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回復) を使用可能にするには、 *logretain* データベース構成パラメーターを RECOVERY に設定するか、 *userexit* データベース構成パラメーターを YES に設定します。または、この両方をそのように設定します。これらの両方のパラメーターのデフォルト値は NO です。最初の段階では、データベースを回復するのに使用できるバックアップ・イメージがないからです。片方または両方のパラメーターの値を変更すると、データベースはバックアップ保留状態になり、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。

ロギングに関連するデータベース構成パラメーターの詳細な説明については、
37ページの『データベース・ロギングの構成パラメーター』を参照してください。

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

ロールフォワードを使用する前に

ロールフォワード回復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。

データベースは、ローカルでもリモートでも構いません。

ロールフォワードの起動

ロールフォワード・ユーティリティーは、以下の方法で起動できます。

- コマンド行プロセッサー (CLP)。

  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
  ```

- コントロール・センターの「データベースのロールフォワード (Rollforward Database)」ノートブック。「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。

  コントロール・センターについての一般情報は、*管理の手引き* を参照してください。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。

- アプリケーション・プログラミング・インターフェース (API) としての **sqluroll**。この API については、156ページの『Rollforward Database API』を参照してください。 DB2 管理用 API を含むアプリケーションの作成についての一般情報は、*アプリケーション構築の手引き* を参照してください。

区分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。

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回復であれば、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、復元したり、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回復方針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回復するよりは、データベースの一部を回復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復元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回復する必要はなく、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復元できます。) 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。

表スペースが復元された後は、常にロールフォワード保留状態になります。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回復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時刻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 *最小回復時間* を戻します。最小回復時間は、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、少なくとも最小回復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回復する場合、表スペースは少なくとも、回復されるすべての表スペースの最小回復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。区分データベース環境では、LIST TABLESPACES SHOW DETAIL コマンドをすべての区分で発行してください。表スペースは、少なくとも、すべての区分にあるすべての表スペースの最小回復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、その表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復元し、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要約表の基礎表
- 別の表スペースにある表の要約表

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、検査保留状態で、要約表がロールフォワード操作の最後に置かれます。

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。同時にロールフォワードしないと、時刻指定ロールフォワード操作終了時に、両方の表スペースは検査保留状態になります。両方の表スペースを同時にロールフォワードすると、時刻指定ロールフォワード操作の終了時に制約が活動状態のままになります。

時刻指定の表スペース・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。

- 時刻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、その特定の時刻がトランザクションのコミットされた時刻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指定時刻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機し、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・ヒストリー・ファイルを調べて静止点を見つけ、それらが最小回復時間よりも後に起こったかを確認することができます。

表スペースの時刻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保留状態になります。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、バックアップ保留状態の表スペース、またはバックアップ保留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。)

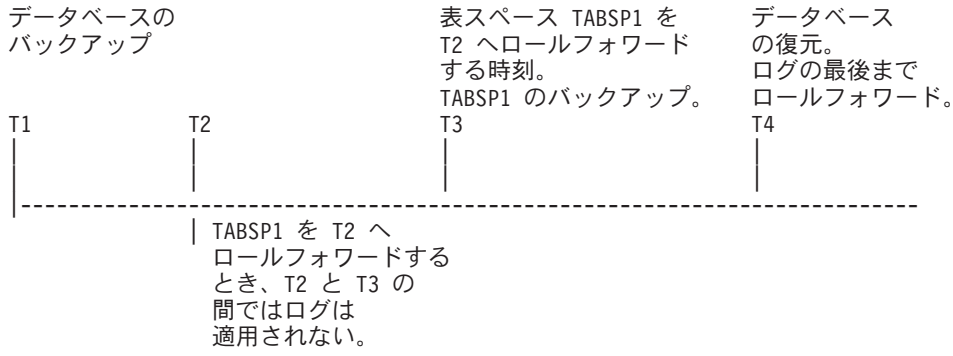


図 17. 表スペースのバックアップ要件

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、時刻 T3 に、表スペース TABSP1 は特定の指定時刻 (T2)までロールフォワードされ、表スペースは時刻 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保留状態であるため、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復元され、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 で復元保留状態になります。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、表スペースが指定時刻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、直前の時刻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 まで回復したい場合は、 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) から表スペースを復元し、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復元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. データベースを復元する。
2. 表スペースを復元する。
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復元するので、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復元したい場合は、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・イメージから復元されているため、表スペースを再び復元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復元し、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。
- 表スペースを除去する。

区分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指定時刻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区画で整合性が保証されます。
- 一部のデータベース区画がロールフォワード保留状態で、さらに他のデータベース区画で一部の表スペースがロールフォワード保留状態である (ただしデータベース区画自体はその状態でない) 場合は、まずデータベース区画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、データベース区画ごとに復元する必要はありません。回復が必要なデータベース区画でのみ復元が必要です。ただし、特定の時刻まで表スペースをロールフォワードしたい場合は、各データベース区画ごとに復元する必要があります。

除去された表の回復

ときに、まだ必要なデータのある表を除去してしまうことがあります。そのような場合は、表除去操作の後でも回復可能なクリティカル表の作成を考慮すると良いでしょう。

表データを、データベース復元操作によって回復し、その後、表が除去される前の時刻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、その回復のあいだはデータは使用できなくなります。

DB2 の、除去された表を回復する機能により、表スペース・レベルの復元およびロールフォワード操作を使用して、除去された表を回復することができます。これはデータベース・レベルの回復より速く、ユーザーもデータベースをそのまま使用できます。

除去された表が回復可能になるには、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 (*SQL 解説書* を参照してください)。 DROPPED TABLE RECOVERY オプションは表スペースに固有で、正規表スペースに限定されます。ある表スペースで、除去された表の回復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。

DROP TABLE ステートメントが、除去された表の除去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、追加のエントリー (除去された表を識別する) がログ・ファイル内に作成されます。エントリーはリカバリー・ヒストリー・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。

一度で回復できる除去された表は 1 つだけです。以下のようにして、除去された表を回復できます。

1. LIST HISTORY DROPPED TABLE コマンドを起動して、除去された表を識別します (331ページの『LIST HISTORY』 を参照してください)。除去された表の ID は、Backup ID 列にリストされます。
2. 表が除去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・イメージを復元します。
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区画からアクセスできるものか、またはそれぞれの区画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、各データベース区画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* はデータベース区画またはノード番号を表します。それぞれのデータベース区画にあったときと同じ、除去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、表が除去された後の特定の時刻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・ヒストリー・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。

除去された表から回復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、回復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。長形式表スペースには 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 列を含む除去された表を回復しようとすると、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 オプションは正規表スペースにのみ使用できるもので、一時または長形式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。 (データは回復されますが、メタデータは回復されません。) タイプ付き表の階層表にあるデータは回復されます。このデータには、除去されたタイプ付き表に現れたよりも詳細にわたる情報が含まれます。

DATALINK 列に関連したリンク・ファイルの名前は回復できます。表データのインポートの後に、表を DB2 データ・リンク・マネージャー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 データ・リンク・マネージャーにより回復されることも回復されないこともあります。

ロード・コピー・ロケーション・ファイルの使用

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. 区分データベース環境では、 DB2LOADREC レジストリー変数は `db2profile` ファイル内に含まれていなければなりません。
2. 区分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区画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エントリーが無効な場合には、古いロード・コピーのロケーション・ファイルが使用され、無効なエントリー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、次のようになります。

```
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
SES            NULL
BUF            NULL
TYP            A
TIM            19940325192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            TORONTO
SES            NULL
BUF            NULL
TYP            O
SHRlib         /@sys/lib/backup_vendor.a
```

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 だけはヌル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・ファイルに無効なエントリーが 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。バックアップ・メディアとして TSM および他のベンダー製品を使うことに関する詳細は、 451ページの『付録G. Tivoli Storage Manager』を参照してください。

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時刻までデータベースや表スペースを復元することはできません。つまり、ロールフォワード回復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復元できるのは、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・ヒストリー・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。区分データベース環境の場合、リカバリー・ヒストリー・ファイルは各データベース区画にローカルに存在します。

ロード・ユーティリティーの詳細については、*データ移動ユーティリティー 手引きおよび解説書* を参照してください。

区分データベース・システムにおいてクロックを同期化する

データベース区画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、順方向回復性に制限が加わらないようにします。データベース区画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、区分データベース・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、データベース区画サーバー 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、そのデータベース区画サーバーの区画で更新トランザクションがコミットされ

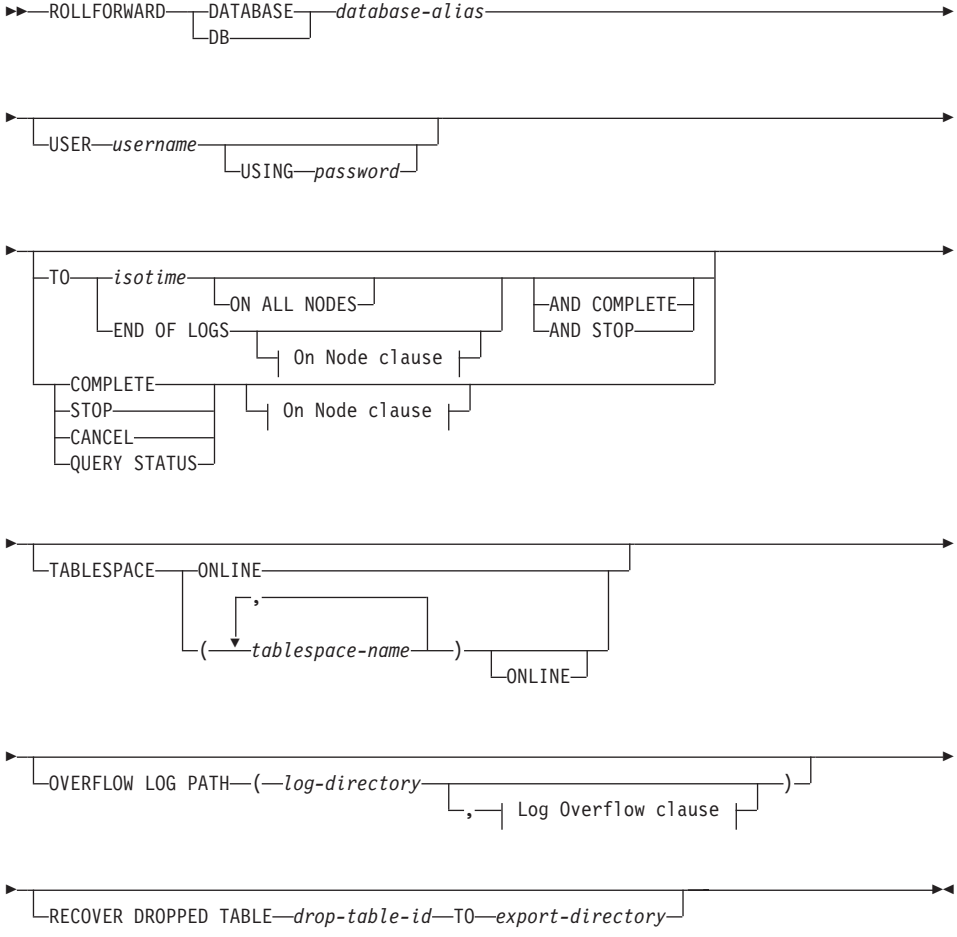
た後 であったとします。そのデータベースが連続的に使用されていてその間で定期的に更新されていると、 1997 年 11 月 7 日から 1999 年 11 月 7 日までの間の任意の時点は、ロールフォワード回復では実際には処理不能になります。データベース区画サーバー A でコミットが行われると、データベース・ログのタイム・スタンプは 1999 に設定され、データベース・ログのクロックは 1999 年 11 月 7 日のままです。この状態は、システム・クロックがこの時間と一致するまで続きます。この時間フレーム内の時点へロールフォワードしようとすると、指定された停止点 (1997 年 11 月 7 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。

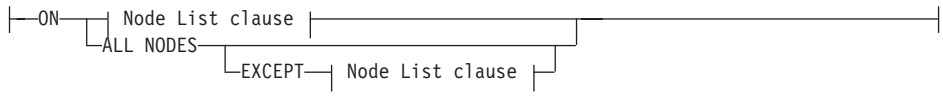
- このパラメーターに指定できる値は、1 分から 24 時間までです。 *max_time_diff* の設定についての詳細は、*管理の手引き: パフォーマンス* を参照してください。
- 非カタログ・ノードに最初の接続要求が出されると、データベース区画サーバーはその時間をデータベースのカタログ・ノードに送信します。カタログ・ノードはその時間を受け取ると、接続を要求するノードの時間と自分自身の時間が、 *max_time_diff* パラメーターで指定された範囲内であることを検査します。この範囲を超えると、接続は拒否されます。
- 更新トランザクションがデータベース内の 3 つ以上のデータベース区画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区画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区画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、他のデータベース区画サーバーへ正しくない時間が伝送されるのを防ぎます。

ROLLFORWARD DATABASE コマンド

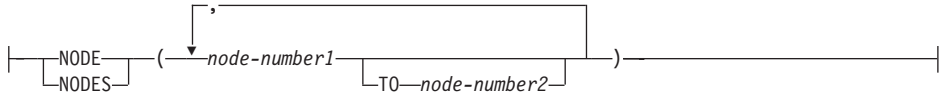
コマンド構文



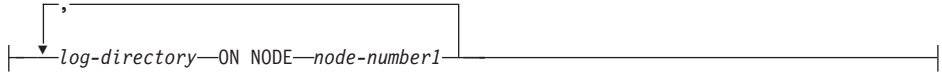
On Node clause:



Node List clause:



Log Overflow clause:



コマンド・パラメーター

DATABASE database-alias
ロールフォワード回復するデータベースの別名。

USER username
データベースをロールフォワード回復する際のユーザー名。

USING password
ユーザー名の認証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されます。

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、バインドされた日時を識別する 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されます。形式は協定世界時 (UTC) で表され、 *yyyy-mm-dd-hh.mm.ss.nnnnnn* (年、月、日、時、分、秒、マイクロ秒) です。 UTC を使用しているので、同じタイム・スタンプが別のログに関連付けられる (たとえば、夏時間調整に関連した時刻変更などが原因) ことはありません。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された地方時に基づいています。 CURRENT TIMEZONE 特殊レジスターが、UTC とアプリケーション・サーバーの地方時の差を指定します。差は時刻期間で表されます (10 進数。このうち最初の 2 桁は時間、次の 2 桁は分、最後の 2 桁は秒数を表す)。地方時から CURRENT TIMEZONE を減算して、地方時を UTC に変換します。

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 NODES
db2nodes.cfg ファイルで指定されている、すべてのノード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。ノード文節が指定されていない場合、これがデフォルトです。

EXCEPT
ノード・リストに指定されているノードを除き、 db2nodes.cfg ファイルで指定されている、すべてのノード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 NODE / ON NODES
一連のノードについてデータベースをロールフォワードします。

node-number1
ノード・リスト内のノード番号を指定します。

node-number2
2 番目のノード番号を指定します。この番号を指定すると、 *node-number1* から *node-number2* までのすべてのノードがノード・リストに含まれます。

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保留状態をオフにすることによって、ロールフォワード回復処理を完了します。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: 表スペースを特定の時刻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保留状態になります。

CANCEL
ロールフォワード回復操作をキャンセルします。これにより、順方向回復が開始されている、すべてのノードのデータベースまたは 1 つ以上の表スペースが回復保留状態になります。

- データベースの ロールフォワード操作が進行中でない (つまり、データベースがロールフォワード保留状態である) 場合、このオプションは、データベースを復元保留状態にします。
- 表スペースの ロールフォワード操作が進行中でない (つまり、表スペースがロールフォワード保留状態である) 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、復元保留状態になります。

- 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状態にあるすべての表スペースが復元保留状態になります。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。リスト内のすべての表スペースが、復元保留状態になります。
- 特定の時刻までロールフォワードするする場合、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状態にある表スペースがすべて復元保留状態になります。
- 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、リストされた表スペースのみが復元保留状態になります。

このオプションは、*実際に実行中* であるロールフォワード操作をキャンセルするのに使用することはできません。進行中ではあってもその時点で実際には実行していないロールフォワード操作をキャンセルすることだけに使用できます。ロールフォワード操作が、進行中であっても実行中ではないのは、以下の場合です。

- 異常終了した。
- STOP オプションが指定されなかった。
- エラーが発生したため失敗した。エラーによっては、たとえば回復不能ロード操作でロールフォワードした場合など、表スペースが復元保留状態になることがあります。

注: このオプションの使用に際しては、注意が必要です。表スペースのいくつかがロールフォワード保留状態または復元保留状態になっているため、進行中のロールフォワード操作を完了できない、という場合にのみ使用してください。それが明らかでない場合は、LIST TABLESPACES コマンドを使用して、ロールフォワード進行状態またはロールフォワード保留状態にある表スペースを識別してください。

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。区分データベース環境では、この状況情報はそれぞれのノードごとに戻されます。返される情報には、以下のフィールドが含まれています。

ノード番号

ロールフォワード状況

状況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

ォワード保留、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 STOP 処理中、あるいは保留なし。

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。区分データベース環境では、ロールフォワード・ユーティリティーに障害が起こり、ログ・ファイルの欠落またはログ情報の不一致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
これ以上回復に必要なく、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、ログ・ファイル *x* 内で最も古いコミットされていないトランザクションが開始した場合、差し替え済みログ・ファイルに *x* は含まれません。範囲は *x* - 1 までで終了します。

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、ロールフォワード回復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回復では、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・スタンプです。

注: TO、STOP、COMPLETE、または CANCEL 文節が省略された場合、QUERY STATUS がデフォルト値になります。 TO、STOP、または COMPLETE が指定された場合、コマンドが正常に完了すれば状況情報が表示されます。個々の表スペースが指定された場合、それらは無視されます。状況要求は指定された表スペースだけに適用されます。

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回復を行う場合に指定します。

tablespace-name
特定の時刻までの表スペース・レベルのロールフォワード回復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最後までのロールフォワード回復を指定できます。区分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ノードに存在している必要はありません。もし存在している 場合、表スペースは適正な状態である必要があります。

ONLINE
このキーワードを指定すると、表スペース・レベルのロールフォワード回復をオンラインで行うことができます。これは、ロールフォワード回復処理の進行中に、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・ファイルを、 *logpath* データベースの構成パラメーターで指定される位置とは別の位置に移動する場合に、このパラメーターを使用してください。区分データベース環境では、これが *すべてのノードの* (完全修飾) デフォルト・オーバーフロー・ログ・パスです。単一区画データベースには、相対オーバーフロー・ログ・パスを指定することができます。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、ログ名が SQLCA に戻され、ロールフォワード回復は停止します。それ以上ログが使用不能の場合、STOP オプションを使用してロールフォワード回復を終了してください。データベースまたは表スペースを整合性のある状態に保つため、未完了のトランザクションはロールバックされます。

log-directory ON NODE

区分データベース環境では、特定のノードに対して異なるログ・パスでデフォルト・オーバーフロー・ログ・パスをオーバーライドできます。

RECOVER DROPPED TABLE drop-table-id

除去された表をロールフォワード操作中に回復します。表 ID を取得するには、331ページの『LIST HISTORY』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、すべてのノードにアクセスできるものでなければなりません。

Rollforward Database API

C API 構文

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

汎用 API 構文

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

SQL_STRUCTURE grfwd_input
{
   unsigned short DbAliasLen,
   unsigned short StopTimeLen,
   unsigned short UserNameLen,
   unsigned short PasswordLen,
   unsigned short OverflowLogPathLen,
   unsigned short ReportFileLen,      /* NOTE: This parameter is no longer used */
                                      /*       for the DB2 Data Links Manager. */
   sqluint32 Version,
   char * pDbAlias,
   unsigned short CallerAction,
   char * pStopTime,
   char * pUserName,
   char * pPassword,
   char * pOverflowLogPath,
   unsigned short NumChngLgOvrflw,
   struct sqlurf_newlogpath * pChngLogOvrflw,
   unsigned short ConnectMode,
   struct sqlu_tablespace_bkrst_list * pTablespaceList,
   short AllNodeFlag,
   short NumNodes,
   SQL_PDB_NODE_TYPE * pNodeList,
   short NumNodeInfo,
   unsigned short DlMode,             /* NOTE: This parameter is no longer used */
                                      /*       for the DB2 Data Links Manager. */
   char * pReportFile,                /* NOTE: This parameter is no longer used */
                                      /*       for the DB2 Data Links Manager. */
   char * pDroppedTblID,
   char * pExportDir
}
/* ... */
```

API パラメーター

pRfwdInput

入力。 *rfwd_input* 構造を指すポインター。この構造の詳細については、165ページの『データ構造: RFWD-INPUT』を参照してください。

pRfwdOutput

出力。 *rfwd_output* 構造を指すポインター。この構造の詳細については、168ページの『データ構造: RFWD-OUTPUT』を参照してください。

DbAliasLen
入力。データベース別名の長さを示す 2 バイトの無符号整数 (バイト単位) です。

StopTimeLen
入力。停止時刻パラメーターの長さを示す 2 バイトの無符号整数 (バイト単位)。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。

UserNameLen
入力。ユーザー名の長さを示す 2 バイトの無符号整数 (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。

PasswordLen
入力。パスワードの長さを示す 2 バイトの無符号整数 (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。

OverflowLogPathLen
入力。オーバーフロー・ログ・パスの長さを示す 2 バイトの無符号整数 (バイト単位)。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。

ReportFileLen
入力。このパラメーターは現在は使用されていないので、ゼロに設定しておく必要があります。

Version
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。

pDbAlias
入力。データベース別名を含むストリングです。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。

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

SQLUM_ROLLFWD
pPointInTime 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回復の場合、データベースはロールフォワード保留状態のままになります。特定の時刻への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進行状態のままになります。

SQLUM_STOP
ロールフォワード回復を終了します。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*ロールフォワード保留* 状態はオフになります。同義語は SQLUM_COMPLETE です。

SQLUM_ROLLFWD_STOP
pPointInTime で指定された時点までロールフォワードし、ロールフォワード回復を終了します。データベースまたは表スペースのロールフォワード保留状態はオフになります。同義語は SQLUM_ROLLFWD_COMPLETE です。

SQLUM_QUERY
pNextArcFileName、 *pFirstDelArcFileName*、 *pLastDelArcFileName*、および *pLastCommitTime* の値を照会します。データベース状況とノード番号を戻します。

SQLUM_PARM_CHECK
ロールフォワード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。

SQLUM_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、回復保留状態に置かれます。

注: このオプションは、ロールフォワード操作が実際に実行中であるときには使用できません。操作が休止されている (つまり、STOP を待っている) 場合、あるいはロールフォワード操作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。

データベースをロールフォワードするには、テープ装置を使用するロード回復操作が必要な場合もあります。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。

SQLUM_LOADREC_CONTINUE
警告メッセージを生成した装置を使用し続けます (たとえば、新しいテープが装てんされた場合)。

SQLUM_LOADREC_DEVICE_TERMINATE
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)。

SQLUM_LOADREC_TERMINATE
ロード回復に使用されているすべての装置を終了させます。

pStopTime
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、データベース回復は停止します。可能なかぎ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。 SQLUM_QUERY、SQLUM_PARM_CHECK、およびいずれかのロード回復 (SQLUM_LOADREC_*xxx*) 呼び出し側アクションの場合は、ヌルにすることができます。

pUserName
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。

pPassword
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。

pOverflowLogPath
入力。使用される代替ログ・パスを指定します。ユーザーは、アクティブ・ログ・ファイルのほかに、アーカイブ・ログ・ファイルを *logpath* (*管理 API 解説書* の『sqlfxdb - データベース構成の入手』 を参照) に移動することが必要です。それからでなければ、このユーティリティーでそれらのファイルを使用することはできません。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。ロールフォワード回復中に、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回復に必要なログ・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。区分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・パスは、各ノードのデフォルト・オーバーフロー・ログ・パスです。単一区画環境では、サーバーがローカルであれば、オーバーフロー・ログ・パスが相対パスであっても構いません。

NumChngLgOvrflw
MPP 専用。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、指定されたノードのデフォルト・オーバーフロー・ログ・パスだけをオーバーライドします。

pChngLogOvrflw
MPP 専用。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、指定されたノードのデフォルト・オーバーフロー・ログ・パスだけをオーバーライドします。

ConnectMode
入力。有効な値 (sqlutil で定義) は、以下のとおりです。

SQLUM_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回復の場合には、必ずこの値を指定してください。

SQLUM_ONLINE
オンライン・ロールフォワード。

pTablespaceList
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。

AllNodeFlag
MPP 専用。入力。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ノードに適用されるかどうかを示します。有効な値は次のとおりです。

SQLURF_NODE_LIST
pNodeList で渡されたノード・リスト内のノードに適用されます。

SQLURF_ALL_NODES
すべてのノードに適用されます。 *pNodeList* はヌルでなければなりません。これがデフォルトです。

SQLURF_ALL_EXCEPT
pNodeList で渡されたノード・リスト内のノードを除く、すべてのノードに適用されます。

SQLURF_CAT_NODE_ONLY
カタログ・ノードにのみ適用されます。 *pNodeList* はヌルでなければなりません。

NumNodes
入力。 *pNodeList* 配列内のノードの数を指定します。

pNodeList
入力。ロールフォワード回復を実行する対象のノード番号の配列を指すポインター。

NumNodeInfo
入力。出力パラメーター *pNodeInfo* 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ノード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。単一区画環境では、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 が呼び出されるノードの数と同じにする必要があります。

DIMode
入力。このパラメーターは現在は使用されていないので、ゼロに設定しておく必要があります。

pReportFile
入力。このパラメーターは現在は使用されていないので、ヌルに設定しておく必要があります。

pDroppedTblID
入力。回復が実行されている消去済み表の ID を含むストリング。

pExportDir
入力。削除した表データのエクスポート先のディレクトリー。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

REXX API 構文

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

REXX API パラメーター

database-alias
ロールフォワードされるデータベースの別名。

value 出力値を含む複合 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	変数内のエレメントの数
XXX.1	アプリケーション ID
XXX.2	ノードから受信された応答の数
XXX.2.1.1	最初のノード番号
XXX.2.1.2	最初のノードの状態情報
XXX.2.1.3	最初のノードの必要とされる次のアーカイブ・ファイル
XXX.2.1.4	最初のノードの削除される最初のアーカイブ・ファイル
XXX.2.1.5	最初のノードの削除される最後のアーカイブ・ファイル
XXX.2.1.6	最初のノードの最後のコミット時刻
XXX.2.2.1	2 番目のノードのノード番号

XXX.2.2.2 2 番目のノードの状態情報

XXX.2.2.3 2 番目のノードの必要とされる次のアーカイブ・ファイル

XXX.2.2.4 2 番目のノードの削除される最初のアーカイブ・ファイル

XXX.2.2.5 2 番目のノードの削除される最後のアーカイブ・ファイル

XXX.2.2.6 2 番目のノードの最後のコミット時刻

XXX.2.3.x 以下同様に続く。

username
データベースのロールフォワードに使用されるユーザー名を識別します。

password
ユーザー名の認証に使用するパスワード。

point-in-time
協定世界時 (UTC) で表現された、 ISO 形式 *yyyy-mm-dd-hh.mm.ss.nnnnnn* (年、月、日、時、分、秒、ミリ秒) のタイム・スタンプ。

tablespacenames
ロールフォワードされる表スペースのリストを含む複合 REXX ホスト変数。以下の項目において、XXX はホスト変数の名前を表しています。

XXX.0 ロールフォワードされる表スペースの数

XXX.1 最初の表スペース名

XXX.2 2 番目の表スペース名

XXX.x 以下同様に続く。

default-log-path
回復中にアーカイブ・ログを探索するためのデフォルト・オーバーフロー・ログ・パス。

logpaths
回復中にアーカイブ・ログを探索するための代替ログ・パスのリストを含む複合 REXX ホスト変数。以下の項目において、XXX はホスト変数の名前を表しています。

XXX.0 変更されるオーバーフロー・ログ・パスの数

XXX.1.1 最初のノード

XXX.1.2 最初のノードのオーバーフロー・ログ・パス

XXX.2.1 2 番目のノード

XXX.2.2 2 番目のノードのオーバーフロー・ログ・パス

XXX.3.1 3 番目のノード

XXX.3.2 3 番目のノードのオーバーフロー・ログ・パス

XXX.x.1 以下同様に続く。

nodelist

ノードのリストを含む REXX ホスト変数。以下の項目において、XXX はホスト変数の名前を表しています。

XXX.0 ノードの数

XXX.1 最初のノード

XXX.2 2 番目のノード

XXX.x 以下同様に続く。

データ構造: RFWD-INPUT

この構造は、 156ページの『Rollforward Database API』 に情報を渡すのに使用します。

表 8. RFWD-INPUT 構造のフィールド

フィールド名	データ・タイプ	説明
VERSION	sqluint32	ロールフォワード・バージョン。
PDBALIAS	ポインター	データベース別名。
CALLERACTION	UNSIGNED SHORT	アクション。
PSTOPTIME	ポインター	停止時刻。
PUSERNAME	ポインター	ユーザー名。
PPASSWORD	ポインター	パスワード。
POVERFLOWLOGPATH	ポインター	オーバーフロー・ログ・パス。
NUMCHNGLGOVRFLW	UNSIGNED SHORT	変更されたオーバーフロー・ログ・パスの数 (MPP 専用)。
PCHNGLOGOVRFLW	構造体	変更されたオーバーフロー・ログ・パス (MPP 専用)。
CONNECTMODE	UNSIGNED SHORT	接続モード。
PTABLESPACELIST	構造体	表スペース名のリストを指すポインター。この構造の詳細については、106ページの『データ構造: SQLU-TABLESPACE-BKRST-LIST』を参照してください。
ALLNODEFLAG	SHORT	すべてのノード・フラグ。
NUMNODES	SHORT	ノード・リストのサイズ。
PNODELIST	ポインター	ノード番号のリスト。
NUMNODEINFO	SHORT	168ページの『データ構造: RFWD-OUTPUT』 の *pNodeInfo* のサイズ。
DLMODE	UNSIGNED SHORT	このパラメーターは現在は使用されていません。
PREPORTFILE	ポインター	このパラメーターは現在は使用されていません。
PDROPPEDTBLID	ポインター	回復が実行されている消去済み表の ID を含むストリング。
PEXPORTDIR	ポインター	削除した表データのエクスポート先のディレクトリー。
NODENUM	SQL_PDB_NODE_TYPE	ノード番号。
PATHLEN	UNSIGNED SHORT	新規のログ・パスの長さ。
LOGPATH	CHAR(255)	新規のオーバーフロー・ログ・パス。

言語構文

C 構造

```
/* File: sqlutil.h */
/* Structure: RFWD-INPUT */
/* ... */
SQL_STRUCTURE rfwd_input
{
  sqluint32              version;
  char                   *pDbAlias;
  unsigned short         CallerAction;
  char                   *pStopTime;
  char                   *pUserName;
  char                   *pPassword;
  char                   *pOverflowLogPath;
  unsigned short         NumChngLgOvrflw;
  struct sqlurf_newlogpath *pChngLogOvrflw;
  unsigned short         ConnectMode;
  struct sqlu_tablespace_bkrst_list *pTablespaceList;
  short                  AllNodeFlag;
  short                  NumNodes;
  SQL_PDB_NODE_TYPE      *pNodeList;
  short                  NumNodeInfo;
  unsigned short         DlMode;         /* This parameter is not */
                                         /*   currently used. */
  char                   *pReportFile;   /* This parameter is not */
                                         /*   currently used. */
  char                   *pDroppedTblID;
  char                   *pExportDir;
};
/* ... */


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

COBOL 構造

```
* File: sqlutil.cbl
01 SQL-RFWD-INPUT.
    05 SQL-VERSION            PIC 9(9) COMP-5.
    05 SQL-DBALIAS            USAGE IS POINTER.
    05 SQL-CALLERACTION       PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-STOPTIME           USAGE IS POINTER.
    05 SQL-USERNAME           USAGE IS POINTER.
    05 SQL-PASSWORD           USAGE IS POINTER.
    05 SQL-OVERFLOWLOGPATH    USAGE IS POINTER.
    05 SQL-NUMCHANGE          PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-CHNG-LOG-OVRFLW  USAGE IS POINTER.
    05 SQL-CONNECTMODE        PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-TABLESPACE-LIST  USAGE IS POINTER.
    05 SQL-ALLNODEFLAG        PIC S9(4) COMP-5.
    05 SQL-NUMNODES           PIC S9(4) COMP-5.
    05 SQL-NODELIST           USAGE IS POINTER.
    05 SQL-NUMNODEINFO        PIC S9(4) COMP-5.
    05 SQL-DLMODE             PIC 9(4) COMP-5.  * This parameter is not
                                                *   currently used.
    05 SQL-REPORTFILE         USAGE IS POINTER. * This parameter is not
                                                *   currently used.
    05 SQL-DROPPEDTBLID       USAGE IS POINTER.
    05 SQL-EXPORTDIR       USAGE IS POINTER.
*


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

データ構造: RFWD-OUTPUT

この構造は、156ページの『Rollforward Database API』 から情報を渡すのに使用されます。

表 9. RFWD-OUTPUT 構造のフィールド

フィールド名	データ・タイプ	説明
PAPPLICATIONID	ポインター	API から戻されたアプリケーション ID を保持する、長さ SQLU_APPLID_LEN+1 (sqlutil で定義される) のバッファーのアドレス。この ID を指定してデータベース・システム・モニター API を呼び出せば、アプリケーションをモニターできます。この情報が必要ない場合は、ヌル・ポインターを指定してください。区分データベース環境では、カタログ・ノードのアプリケーション ID だけを戻します。
PNUMREPLIES	ポインター	受信されたノード応答の数。応答するそれぞれのノードが、 *pNodeInfo* の *sqlurf_info* 構造に入ります。単一区画環境では、このパラメーターの値は 1 でなければなりません。
PNODEINFO	構造体	ノード応答情報。 *NumNodeInfo* 個の *sqlurf_info* 構造のユーザー定義の配列。

表 10. SQLURF-INFO 構造のフィールド

フィールド名	データ・タイプ	説明
NODENUM	SQL_PDB_NODE_TYPE	ノード番号。
STATE	LONG	状態情報。
NEXTARCLOG	UNSIGNED CHAR(13)	必要とされる次のアーカイブ・ログ・ファイルの戻された名前を保持する 12 バイトのバッファー。 SQLUM_QUERY 以外の呼び出し側アクションを指定した場合には、このフィールドに戻される値によって、ファイルへのアクセス時にエラーが生じたことが示されます。考えられる原因は、以下のとおりです。 • データベース・ログ・ディレクトリー内、またはオーバーフロー・ログ・パス・パラメーターで指定されたパス上にファイルが見つからない。 • ユーザー出口プログラムがアーカイブ・ファイルを戻すのに失敗した。

表 10. SQLURF-INFO 構造のフィールド (続き)

フィールド名	データ・タイプ	説明
FIRSTARCDEL	UNSIGNED CHAR(13)	回復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持する 12 バイトのバッファー。このファイルと、*lastarcdel* までのすべてのファイルを移動させて、ディスク上のスペースを空けることができます。 たとえば、 *firstarcdel* と *lastarcdel* で戻された値が S0000001.LOG と S0000005.LOG である場合には、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	UNSIGNED CHAR(13)	データベース・ログ・ディレクトリーから除去できる最後のアーカイブ・ログ・ファイルの戻された名前を保持する 12 バイトのバッファー。
LASTCOMMIT	UNSIGNED CHAR(27)	ISO 形式のタイム・スタンプを含む 26 文字のストリング。この値は、ロールフォワード操作の終了後、最後にコミットされたトランザクションのタイム・スタンプを示します。

STATE に有効な値 (sqlutil で定義) は、以下のとおりです。

SQLURFQ_NOT_AVAILABLE
ノードに接続できなかった。

SQLURFQ_NOT_RFW_PENDING
データベースがロールフォワード保留状態ではない。

SQLURFQ_DB_RFW_PENDING
データベースがロールフォワード保留状態である。

SQLURFQ_TBL_RFW_PENDING
表スペースがロールフォワード保留状態である。

SQLURFQ_DB_RFW_IN_PROGRESS
データベースがロールフォワード進行状態である。

SQLURFQ_TBL_RFW_IN_PROGRESS
表スペースがロールフォワード進行状態である。

SQLURFQ_DB_RFW_STOPPING
STOP 要求の処理中にデータベースのロールフォワード操作が中断された。

SQLURFQ_TBL_RFW_STOPPING
STOP 要求の処理中に表スペースのロールフォワード操作が中断された。

言語構文

C 構造

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


/* File: sqlutil.h */
/* Structure: SQLURF-INFO */
/* ... */
SQL_STRUCTURE sqlurf_info
{
  SQL_PDB_NODE_TYPE nodenum;
  long            state;
  unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
};
/* ... */
```

COBOL 構造

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

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

ロールフォワード・セッションの例

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
```

この 2 つは同等ですが、このような操作は 2 つのステップで行うことをお勧めします。ロールフォワード操作を停止してログが欠落している可能性が生じる前に、操作が予期したとおり進行したことを検証することが重要です。このことは、ロールフォワード回復中に不良ログが検出され、不良ログが 『ログの終わり』 を示していると解釈されてしまう場合に、特に重要です。そのような場合、そのログの損傷していないバックアップ・コピーを使用して、さらに続くログへロールフォワード操作を継続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 つの表スペースが復元されています)。

```
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、ロールフォワード回復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 つの表スペースが復元された後、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

データベースを復元した後、 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 (MPP)

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 でデータベースを復元し、ノード 0 および 2 で TBS1 を復元した後、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 (『データベースは回復されましたが、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) は任意指定です。

例 6 (MPP)

ノード 0 および 2 でのみ表スペース TBS1 を復元した後、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時刻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 を復元した後、

```
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 (MPP)

すべてのノードで表スペースを回復した後、 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 (MPP)

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回復します。

```
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 (MPP)

(ノード 6 上の) 1 つのノード・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回復します。

```
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。

DB2 コマンド・スクリプトの例と、その使用法に関する情報は、443ページの『付録F. 回復 CLP スクリプト』を参照してください。

API の例

DB2 API と組み込み SQL 呼び出しを含むサンプル・プログラムと、その使用法に関する情報は、377ページの『付録E. 回復サンプル・プログラム』を参照してください。

ロールフォワードの制約事項

ロールフォワード・ユーティリティーには、以下の制約事項が適用されます。

- 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。回復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時刻まで*継続する* ことはできません。特定の時刻が指定されない場合は、直前の時刻が使用されます。 STOP を指定するだけで、特定の時刻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・イメージから復元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 (ロールフォワード保留状態) 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時刻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。

ロールフォワードのトラブルシューティング

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復元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保留状態にある、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。

ロールフォワード・ユーティリティーに回復不能操作 (たとえば、ロードにコピーがない) が生じた場合、関連した表スペースは復元保留状態になります。表スペースを復元保留状態でなくすには、より最近のデータベース・レベルまたは表スペース・レベルのバックアップ・イメージから復元する必要があります。

ロールフォワード保留状態または復元保留状態の表スペースがあることの警告として、(AND STOP がまだ要求されていない場合でも) SQL1271 が戻されます。これは、データベースのロールフォワード操作中の場合、表スペースの 1 つがロールフォワード・ユーティリティーによってオフラインになったことを示します。表スペース のロールフォワード操作の場合は、ロールフォワード中の表スペースの 1 つがロールフォワード・ユーティリティーによってオフラインになったか、あるいは、ロールフォワードを依然必要とする別の (リストされていない) 表スペースがあることを示します。

ロールフォワードされていた (リストで指定されていた、またはロールフォワード保留状態にあったため) すべての表スペースがロールフォワード・ユーティリティーによってオフラインになった場合、SQL1272 が戻されます。これは、回復不能ロード操作が生じた表、または NOT LOGGED INITIALLY 処理を指定した表が含まれている可能性があることを示します。このエラーが戻された場合、表スペースのロールフォワード操作は停止されているはずです (つまり、もはや進行状態にない)。

第2部 高可用性

第5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 x 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。

高可用性

高可用性 (HA) とは、常時稼働していて、カスタマー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。区分データベース環境では、DB2 は区画内および区画間の並列性の、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。*区画内並列性* は、SMP 環境で使用することができ、これにより複合 SQL ステートメントのさまざまなコンポーネントを同時に処理することができます。一方、区分データベース環境での *区画間並列性* 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。並列性についての詳細は、*管理の手引き* を参照してください。
- 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回復できなければなりません。これにはジャーナル・ファイル・システムが役立ちます。ジャーナル・ファイル・システム とは、自動的に変更をその構造に記録するファイル・システムです。これにより、ストレージ・メディアが破損または破壊されていなければ、システム破損が起きた場合でもファイル・システムの構造へのすべての変更は安全です。システム破損の後、当初ログに記録された順序でこれらのログ内の変更がファイル・システムに適用されます。ジャーナル・ファイル・システムの特徴は、回復時間が速いことです。これは大規模なファイル・システムを持つ環境や、データ保全性が特に重要な場合に適しています。

  即時に回復できるかどうかは、証明済みのバックアップおよび回復ストラテジーが配備されていることに大いに依存しています。回復ストラテジーについての詳細は、3 ページの『第1章 適切なバックアップと回復の方針の開発』 を参照してください。
- 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ワークロードを自動的に移すことが可能になります。

フェイルオーバー保護は、ログ・ファイルを絶えずロールフォワードしている別のマシンに、データベースのコピーを保持しておくことで達成されます。ログ・シップ とは、ログ・ファイル全体を待機マシンにコピーするプロセスのことです。これはアーカイブ装置から、または 1 次データベースに対するユーザー出口プログラムを使用して行います。この方法では、1 次データベースは、 DB2 復元ユーティリティーまたは分割ミラー機能を使用して待機マシンに復元されます。新しいサスペンド入出力サポートを使用して、新規データベースを即時に初期化することができます (182ページの『オンライン分割ミラーおよびサスペンド入出力サポートを使用した高可用性』 を参照してください)。待機マシン上の 2 次データベースはログ・ファイルをロールフォワードします。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・ファイルはすべて待機マシンにコピーして移されます。ログの最後までのロールフォワードおよび停止操作の後、すべてのクライアントは待機マシン上の 2 次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、次のようなものがあります。

- AIX 用高可用性クラスター・マルチプロセッシング、拡張スケーラビリティー (HACMP ES)。

  HACMP/ES についての情報は、185ページの『第6章 AIX での高可用性』 、または『IBM DB2 Universal Database Enterprise Edition for AIX and HACMP/ES』 というタイトルのホワイトペーパー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 NT または Windows 2000 版)。

  MSCS についての詳細情報は、227ページの『第7章 Windows オペレーティング・システムでの高可用性』 を参照してください。

- Sun Cluster、または VERITAS Cluster Server (Solaris 実行環境版)。

  Sun Cluster 2.x についての情報は、261ページの『第8章 Solaris 実行環境での高可用性』 を参照してください。 Sun Cluster 3.0 についての情報は、 『DB2 and High Availability on Sun Cluster 3.0』 というタイトルのホワイトペーパー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 についての情報は、 『DB2 and High Availability on VERITAS Cluster Server』 というタイトルのホワイトペーパーを参照してください。これも 『DB2 UDB and DB2 Connect Online Support』 Web サイトから入手できます。

- Multi-Computer/ServiceGuard (Hewlett-Packard 用)。

HP MC/ServiceGuard についての詳細情報は、 『IBM DB2 EE v.7.1 Implementation and Certification With Hewlett-Packard's MC/ServiceGuard High Availability Software』というタイトルのホワイトペーパー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。

フェイルオーバー・ストラテジーは通常、システムのクラスターに基づいています。クラスター は、単一のシステムとして協働する、接続されたシステムのグループです。各プロセッサーはクラスター内でノードと呼ばれます。クラスター化により、障害が発生したときには、障害の発生したサーバーのワークロードを収集する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 アドレス引き受け (または IP 引き受け) とは、あるサーバーが使用不能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、クライアント・アプリケーション側には、その 2 つのマシンは同一のサーバーが別々の時に現れていると認識されます。

フェイルオーバー・ソフトウェアは、ハートビート・モニター 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、可用性を確保することもあります。ハートビート・モニターには、クラスター内のすべてのノード間での常時通信を維持するシステム・サービスが関係します。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・スタンドバイ* および*相互引き受け* 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・スタンドバイ
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』、つまりスタンドバイ・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。スタンドバイ・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。

相互引き受け
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余分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、区画、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。

オンライン分割ミラーおよびサスペンド入出力サポートを使用した高可用性

サスペンド入出力 は、オンライン分割ミラー処理 (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、連続システム使用可能性をサポートしています。*分割ミラー* はデータを含むディスクをミラーリングし、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、ミラーのバックアップ・コピーを作成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分割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。

- 実動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。
- システムを複製する高速の方法です。
- アイドル・スタンドバイ・フェイルオーバーの高速のインプリメンテーションです。初期の復元操作は行われません。ロールフォワード操作が遅すぎたりエラーが発生する場合に、再初期設定が非常に高速です。

db2inidb コマンドは、分割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。

- 複製データベースを作成する

  1 次データベースの、読み取り専用の複製を使用すると、たとえばレポートを作成することができます。
- スタンドバイ・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、分割されたミラーに対してのみ発行することができます。分割されたミラーはまず最初に **db2inidb** を実行しなければなりません。それから使用することができるようになります (326ページの『db2inidb - ミラーリングされたデータベースの初期化』 を参照してください)。

区分データベース環境では、**db2inidb** コマンドは各区画上でそれぞれ実行する必要があります。それから、すべての区画の分割イメージを使用することができます。このツールはすべての区画で同時に実行することができます。

複製データベースの作成

複製したデータベースは、 1 次 (活動状態の) データベースのオフライン 『バックアップ』 にすることができます。ただし、複製したデータベースをバックアップしたり、そのイメージを元のシステムに復元したり、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。

データベースを複製するには、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. 該当するオペレーティング・システムの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. ミラーリングされたデータベースに別のマシンから接続します。
5. データベース・インスタンスを開始します。

   ```
   db2start
   ```

6. ミラーリングされたデータベースを 1 次データベースの複製として初期設定します。

   ```
   db2inidb database_alias as snapshot
   ```

   注: 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ます。

分割ミラーをスタンドバイ・データベースとして使用する

ミラーリングされた (スタンドバイ) データベースはログを継続してロールフォワードし続けるため、 1 次データベースが作成する新規ログは常に 1 次システムから取り出されます。分割ミラーをスタンドバイ・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. 該当するオペレーティング・システムの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. ミラーリングされたデータベースを別のインスタンスに接続します。
5. ミラーリングされたデータベースをロールフォワード保留状態にします。

   ```
   db2inidb database_alias as standby
   ```

   DMS 表スペース (データベース管理スペース) がある場合は、完全データベース・バックアップを取ることで、実動データベース上でバックアップを取る場合のオーバーヘッドを軽減できます。

6. ユーザー出口プログラムをセットアップし、1 次システムから最新のログ・ファイルを取得します。
7. データベースをログの最後までロールフォワードします。

8. 1 次データベースがダウンするまで、ログ・ファイルの取得を続け、データベースをログの最後までロールフォワードします。

分割ミラーをバックアップ・イメージとして使用する

分割ミラーを 『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. 該当するオペレーティング・システムの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. オペレーティング・システム・レベルのコマンドを使用して、ミラーリングされたデータおよび 1 次システムのログをコピーします。
5. データベース・インスタンスを開始します。

   ```
   db2start
   ```

6. ミラーリングされたデータベースを、『バックアップ・イメージ』 として初期設定します。これは、分割されたディスク上のデータを元のシステム上のディスクにコピーして戻すのに使用できます。 (ログ・ファイルを含むファイル・システムは戻さないでください。ロールフォワード処理中にログが必要になるからです。)

   ```
   db2inidb database_alias as mirror
   ```

7. 元のシステム上のデータベースをログの最後までロールフォワードします。

第6章 AIX での高可用性

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、フェイルオーバー回復を提供し、 HACMP と同じイベント構造を持ちます (*HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide* を参照してください)。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、フェイルオーバー回復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、異なる実装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。

  ユーザー定義イベントおよび規則ファイルについて詳しくは、 205ページの『HACMP ES イベント・モニターおよびユーザー定義イベント』を参照してください。
- 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回復を呼び出します。回復処理は *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。回復処理が完了すると、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。回復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) スタンドバイ」と「相互引き受け」を指定できます。

クラスター構成

ホット・スタンドバイ 構成では、引き受けノードではない AIX プロセッサー・ノードは他の作業負荷を一切実行していません。*相互引き受け* 構成では、引き受けノードである AIX プロセッサー・ノードは他の作業負荷を実行しています。

一般に、DB2 ユニバーサル・データベース エンタープライズ拡張エディション (UDB EEE) は、各ノード上の区画で相互引き受け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・スタンド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、残りのフレームを回復しなければならない場合があります。この回復手順については、SP 管理の手引きで説明されています。 HACMP ES では、SP ノード障害を回復できます。その場合、フレーム障害の回復は、 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 フェイルオーバー回復では、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回復手順では、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・グループに配置されるノード数によって指定されます。

たとえば、DB2 データベース区画 (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) データベース区画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・システムを回復する場合にも使用できます。

DB2 UDB EEE の回復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、環境に合った回復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、回復を要する各サブシステムについては、必要な HACMP クラスターと回復ノード (ホット・スタンドバイまたは相互引き受け) を識別してください。これは、資料にある HACMP ワークシートを完成させる上で開始点となります。

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用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、いくらか余分の計画を立てることが必要です。 DB2 UDB EEE 内では、すべてのデータベース間でコンテナー名をすべて固有のものにしておく必要があります。

固有のものにする 1 つの方法は、名前の一部に区分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 11. 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
```

図18 および 190ページの図19 は、 DB2 SSA I/O サブシステム構成の例を示しています。また、高可用性外部ディスク構成、および競合を起こさずにすべてのボリューム・グループにアクセスする機能の両方を実現するために必要な計画の一部を示しています。

DB2 SSA I/O サブシステム構成 – 障害の単一ポイントはない

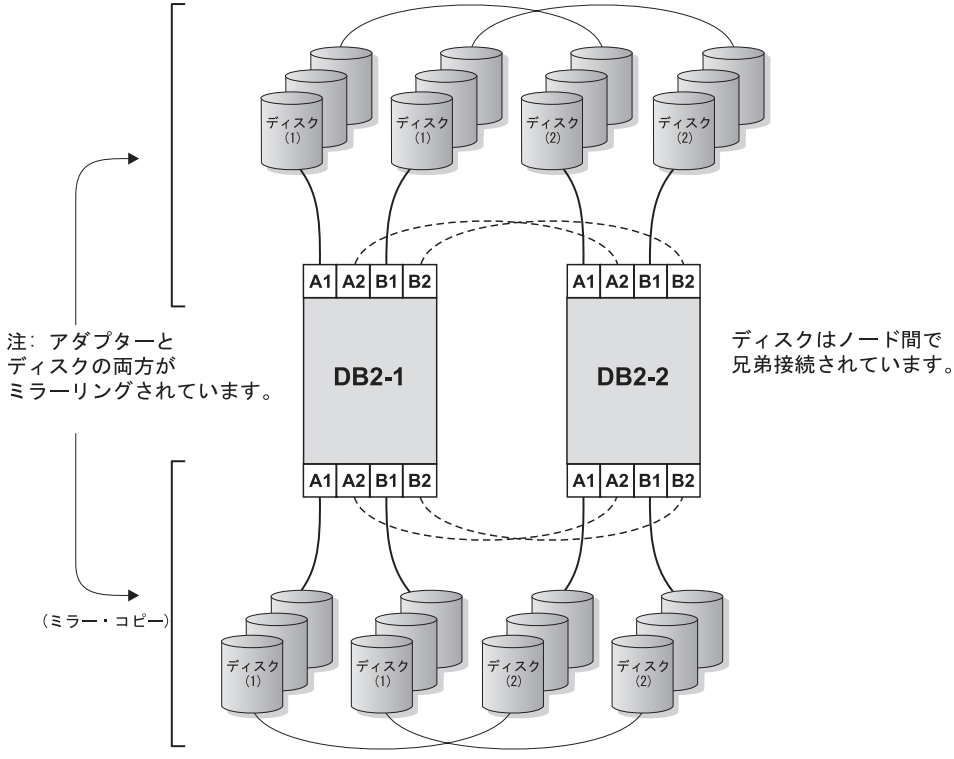


図 18. 単一の障害点がない

DB2 SSA I/O サブシステム構成 –
ボリューム・グループと論理ボリュームのセットアップ

db2 database testdata on filesystem /database instance name powertp

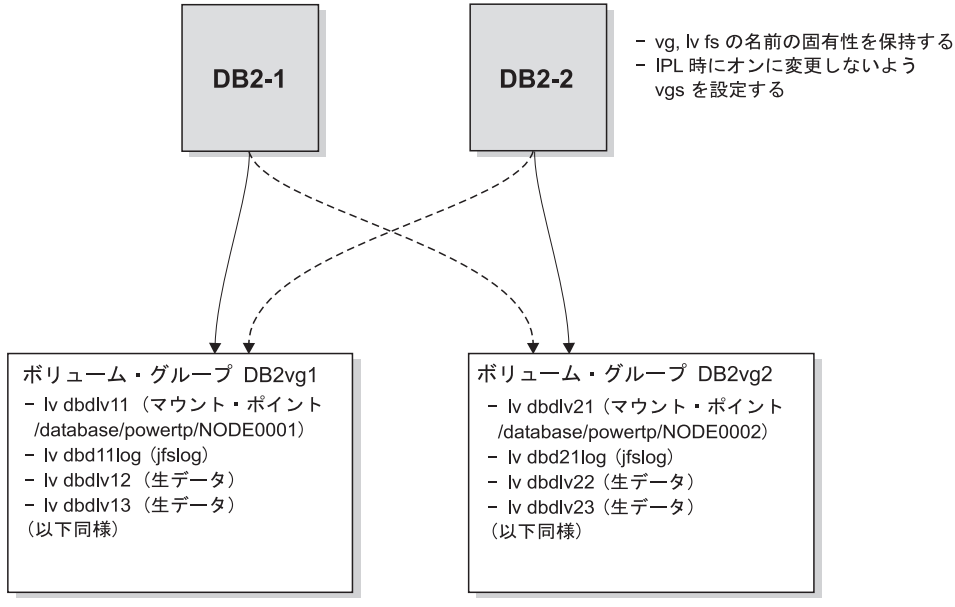


図 19. ボリューム・グループおよび論理ボリュームのセットアップ

DB2 データベース区画の構成

構成が済むと、インスタンス中の各データベース区画は 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・スタンド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回復を構成するための補助機構として、 DB2 UDB EEE 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 つのデータベース区画を稼働する場合に適切なパフォーマンスを確保できるように構成する必要があります。)

DB2 データベース区画の HACMP 構成でアプリケーション・サーバーを作成する場合、次のようにして rc.db2pe を開始および停止スクリプトとして指定します。

```
/usr/bin/rc.db2pe <instance> <dpn> <secondary dpn> start <use switch>
/usr/bin/rc.db2pe <instance> <dpn> <secondary dpn> stop <use switch>
```

ここで、各パラメーターは以下のとおりです。

```
<instance> はインスタンス名。
<dpn> はデータベース区画番号。
<secondary dpn> は、相互引き受け構成のみにおける
   'コンパニオン' データベース区画番号。
   ホット・スタンバイ構成では、<dpn> と同じ。
<use switch> は通常はブランク。ブランクの場合は、
   SP switch ネットワークが db2nodes.cfg ファイルのホスト名 フィールドに使用
   される。(DB2 のためのすべてのトラフィックが SP スイッチを介して経路指定
   される。)ブランクでない場合、ここで使用する名前は、使用される SP ノード
   のホスト名。
```

このデータベース区画用に構成したすべてのデータベースを検索するには、 rc.db2pe 内から DB2 コマンド LIST DATABASE DIRECTORY を使用します。すると、スクリプト・ファイルは、 /usr/bin/reg.parms.*DATABASE* ファイル、および /usr/bin/failover.parms.*DATABASE* ファイルを探します。ただし、*DATABASE* はこのデータベース区画用に構成された個々のデータベースを表します。相互引き受け構成では、パラメーター・ファイル reg.parms.*xxx* および failover.parms.*xxx* を作成することをお勧めします。 failover.parms.*xxx* ファイルでは、複数のバッファー・プールがある可能性を考慮して、 BUFFPAGE、DBHEAP、およびバッファー・プールに影響を与える他のパラメーターの設定を調整します。サンプル・ファイル reg.parms.SAMPLE および failover.parms.SAMPLE は、ユーザーが使用するために用意されました。

この環境で重要なパラメーターの 1 つは、 *start_stop_time* データベース・マネージャー構成パラメーターです。このパラメーターのデフォルト値は 10 分です。ただし、rc.db2pe を指定すると、このパラメーターは 2 分に設定されます。このパラメーターは rc.db2pe で調整して、 10 分か、もう少し大きめの値に設定するとよいでしょう。このコンテキストでは、指定した時間は、区画の障害発生からその区画の回復までの時間間隔です。区画上で実行されているアプリケーションが頻繁に COMMIT を出す場合、データベース区画上の障害後、10 分もあれば、コミットされていないトランザクションをロールバックし、その区画上のデータベースの一貫性ポイントに到達することが可能です。作業負荷が重いか、または多くの区画がある場合、ロールバック操作が完了する前にタイムアウトになってしまう可能性を少なくするために、時間を増やす必要があるかもしれません。

以下は、ホット・スタンドバイ構成および相互引き受け構成の例です。どちらの例でも、リソース・グループにはサービス IP スイッチ別名アドレスが入っています。このスイッチ別名アドレスは、以下の目的で使用されます。

- DB2 インスタンス所有者ファイル・システムを使用するために行うファイル・サーバーへの NFS アクセス。
- フェイルオーバー、TSM (Tivoli Storage Manager、以前の ADSM) 接続、または他の同様の操作で管理が必要となる他のクライアント・アクセス。

実際の設定でこれらの別名が必要でなければ、削除しても構いません。削除した場合は、 rc.db2pe スクリプト・ファイルで *MOUNT_NFS* パラメーターを NO に設定してください。

ホット・スタンドバイ構成の例

この例では、ホット・スタンバイ構成がノード 1 とノード 2 の間にあり、 DB2 インスタンス名が POWERTP であると仮定します。データベース区画は 1 で、データベースはファイル・システム /database 上の TESTDATA です。

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1     (<<< this is the switch alias address)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 1 stop
```

相互引き受け構成の例

この例では、相互引き受け構成がノード 1 とノード 2 の間にあり、 DB2 インスタンス名が POWERTP であると仮定します。データベース区画は 1 および 2 で、データベースはファイル・システム /database 上の TESTDATA です。

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1     (<<< this is the switch alias address)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 2 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 2 stop

Resource group name: db2_pd_2
Node Relationship: cascading
Participating nodenames: node2_eth, node1_eth
Service_IP_label: nfs_switch_2     (<<< this is the switch alias address)
Filesystems: /database/powertp/NODE0002
Volume Groups: DB2vg2
Application Servers: db2_dp2_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 2 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 2 1 stop
```

NFS サーバー・ノードの構成

また、rc.db2pe スクリプトを使用して、 DB2 並列インスタンス・ユーザー・ディレクトリーの NFS マウント・ディレクトリーが使用可能になります。これは、 rc.db2pe スクリプト・ファイルで *MOUNT_NFS* パラメーターを YES に設定し、 NFS フェイルオーバー・サーバーのペアを以下のように構成することによって実行できます。

- ホーム・ディレクトリーを構成し、「ルート」として、 /etc/exports および **exportfs** コマンドを使用して、このディレクトリーをエクスポートします。エクスポート先は、 NFS サーバーの IP アドレスと同じサブネット内のノードで使用する IP アドレスとします。 HACMP ブートおよびサービス・アドレスの両方を指定します。 NFS サーバーの IP アドレスは、バックアップで引き受け可能な HACMP 内のサービス・アドレスと同じアドレスです。 DB2 インスタンス所有者のホーム・ディレクトリーには、自動マウントではなく、 NFS マウント・ディレクトリーを指定してください。 (スクリプトは、DB2 インスタンス所有者ホーム・ディレクトリーとして、自動マウント・プログラムの使用をサポートしていません。)
- このファイル・システムでは、SMIT または実用的な構成を使用して別個の /etc/filesystems 項目を作成し、 DB2 並列グループ化ですべてのノード (ファイル・サーバーを含む) が NFS ファイル・システム・コマンドでマウントできるようにします。

  たとえば、/nfshome JFS ファイル・システムは、 /dbhome としてすべてのノードにエクスポートできます。各ノードは、 nfs_server:/nfshome である NFS ファイル・システム /dbname を作成します。したがって、インスタンス名が「powertp」の場合、 DB2 インスタンス所有者のホーム・ディレクトリーは /dbhome/powertp となります。

  /etc/filesystems のマウント用 NFS パラメーターが、「hard」、「bg」、「intr」、「rw」であることを確かめてください。
- /etc/passwd にあるホーム・ディレクトリー /dbhome/powertp と関連付けられた DB2 インスタンス所有者定義が、すべてのノードで同じものであることを確認します。

  SP 環境内のユーザー定義は普通、コントロール・ワークステーションで作成され、「supper」または「pcp」は、/etc/passwd、 /etc/security/passwd、/etc/security/user、および /etc/security/group をすべてのノードに配布するために使用します。
- HACMP リソース・グループで、ボリューム・グループおよびエクスポートされるファイル・システムに対して「エクスポートする nfs_filesystems」を構成しないでください。代わりに、NFS に対して通常どおりに構成します。 NFS サーバーのスクリプトは、ファイル・システムのエクスポートを制御します。
- ファイル・システムがあるボリューム・グループの主要な番号が、 1 次ノードでも引き受けノードでも同じ番号であることを確かめてください。この確認は、 **importvg** に -V オプションを指定して行います。

- rc.db2pe で *MOUNT_NFS* パラメーターが YES に設定され、 /etc/filesystems にマウントする NFS ファイル・システムが各ノードにあることを検査します。もしなければ、 rc.db2pe がファイル・システムをマウントして DB2 を始動することはできません。
- DB2 インスタンス所有者がすでに作成されており、作成するファイル・システムにユーザーのディレクトリー構造をコピーするのであれば、必ずディレクトリーを **tar** (-cvf) してください。これにより、シンボリック・リンクが確実に保存されます。
- 作成するファイル・システムの論理ボリュームとファイル・システム・ログについて、アダプターとディスクの両方を必ずミラーリングしてください。

NFS サーバー引き受け構成の例

この例では、 IP アドレス「nfs_server」を介してボリューム・グループ nfsvg に NFS サーバー・ファイル・システム /nfshome があることを想定しています。 DB2 インスタンス名は POWERTP で、ホーム・ディレクトリーは /dbhome/powertp です。

```
Resource group name: nfs_server
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_server     (<<< this is the switch alias address)
Filesystems: /nfshome
Volume Groups: nfsvg
Application Servers: nfs_server_app
Application Server Start Script: /usr/bin/rc.db2pe powertp NFS SERVER start
Application Server Stop Script: /usr/bin/rc.db2pe powertp NFS SERVER stop
```

この例では、以下のようになります。

- 全ノード上の /etc/filesystems には、マウントする nfs_server:/nfshome として /dbhome の項目が入っています。 nfs_server は サービス IP スイッチ別名アドレスです。
- nfs_server ノードおよびバックアップ・ノード上の /etc/exports には、ブートおよびサービス・アドレスが入っており、 /nfsfs -root=nfs_switch_1、nfs_switch_2、 ... の項目が含まれています。

SP スイッチ構成時の考慮事項

HACMP ES に SP スイッチを実装する場合は、以下の事柄を考慮します。

- SP スイッチには、「基底」アドレスおよび「別名」アドレスがあります。基底アドレスは SP システム・データ・リポジトリー (SDR) で定義されたアドレスであり、システムが「ブートされる」ときに rc.switch で定義されます。別名アドレスは、別名属性を指定した ifconfig コマンドにより、基底アドレスに加えて **css0** インターフェースへ構成された IP アドレスです。たとえば、次のようなものがあります。

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

- DB2 db2nodes.cfg ファイルを構成するときは、「hostname」フィールドでも「netname」フィールドでも、 SP スイッチ「基底」IP アドレスを使用しなければな

りません。スイッチ IP アドレスの別名は、 NFS 接続性を維持することだけ を目的として使用します。 DB2 フェイルオーバーは、 **db2start** (RESTART) コマンド (db2nodes.cfg を更新する) を使用して DB2 を再始動することにより実行します。

- スイッチ・アドレスと etc/hosts 別名を混同しないでください。 SP スイッチ・アドレスも SP スイッチ別名アドレスも、 etc/hosts または DNS のいずれかに実在します。スイッチ別名アドレスは、SP スイッチ基底アドレスの別名ではありません。どちらのアドレスにも、固有の別個のアドレスがあります。
- 活動時のノードには常に、SP スイッチ基底アドレスがあります。 HACMP ES は、ノード間でこれらのアドレスを構成したり移動したりしません。
- SP スイッチ別名アドレスを使用する場合は、「ハートビート」用のブート・アドレスとサービス・アドレスおよび IP アドレス引き受けとして、 HACMP に対して別名アドレスを構成します。 SP スイッチ別名アドレスを使用しない場合は、「ハートビート」*専用* のサービス・アドレス (IP アドレス引き受けではない) として、HACMP に対して基底アドレスを構成します。どのような構成を行う場合でも、別名アドレスとスイッチ基底アドレスの*両方* は構成しないでください。そのような構成は、HACMP ES ではサポートされません。
- SP スイッチ基底アドレスではなく、 SP スイッチ別名アドレスだけが、 IP 引き受け構成のノード間を移動されます。
- ノードあたりの SP スイッチ・アダプターが 1 つだけの場合もあるので、 SP スイッチ別名の必要性が生じます。別名アドレスを使用すると、ノードは、別のスイッチ・アダプターを追加しなくても、別のノードのスイッチ別名 IP アドレス を引き受けることができます。この方法は、「スロットに制約がある」ノードで役立ちます。SP スイッチ・アダプターの障害からの回復方法の詳細については、 208ページの『HACMP ES スクリプト・ファイル』のネットワーク障害のセクションを参照してください。
- IP アドレス引き受け用の SPスイッチを構成する場合、ノードごとに 2 つの余分の IP アドレスが必要になります。 1 つはブート・アドレス、もう 1 つはサービス・アドレスになります。
- SP スイッチ IP アドレスまたは SP スイッチ別名 IP アドレスの HACMP ES ネットワーク名定義では、必ず「HPS」を使用してください。
- HACMP の rc.cluster は、 HACMP の始動時に SP スイッチ・ブート・アドレスで自動的に **ifconfig** を実行します。 IP アドレスと名前を作成し、それらを HACMP に定義する以外に、余分の構成は必要とされません。
- SP スイッチの Eprimary ノードは、 Estart、Efence、および Eunfence コマンドを実装するサーバーです。スイッチがネットワークの 1 つとして定義されると、HACMP スクリプトは、HACMP の始動時にノードに対して Eunfence または Estart を実行し、スイッチを使用可能にしようとします。それで、HACMP を始動するときは Eprimary が使用可能であることを確認してください。 HACMP コードは、Eprimary フェイルオーバーがエラー終了する前に完了できるよう、最大で 12 分間待機します。

- ノード間では、HACMP ではなく、 SP AIX 並列システム・サポート・プログラム (PSSP) が SP スイッチの Eprimary ノードを移動します。 Eprimary ノードがオフラインになると、 PSSP は自動的にバックアップ・ノードで Eprimary ノードとしての責任を果たします。スイッチ・ネットワークはこの変更による影響を受けず、活動状態を保ちます。

DB2 HACMP 構成の例

次の例では、異なるフェイルオーバー・サポート構成と、障害発生時に生じる事柄を示します。

DB2 HACMP 相互引き受け構成 (197ページの図20、198ページの図21、および 199ページの図22) については次のとおりです。

- HACMP アダプターはイーサネット、および SP スイッチ別名のブートおよびサービス別名用に定義され、基底アドレスは未定義のままです。 HACMP ネットワーク名では必ず、「HPS」ストリングを使用してください。
- `NFS_server/nfshome` は、スイッチ別名を使用して全ノードで `/dbhome` としてマウントされます。
- `db2nodes.cfg` ファイルには、 SP スイッチ基底アドレスが入っています。 DB2 データベース区画 (論理ノード) のフェイルオーバー後、 `db2nodes.cfg` ファイルは **DB2START** (RESTART) コマンドで変更されます。
- SP スイッチ別名ブート・アドレスは表示されません。
- ノードは、異なる SP フレームに置くことができます。

NFS フェイルオーバーを使用した DB2 HACMP 相互引き受け – 正常

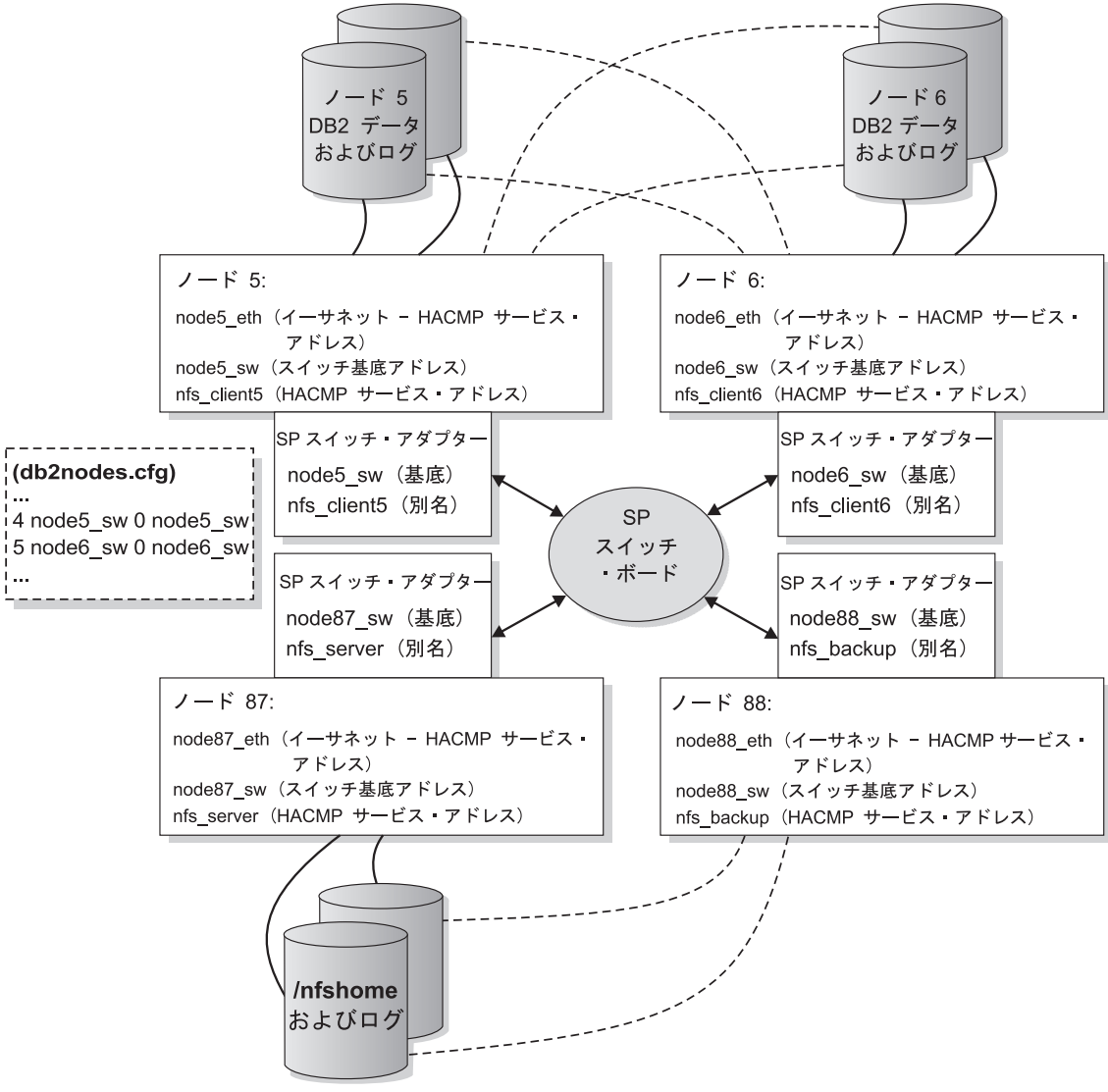


図 20. NFS フェイルオーバーを使用した相互引き受け - 正常

NFS フェイルオーバーを使用した DB2 HACMP 相互引き受け – NFS フェイルオーバー

– nfs_server SP スイッチの別名 IP アドレスと nfs マウントの /nfshome が、ノード 87 から 88 へ移動。
– SP スイッチ arp コードには、すべてのスイッチ arp キャッシュをこの移動で更新する機能がある。

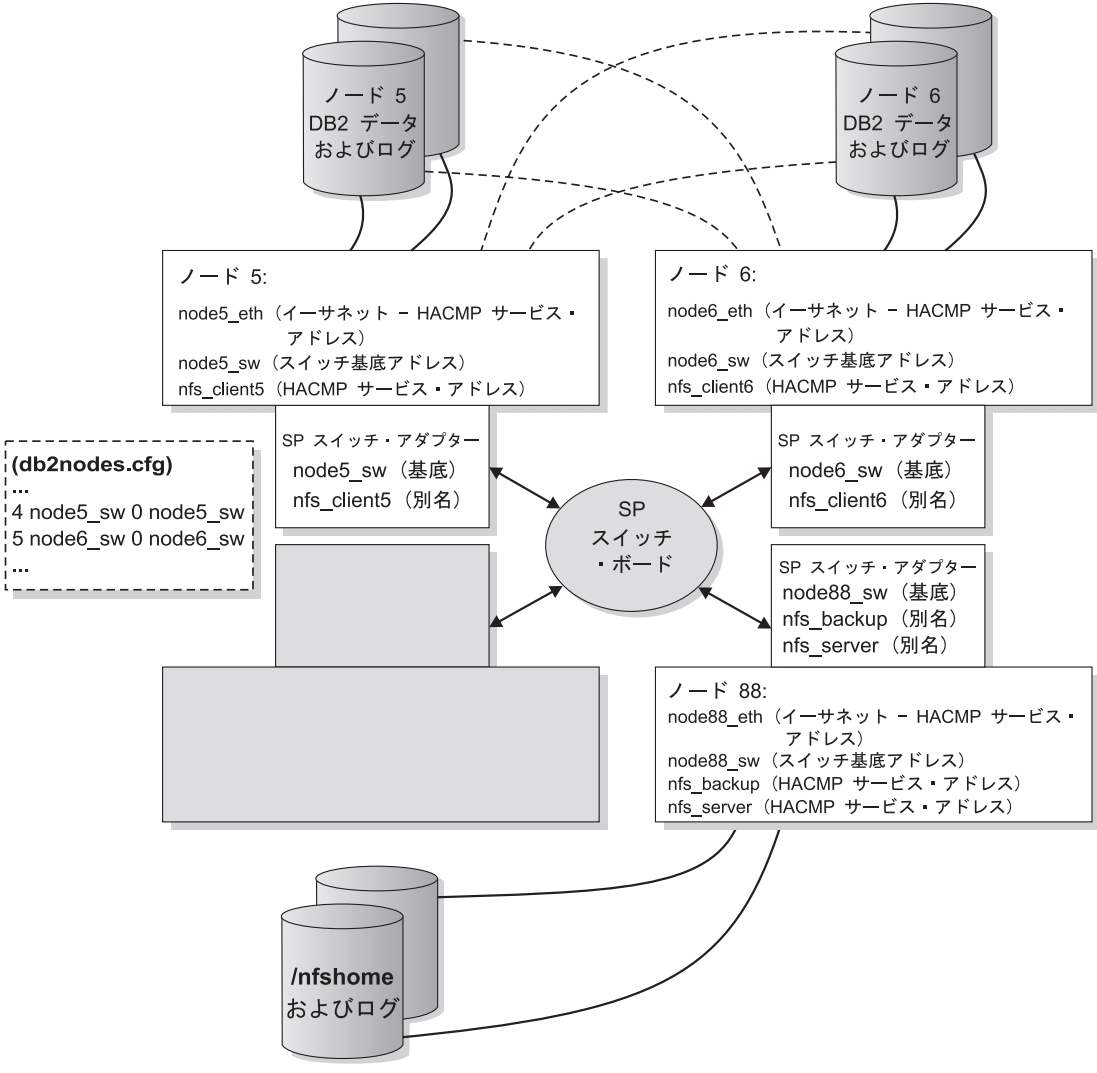


図 21. NFS フェイルオーバーを使用した相互引き受け - NFS フェイルオーバー

NFS フェイルオーバーを使用した DB2 HACMP 相互引き受け – DB2 フェイルオーバー

– スイッチ IP アドレス引き受けにより、他のサーバー（ADSM など）は接続性を維持できる。
– ノード 5 は DB2 の 2 つの論理ノードを実行する。

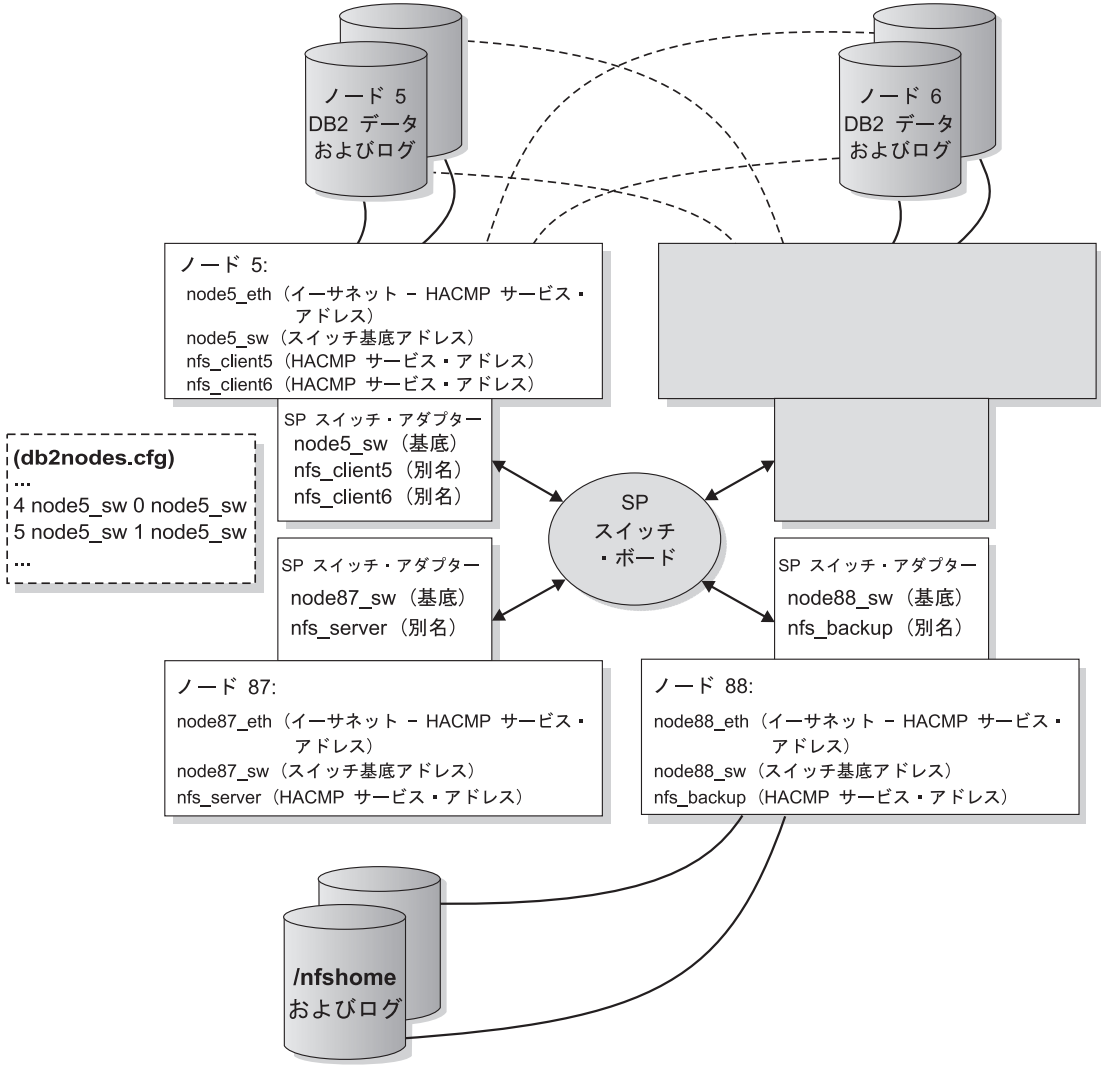


図 22. NFS フェイルオーバーを使用した相互引き受け - DB2 フェイルオーバー

DB2 HACMP ホット・スタンバイ構成 (201ページの図23 および 202ページの図24) については次のとおりです。

- HACMP アダプターはイーサネット、および SP スイッチ別名のブートおよびサービス別名用に定義され、基底アドレスは未定義のままです。 HACMP ネットワーク名では必ず、「HPS」ストリングを使用してください。
- NFS_server/nfshome は、スイッチ別名を使用して全ノードで /dbhome としてマウントされます。
- db2nodes.cfg ファイルには、 SP スイッチ基底アドレスが入っています。 DB2 データベース区画 (論理ノード) のフェイルオーバー後、 db2nodes.cfg ファイルは **DB2START** (RESTART) コマンドで変更されます。
- SP スイッチ別名ブート・アドレスは表示されません。

NFS フェイルオーバーを使用した DB2 HACMP ホット・スタンドバイ – 正常

注: ディスク配線によりますが、ホット・スタンドバイ・ノードは複数のノードを
バックアップすることができます。

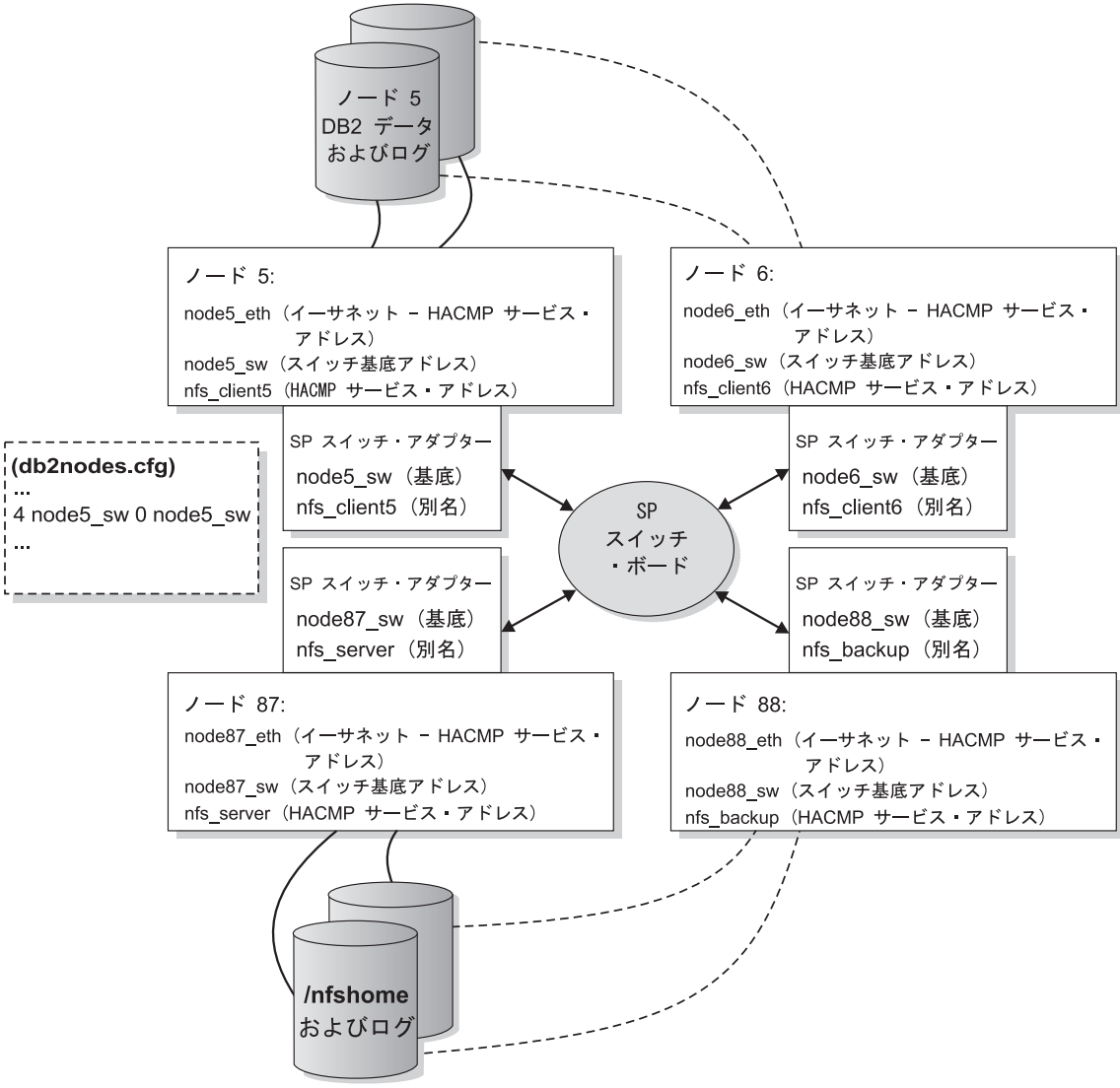


図 23. NFS フェイルオーバーを使用したホット・スタンドバイ - 正常

NFS フェイルオーバーを使用した DB2 HACMP ホット・スタンドバイ - DB2 フェイルオーバー

注: ディスク配線によりますが、ホット・スタンドバイ・ノードは複数のノードをバックアップすることができます。

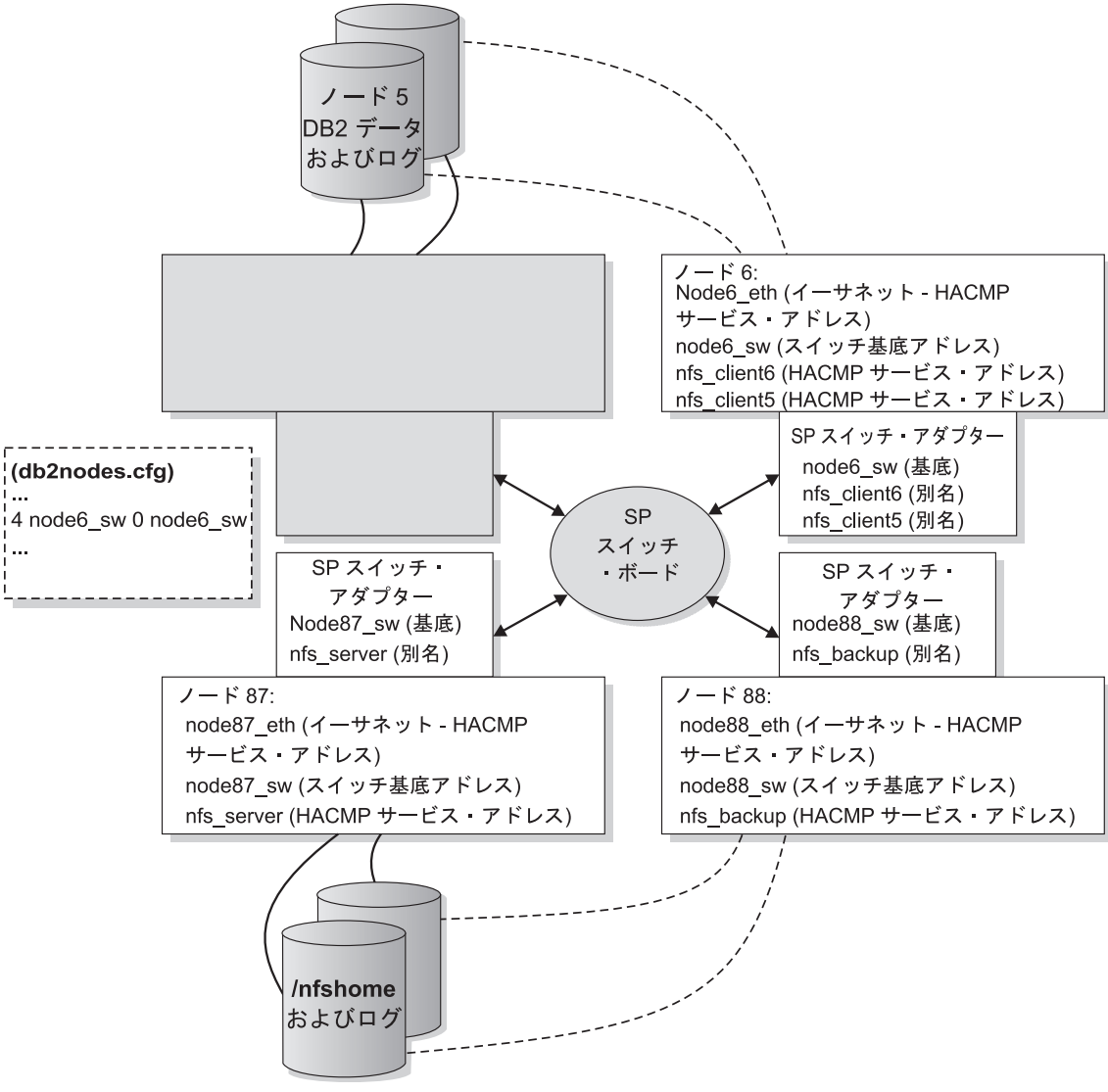


図 24. NFS フェイルオーバーを使用したホット・スタンドバイ - DB2 フェイルオーバー

NFS フェイルオーバー構成を使用しない DB2 HACMP 相互引き受け (203ページの図 25 および 204ページの図26) については次のとおりです。

- HACMP アダプターはイーサネット、および SP スイッチ基底アドレス用に定義されています。基底アドレスがサービス・アドレスとして HACMP に構成される場合、ブート・アドレスはない (「ハートビート」のみ) ということを忘れないでください。 SP スイッチには必ず、 HACMP ネットワーク名で「HPS」ストリングを使用してください。
- db2nodes.cfg ファイルには、 SP スイッチ基底アドレスが入っています。 DB2 データベース区画 (論理ノード) のフェイルオーバー後、 db2nodes.cfg ファイルは **DB2START** (RESTART) コマンドで変更されます。
- NFS フェイルオーバーの機能は表示されません。
- ノードは、異なる SP フレームに置くことができます。

NFS フェイルオーバーを使用しない DB2 HACMP 相互引き受け - 正常

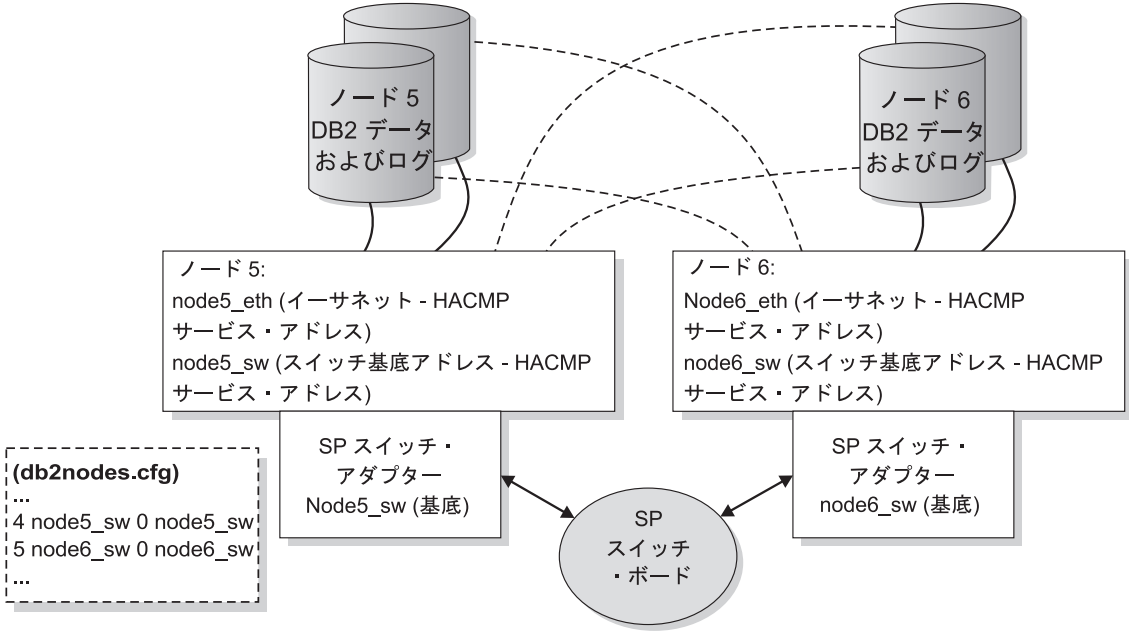


図 25. NFS フェイルオーバーを使用しない相互引き受け - 正常

NFS フェイルオーバーを使用しない DB2 HACMP 相互引き受け - DB2 フェイルオーバー

- ノード 5 は DB2 の 2 つの論理ノードを実行する。

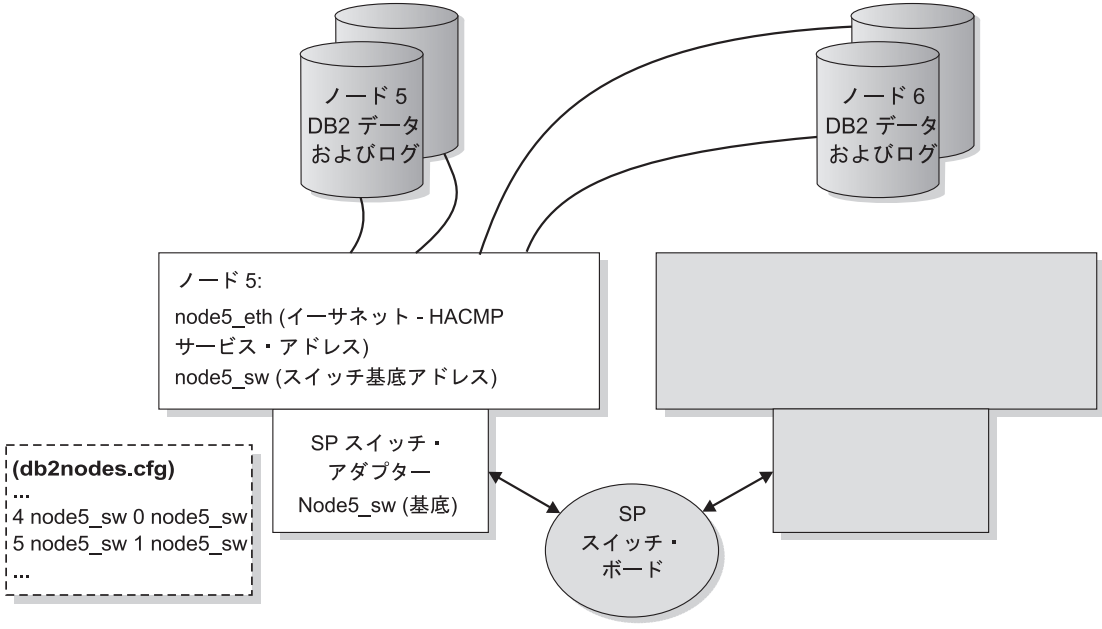


図 26. NFS フェイルオーバーを使用しない相互引き受け - DB2 フェイルオーバー

DB2 HACMP 始動に関する推奨事項

/etc/inittab でブート時での HACMP の始動を指定しないことをお勧めします。HACMP は、ノードのブート後に手作業で始動してください。この方法により、障害ノードの破壊的な保守を避けることができます。

「破壊的な保守」の例として、ハードウェアの障害と破損が生じたノードについて考慮してみます。フェイルオーバーは HACMP により自動的に開始され、回復は正常に完了します。しかし、障害ノードは、修復する必要があります。 HACMP がリブート時に始動するように /etc/inittab で構成された場合、このノードはブート完了後に再統合を試みますが、それはこのような状況では望ましいことではありません。

「非破壊的な保守」として、各ノードで HACMP を手作業で始動する場合を考慮してみましょう。手作業での始動により、他のノードに影響を与えないで障害ノードを修正し、再統合することができます。コントロール・ワークステーションから HACMP の始動および停止コマンドを制御するには、 ha_cmd スクリプトを使用できます。

注: はじめて DB2 インスタンスを作成するときは、次の項目が /etc/inittab ファイルに付加されます。

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

HACMP または HACMP ES が使用可能になっている場合、 HACMP 項目の前に上記の行を置き、 /etc/inittab ファイルを更新してください。以下は、/etc/inittab ファイル内のサンプル HACMP 項目です。

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

この項目は、 /etc/inittab ファイル内の最後の項目でなければなりません。

HACMP ES イベント・モニターおよびユーザー定義イベント

ユーザー定義イベントの例には、ページング・スペースが一定の満了状態に達するときに、 AIX 物理ノードで DB2 データベース区画をシャットダウンするイベントや、特定のノードで処理が停止する場合に、 DB2 データベース区画を再始動したりフェイルオーバー操作を開始したりするイベントが挙げられます。データベース区画のシャットダウン、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打ち切りといったユーザー定義イベントを説明する例が samples 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。この名前は固有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 クラスター管理プログラムは、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、回復処理を開始します。
- リソース・プログラム・パス。回復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。
- 回復タイプ。これは、将来の利用のために予約されています。
- 回復レベル。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。イベント管理プログラムのイベントで使用します。
- インスタンス・ベクトル。イベント管理プログラム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。イベント管理プログラム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。イベント管理プログラム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、関心のあ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 クラスター管理プログラム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。

以下は node_up のイベント定義の例です。

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

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

# 8) Predicate  - only used for event management events

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

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

この例は、 rules.hacmprd ファイルにあるイベント定義の 1 つです。この例では、node_up イベントが発生すると、回復プログラム /usr/sbin/cluster/events/node_up.rp が呼び出されます。値は、状態、回復タイプ、および回復レベルについて指定されます。ここでは、リソース変数、インスタンス変数、述部、およびリアーム述部用に 4 つの空白行があります。

非標準 HACMP ES イベントに対応するために、他のイベントを定義することができます。たとえば、/tmp ファイルが 90 % 以上いっぱいになったイベントを定義するには、 rules.hacmprd ファイルを修正する必要があります。

IBM AIX 並列システム・サポート・プログラム (PSSP) では、多くのイベントが事前定義されています。これらのイベントは、(ユーザー定義イベント内で使用されるときに) 次のように活用できます。

1. クラスターを停止する。
2. rules.hacmprd ファイルを編集する。ファイルを修正する前にバックアップしてください。事前定義した PSSP イベントを追加します。クラスター内の全ノード間で同期点を必要とする場合は、回復プログラムで **barrier** コマンドを使用します。 (バリ

ア・コマンドおよび回復プログラムの同期については、 HACMP 概念、インストール、管理の手引きで詳細をお読みください。)

3. クラスターを再始動する。クラスター管理プログラムの始動時に rules.hacmprd ファイルがメモリーに保管されます。変更を正確に実装するには、すべてのクラスターを再始動します。クラスター内に矛盾する規則があってはなりません。
4. クラスター管理プログラムは、 rules.hacmprd ファイルにあるすべてのイベントを使用します。

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。

リソース状態は、リソース変数によって表されます。リソース条件は、述部と呼ばれる式によって表されます。

イベント管理は、リソース・モニターからリソース変数を受け取ります。リソース・モニターは、特定のリソースの状態を観察し、その状態をいくつかのリソース変数に変換します。これらの変数は周期的にイベント管理に渡されます。イベント管理は、rules.hacmprd で HACMP ES クラスター管理プログラムが指定した述部を各リソース変数に適用します。述部が真として評価されると、イベントが生成されて、クラスター管理プログラムに送信されます。クラスター管理プログラムは票決プロトコルを開始し、回復プログラム内の「ノード・セット」で指定されたノード・セットで、 (イベント優先順位にしたがって) 回復プログラム・ファイル (xxx.rp) が実行されます。

回復プログラム・ファイル (*xxx*.rp) は、 1 つまたは複数のプログラム行でできています。各行は、次の書式で宣言されています。

```
relationship     command_to_run     expected_status     NULL
```

行の各値の間には少なくとも 1 つのスペースがなければなりません。「Relationship」は、どの種類のノードでどのプログラムを実行するかを決定する値です。次の 3 つの関係タイプがサポートされています。

- すべて (All)。現行 HACMP クラスターの全ノードで指定コマンドまたはプログラムが実行されます。
- イベント (Event)。イベントが発生したノードでのみ指定コマンドまたはプログラムが実行されます。
- その他 (Other)。イベントが発生しなかった 全ノードで、指定コマンドまたはプログラムが実行されます。

「Command_to_run」は引用区切りストリングで、実行可能プログラムに対する絶対パスが指定される場合もあれば、指定されない場合もあります。 HACMP で送達されるイベント・スクリプトでは、相対パス定義を使用できます。他のスクリプトまたはプログラムでは、 HACMP イベント・スクリプトと同じディレクトリーにある場合でも、絶対パスの指定をしなければなりません。

「Expected_states」は、指定コマンドまたはプログラムの戻りコードです。このコードは整数値または「x」です。「x」を使用すると、クラスター管理プログラムは戻りコードを無視します。他のすべてのコードでは、予期された戻りコードと等しくなければなりません。等しくならない場合、クラスター管理プログラムはイベント障害を検出します。このイベントを処理すると、 (手作業による介入で) 回復が行われるまで、処理が「停止」します。手作業による介入がなければ、ノードは他のノードとの同期を図れません。クラスター管理プログラムが全ノードを制御するには、全ノード間の同期が必須です。

「NULL」は、将来の利用のために予約されたフィールドです。「NULL」という語は、バリア行を除いて、個々の行の末尾に表示されます。 2 つのバリア・コマンドの間、または最初のコマンドの前で複数の回復コマンドを指定すると、ノード自身およびノード間で回復コマンドが並列処理されます。

バリア・コマンドは、全クラスター・ノード間の全コマンドを同期させるために使用します。ノードが回復プログラムのバリア・ステートメントをヒットすると、クラスター管理プログラムはそのノードでバリア・プロトコルを開始します。バリア・プロトコルは 2 フェーズ・プロトコルなので、すべてのノードが回復プログラム内のバリアに遭遇し、プロトコルの承認を「票決した」ときに、両方のフェーズが完了したことがすべてのノードに通知されます。

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 クラスター管理プログラムに通知します。
2. クラスター管理プログラムは `rules.hacmprd` ファイルを読み取り、イベントにマップされた回復プログラムを判別します。
3. クラスター管理プログラムは、一連の回復コマンドから成る回復プログラムを実行します。
4. 回復プログラムは、シェル・スクリプトまたは 2 進コマンドである回復コマンドを実行します。 (HACMP for AIX では、回復コマンドは、HACMP イベント・スクリプトと同じです。)
5. クラスター管理プログラムは、回復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。

HACMP ES スクリプト・ファイル

DB2 UDB EEE には、フェイルオーバー回復およびユーザー定義イベントの以下のサンプル・スクリプトが組み込まれています。スクリプト・ファイルは、`$INSTNAME/sqllib/samples/hacmp/es` ディレクトリーにあります。このスクリプトは、「現状のまま」でも動作し、また回復アクションをカスタマイズすることもできます。

- DB2 データベース区画回復スクリプト rc.db2pe。これは、データベース区画上で HACMP 構成を開始したり、停止したりするのに使用されるスクリプト・ファイルです。また、DB2 インスタンス所有者の NFS サーバーでは、 HACMP 開始および停止スクリプトとしても動作します。
- HACMP ES 用の DB2 固有のユーザー定義イベント。 6 つのデフォルト・イベントが含まれています。 1 つは処理回復用、2 つはページング・スペース用、 3 つは NFS および自動マウント回復用です。
- DB2 インスタンス NFS ファイル・サーバーのフェイルオーバー。このスクリプトはフェイルオーバーによって、 DB2 インスタンス用のファイル・システム・サーバーを回復してバックアップします。
- ネットワークのフェイルオーバー。スクリプト network_up_complete、network_back、 network_down_complete、および network_down は、 SP スイッチ・アダプターが失敗した場合に SP DB2 データベース区画のフェイルオーバーを可能にします。
- SP GUI 透視図のモニター・イベントを定義するためのスクリプト。イベントおよびハードウェア透視図を使用すれば、フェイルオーバーおよびユーザー定義回復のモニターをすることができます。透視図の詳細については、PSSP 管理に関する資料を参照してください。
- HACMP ES ノードでコア・スクリプトとイベントをインストールおよび削除するためのインストール・スクリプト。
- HACMP および DB2 構成をモニターするための SP 透視図問題管理 (pman) リソースを作成および削除するためのスクリプト・ファイル。

回復スクリプトは、回復操作を実行する各ノードにインストールする必要があります。スクリプト・ファイルは、 SP コントロール・ワークステーションまたは他の指定 SP ノードから集中方式でインストールすることができます。

1. スクリプトを $INSTNAME/sqllib/samples/hacmp/es ディレクトリーから、 SP コントロール・ワークステーションか、 **pcp** および **pexec** コマンドを実行できる別の SPノードにコピーする。これらのコマンドはインストール操作の際に必要です。
2. フェイルオーバー構成で (BUFFPAGE などの) キー・パラメーターを設定することにより、実際の環境に合わせて reg.parms.SAMPLE ファイルおよび failover.parms.SAMPLE ファイルをカスタマイズする。相互引き受け構成の場合、一般にフェイルオーバーの設定は、通常の設定サイズの半分またはそれ以下に調整されます。また、独自の名前 (「SAMPLE」ではない) で名前変更したこれらのファイルのコピーも使用します。
3. 必要に応じて、rc.db2pe ファイルで 5 つのパラメーター、 NFS_RETRIES、START_RETRIES、MOUNT_NFS、STOP_RETRIES、および FAILOVER をカスタマイズする。大半の実装では、再試行およびフェイルオーバーの設定で十分でしょう。MOUNT_NFS 設定は、NFS サーバー可用性のパッケージを使用するかどうかに応じて構成します。この設定は、rc.db2pe をマウントし、 DB2 インスタンス所有者の NFS ホーム・ディレクトリーを自分で検証する場合に設定してください。

FAILOVER パラメーターを「YES」に設定すると、 db2_proc_restart が呼び出され、 DB2 データベース区画の再始動が試行されます。再始動操作が失敗すると、HACMP はフェイルオーバーによりシャットダウンします。

4. イベント・ファイルで db2_paging_action、 db2_proc_recovery、nfs_auto_recovery をカスタマイズする。 pwq を編集して、 DB2 インスタンス所有者に変更します。 db2_paging_action をカスタマイズして、ページング・スペースの使用量が 90 % を超えた場合に実行するアクションを指定します。 (この処置が実行されない場合、DB2 データベース区画は停止します。) 回復アクションがさらに必要であれば、スクリプトを修正します。
5. db2_inst_ha を使用して、指定したノードにスクリプトとイベントをインストールする。 (これらのノードには、インストール前に HACMP ES をインストールしておく必要があります。) db2_inst_ha の構文は、次のとおりです。

   ```
   db2_inst_ha $INSTNAME/sqllib/samples/hacmp/es <nodelist> <DATABASENAME>
   ```

   ここで、

   ```
   $INSTNAME/sqllib/samples/hacmp/es は、スクリプトおよびイベントがある
      ディレクトリーを表します。
   <nodelist> は、ノードの pcp または pexec スタイルを表します。
      たとえば、1-16 または 1、2、3、4 です。
   <DATABASENAME> は、通常およびフェイルオーバーのパラメーター・
      ファイルに使用するデータベースの名前です。
   ```

   reg.parms.SAMPLE および failover.parms.SAMPLE は各ノードにコピーされ、reg.parms.DATABASENAME に名前変更されます。 db2_inst_ha は /usr/bin の各ノードにファイルをコピーし、次の HACMP イベント・ファイルを更新します。

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

6. HACMP を使用してシステムとスクリプトを構成する。
7. **create_db2_events** コマンドを使用して、問題管理リソース (pman) および SP GUI 透視図のモニター・イベントをインストールする。透視図では、構成およびカスタマイズがさらに必要です。透視図の詳細については、PSSP 管理の手引きを参照してください。
8. ha_db2stop コマンドを使用して、 HACMP ES フェイルオーバー回復を起こさずにデータベース区画をシャットダウンします。このコマンドを使用するには、ファイルをデータベース・ユーザーのホーム・ディレクトリーにコピーし、そのユーザーに許可と所有権が設定されていることを確認してください。フェイルオーバー回復なしでデータベースを停止するには、そのユーザーとして次のように入力します。

   ```
   ha_db2stop
   ```

   注: コマンドが戻るのを待機することが必要です。 ctrl-C 割り込みを使用して終了するか、処理を強制終了すると、フェイルオーバー回復を早まって再度使用可能にしてしまい、一部のデータ区画が停止しない可能性があります。

HACMP ES を使用した DB2 回復スクリプトの操作

HACMP ES は、次の方法で DB2 回復スクリプトを呼び出します。

- node_up_local (ノードの始動)

  HACMP は node_up 順序列を実行し、このノードで所有されている (カスケードを使用) か割り当てられている (回転を使用) リソース・グループで指定されたボリューム・グループ、論理ボリューム、ファイル・システム、および IP アドレスを獲得する。

  node_up_local_complete が実行されると、 rc.db2pe を含むアプリケーション・サーバー定義が開始され、この物理ノード上のアプリケーション・サーバー定義で指定されたデータベース区画が始動する。

  注: rc.db2pe は、始動モードで実行されると、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、reg.parms.DATABASE で指定された DB2 パラメーターを調整します。

  各ノードは始動時にこの順序に従います。複数の HACMP クラスターを持っていて、並列始動すると、一度に複数のノードが始動します。

- node_down_remote (フェイルオーバー)

  HACMP は、指定引き受けノード上のリソース・グループで指定されたボリューム・グループ、論理グループ、ファイル・システム、および IP アドレスを獲得する。

  node_down_remote_complete が実行されると、HACMP は、このデータベース区画用のリソース・グループで指定されたアプリケーション・サーバー始動スクリプトとして rc.db2pe を実行する。

  注: rc.db2pe は、相互引き受けモードで実行されると、そこで実行される DB2 データベース区画を停止し、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、 failover.parms.DATABASE で指定された DB2 パラメーターを調整した後、物理引き受けノード上の両方のデータベース区画を始動します。

- node_up_remote (障害ノードの再統合 - リソース・グループの引き受けは必ずカスケードが行う)

  node_up_remote が古い引き受けノードで実行されると、アプリケーション・サーバー定義により、rc.db2pe が停止モードで実行される。

  注: rc.db2pe は、再統合モード (相互引き受け) で実行されると、そこで実行される両方の DB2 データベース区画を停止し、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、reg.parms.DATABASE で指定された DB2 パラメーターを調整した後、この物理引き受けノードで保持されるデータベース区画だけを始動します。

古い引き受けノードは、再統合されるノードに所有されるリソース・グループで指定されたボリューム・グループ、論理ボリューム、ファイル・システム、および IP アドレスを解放する。

HACMP は、再統合されるノードが現在所有するリソース・グループで指定されたボリューム・グループ、論理ボリューム、ファイル・システム、および IP アドレスを再獲得する。

node_up_local_complete が実行されると、 rc.db2pe を含むアプリケーション・サーバー定義が開始され、この再統合物理ノード上のアプリケーション・サーバー定義で指定されたデータベース区画が始動する。

注: rc.db2pe は、始動モードで実行されると、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、reg.parms.DATABASE で指定された DB2 パラメーターを調整します。

- node_down_local (ノード停止または引き受けによる停止)

  node_down_local が停止ノードで実行されると、アプリケーション・サーバー定義により、 rc.db2pe が停止モードで実行される。

  注: rc.db2pe は、停止モードで実行されると、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、failover.parms.DATABASE で指定された DB2 パラメーターを調整した後、 DB2 データベース区画 (これは引き受け用) を停止します。

  HACMP は、ノードが現在所有するリソース・グループで指定されたボリューム・グループ、論理ボリューム、ファイル・システム、および IP アドレスを解放する。

- db2_proc_recovery (db2 処理停止)

  すべてのノードで db2_proc_restart スクリプトが実行される。障害の発生したノードでは、正しい DB2 データベース区画が再始動されます。

- db2_paging_recovery (ページング・スペースの回復)

  すべてのノードで db2_paging_action スクリプトが実行される。ノードでページング・スペースが全容量の 70 % を超える場合は、 wall コマンドが発行される。ノードでページング・スペースが全容量の 90 % を超える場合は、この物理ノード上の DB2 データベース区画が停止され、再始動されます。

- nfs_auto_recovery (nfs または自動マウント処理の失敗)

  すべてのノードで、rc.db2pe スクリプトが NFS モードで実行される。 NFS 処理の実行が停止すると、実行が再開されます。自動マウント処理の実行が停止すると、同様の方法で再開されます。

- network_down_complete (ネットワーク障害 - SP スイッチ)

net_down スクリプトが呼び出される。この呼び出しにより、SP スイッチ・ネットワークとしてのネットワークが検査され、ダウンしているかどうかが検証されます。ダウンしていれば、ユーザー定義の時間間隔だけ待機されます。デフォルトの時間間隔は、100 秒です。

network_up_complete イベントで指示されたとおりに SP スイッチ・ネットワークが復旧すると、回復は実行されない。

時間制限に達すると、HACMP はフェイルオーバーで停止する。

注: すべてのイベントは、SP 問題管理および SP 透視図 GUI を使用してモニターできます。

他のスクリプト・ユーティリティー

他にも、以下のようなスクリプト・ユーティリティーを使用できます。

- ha_cmd は、コントロール・ワークステーションから SP ノードで HACMP を始動するために使用するコマンドです。構文は次の通りです。

  ```
  ha_cmd <noderange> <START|STOP|TAKE|FORCE>
  ```

  ここで、

  ```
  <noderange> は SP ノード範囲の pcp または pexec スタイルです。
     たとえば、「ha_cmd 3-6 START」とすると、HACMP はノード 3、4、5、6 で始動します。
              「ha_cmd 5 TAKE」は、ノード 5 の HACMP をシャットダウンして
              相互引き受けを行います。
  ```

- ha_mon は、 SP コントロール・ワークステーションから HACMP の hacmp_out ファイルをモニターするためのコマンドです。構文は次の通りです。

  ```
  ha_mon <node>
  ```

  ここで、

  ```
  <node> は、モニターされる SP ノードを表します。

  ha_mon は、指定するノード上の /tmp/hacmp.out ファイルに「-f を追加」します。
  ```

- db2_turnoff_recov は、すべての HACMP (非フェイルオーバー) 回復を一時的に使用不能にする、極めて珍しい状況下で使用するコマンドです。 DB2 処理、ページング、NFS、または自動マウント回復が開始されなくなります。この機能は、HACMP 規則ファイルから該当する回復のイベント・スタンザを削除します。 HACMP を停止し、再始動する必要があります。構文は次の通りです。

  ```
  db2_turnoff_recov <nodelist>
  ```

- db2_turnon_recov は、HACMP (非フェイルオーバー) 回復を再び使用可能にするためのコマンドです。このコマンドは、 db2_turnoff_recov の後に使用して HACMP の規則ファイルを復元し、ユーザー定義イベントによる回復を実行できるようにします。 HACMP を停止し、再始動する必要があります。構文は次の通りです。

  ```
  db2_turnon_recov <nodelist>
  ```

HACMP クラスターのモニター

HACMP ES にある既存のモニター・ユーティリティーに加えて、 DB2 HACMP ES 構成をモニターする SP 問題管理 (pman) イベントを作成するためのスクリプトが提供されています。 SP コントロール・ワークステーションから HACMP 状況をモニターするには、次のようにします。

- コントロール・ワークステーションで HACMP クライアント・コードをインストールします。
- /usr/sbin/cluster/etc/clhosts ファイルを編集し、モニターしたいノードの SP イーサネット IP アドレスを含めます。
- コマンド startsrc -s clinfo を呼び出して、クラスターのモニターを開始します。

HACMP は、クラスターをモニターするためのインターフェース、つまり、/usr/sbin/cluster/clstat を提供します。

HACMP RS およびユーザー定義イベントの SP 透視図 GUI で問題管理モニターを使用するには、次のようにします。

1. create_db2_events <nodelist> を呼び出す。ここで、*nodelist* には、 pcp または pexec スタイル・ノードが含まれます。このスクリプトは透視図により、モニターするための 5 つの pman イベントを作成します。

   注: これらのイベントの作成では、リソース変数 PSSP.pm.User_state12-16 が使用されます。これらのリソース変数がすでに他の目的で使用されている場合は、create_db2_events および update_db2_events を更新して、異なるリソース変数を使用する必要があります。

2. コントロール・ワークステーションで透視図を開始する。ランチ・パッドから、イベント透視図を選択します。 5 つのイベント、つまり、db2_hacmp_recovery、db2_process_recovery、db2_paging_err、 db2_nfs_err、および Errlog_PERM_entry があるはずです。
3. 個々のイベントをダブルクリックする。表示される画面上で、イベントの条件を (定義表の中で) 登録してください。下矢印の横にある Name: "unnamed" をクリックし、条件として指定するイベントと同じ名前を選択します。 「応答オプション (Response Options)」 タブを選択します。画面上部のボタン (「透視図にメッセージを送信するイベント・セッション (Send Message to Perspectives event session)」) をクリックします。これらのイベントの実行で使用するコマンド、errlog 項目、および SNMP トラップを指定できます。イベント・ログの表示は透視図セッションでのみ保持されます。したがって、個々のイベントに応じた AIX エラー・ログ項目を作成することもできます。**「了解 (OK)」**を選択し、ウィンドウを閉じます。
4. 「透視図 (Perspectives)」ランチ・パッドから、「ハードウェア透視図 (hardware Perspective)」を選択する。
5. ハードウェア・フレーム GUI が表示されたら、「視点 (View)」—>「モニター (Monitor)」を選択する。すると、SP でモニターできるイベントのリストが表示され

ます。リストを下までスクロールすると、2 つの追加イベントがあります。一方は HACMP DB2 回復用 (db2_ha_ind) で、他方は SP ノード PERM エラー用 (Errlog_PERM_mon) です。モニターしたいイベントを選択します。 (イベントが発生すると、ノードは赤い「X」を表示します。モニターするすべての条件を満たすと、ノードの表示は緑色になります。) host_responds、 switch_responds、および node_power_LED が通常使用されます。また、DB2 HACMP 回復やノード上の PERM エラーもモニターできます。

注: pman および透視図用の db2_hacmp_mon および db2_hacmp_recovery 変数は、HACMP クラスター状況を反映しません。むしろ、これらの変数は、 DB2 を開始したり停止したりするための rc.db2pe 操作の状況を反映します。 HACMP clstat モニターには「実際の」 HACMP 状況が表示され、 HACMP クラスター状態が反映されます。 db2_hacmp_ind で HACMP 状況に似たモニターを反映させたい場合は、 /etc/inittab ファイルに次の行を追加します。

```
haind:2:wait:/usr/bin/db2_update_events HAIND OFF 2>&1 >/dev/null
```

実装で NetView を使用する計画であれば、構成のモニターに HAVIEW (HACMP の一部) を使用することを検討してください。この製品の構成については、 NetView の資料を参照してください。

DB2 SP HACMP ES のインストール

DB2 ユニバーサル・データベースで HACMP ES をインストールをするための計画に役立つ資料として、以下にインストールおよび移行プロセスを段階的に示した説明を行います。

DB2 SP HACMP ES の新規インストール

HACMP ES をインストールするには、次のようにします。

1. 個々の SP ノードに AIX オペレーティング・システムをインストールする (SP インストールおよび管理の手引きを参照してください)。コントロール・ワークステーションでも各 SP ノードでも、適切なページング・スペースを使用できることを確認します。他の修正可能構成パラメーターと一緒にスイッチ構成が考慮され、実装されていることを確かめてください。使用する SP モニター (透視図) を適切な位置に置きます。 SP dsh、pcp、および pexec コマンドが動作することを確かめます。
2. 実際のデータベース・レイアウトを設計する。この設計には少なくとも、使用するノードの数、物理ノードに対する DB2 データベース区画のマッピング、ノードまたは区画あたりのディスク要件、表スペースに関する考慮事項を含めます。また、DB2 インスタンスの主要な所有者となる人、およびその所有者と他のユーザーが必要とするアクセス許可も考慮してください。
3. 冗長アダプター、ミラーリングされるディスク、ディスクの兄弟接続をはじめとする、外部 SSA ディスクの構成を計画する。

4. データベースのレイアウトおよび SSA 構成を使用して、HACMP の計画、インストール、および管理の手引きにある HACMP のワークシートを完成させる。
5. 外部 SSA ディスク構成を実装する。全ドライブでマイクロコード・レベルが一貫していることを確認し、 Maymap ユーティリティーを使用して、ワークシートを検証し、相違をなくします。
6. 各 SP ノードに DB2 UDB EEE をインストールする。
7. 各 SP ノードに HACMP ES をインストールする。
8. **db2_inst_ha** コマンドを使用して、 SP パッケージに DB2 UDB EEE HACMP ES をインストールする。
9. DB2 のメイン・インスタンス・ユーザーを作成し、すべてのノードにアクセスできることを確認する。これにより、高可用性ユーザーになったというわけではありません。 SP コントロール・ワークステーション上で一時的に SP ユーザーとなることが可能です。
10. DB2 インスタンスおよびデータベースを作成する。 **db2start** を呼び出してこの DB2 インスタンスおよびデータベースを操作可能にしてから、次のステップに進む前に、**db2stop** を呼び出します。
11. HACMP を追加する前にデータベースをロードしたい場合は、この時点で実行してください。
12. HACMP ワークシートと本書の情報にしたがって、 SP ノード・トポロジーおよびリソース上に HACMP ES を構成する。
13. メイン DB2 インスタンス・ユーザーの NFS サーバー・ノードから始めて、本書で指定された内容に従って、全ノード上で (`/etc/security/user` および `/etc/passwd` を修正して) このユーザーを変更する。このユーザーは、高可用性 NFS ユーザーとなります。また、このノードおよびバックアップでは、`/etc/exports` が更新されます。すべてのノードは、スイッチ別名 IP アドレス全体で、 NFS (各ノードの `/etc/filesystems` で項目を持つ) を使用して、このディレクトリーをマウントすることができます。
14. メイン・インスタンス・ユーザーのホーム・ディレクトリーを「tar 圧縮」し、新しい位置のホーム・ディレクトリーを「tar 圧縮解除」する。
15. 個々の SP ノードに NFS ファイル・システムを作成し、新しいメイン・インスタンスのホーム・ディレクトリーをマウントする。
16. NFS サーバー・ノードで HACMP を始動する。 `/tmp/hacmp.out` を調べて、HACMP の始動が成功したことを確かめてください。 **ha_mon** コマンドは、このファイルを作成されたとおりにモニターするために使用できます。
17. 他のノードを一度に 1 つずつ立ち上げる。 `/tmp/hacmp.out` を調べて、各ノードが正常に完了したことを確かめます。 **ha_mon** コマンドは、このファイルを作成されたとおりにモニターするために使用できます。
18. 透視図および問題管理を使用して、任意のモニターをセットアップする。

19. 各ノードで並行保守アクションをシミュレートして、各ノードでフェイルオーバーが機能するかどうかを検証する。 (TAKE オプションを指定して) **ha_cmd** コマンドを使用すれば、引き受けで HACMP を正常に停止できます。 `/tmp/hacmp.out` を調べ、モニター・ツールを使用して、引き受けおよび再統合が成功したことを検証します。

DB2 SP HACMP ES の移行

HACMP をインストールしていない環境から HACMP をインストールした環境へ移行する場合は、以下の概説を考慮してください。

1. 既存の外部ディスクを、高可用性で、兄弟接続され、ミラーリングされた構成に変換する。異なるノード上の異なる論理ボリュームの名前は兄弟接続時に固有名にしなければならない ことを念頭に置いて、この構成を達成するために余分のハードウェアおよびディスクを追加します。これは、ボリューム・グループ、論理グループ、およびファイル・システムに適用されます。
2. HACMP の計画、および本書にあるワークシートを含め、関連ワークシートを完成させる。
3. 外部 SSA ディスク構成の変更を実装する。全ドライブでマイクロコード・レベルが一貫していることを確認し、 Maymap ユーティリティーを使用して、ワークシートでの検証と相違の訂正を行います。

   注: RAID5 構成では SSA ディスクがサポートされています。同一の RAID ループにある 2 つの SSA アダプターだけが、許可されている構成です。兄弟接続されている RAID の HACMP 構成には、 1 つのノードにつき 1 つしかアダプターはサポートされていません。この構成では、アダプターはディスクへのアクセスの障害の単一ポイントであり、アダプターの故障を検出したり HACMP フェイルオーバー・イベントにこれをプロモートしたりするために、さらに構成することが勧められています。 SSA アダプターに障害が起きた場合には、 AIX エラー通知を使用してノードを構成することが最も簡単な方法です。 AIX エラー通知についての詳細は、*HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide* を参照してください。
4. 各 SP ノードに HACMP ES をインストールする。
5. **db2_inst_ha** コマンドを使用して、 SP パッケージに DB2 UDB EEE HACMP ES をインストールする。
6. HACMP ワークシートと本書の情報にしたがって、 SP ノード・トポロジーおよびリソース上に HACMP ES を構成する。
7. メイン DB2 インスタンス・ユーザーの NFS サーバー・ノードから始めて、本書で指定された内容に従って、全ノード上で (`/etc/security/user` および `/etc/passwd` を修正して) このユーザーを変更する。このユーザーは、高可用性 NFS ユーザーとなります。また、このノードおよびバックアップでは、`/etc/exports` が更新されます。すべてのノードは、スイッチ別名 IP アドレス全体

で、 NFS (各ノードの /etc/filesystems で項目を持つ) を使用して、このディレクトリーをマウントすることができます。

8. メイン・インスタンス・ユーザーのホーム・ディレクトリーを「tar 圧縮」し、新しい位置のホーム・ディレクトリーを「tar 圧縮解除」する。
9. 個々の SP ノードに NFS ファイル・システムを作成し、新しいメイン・インスタンスのホーム・ディレクトリーをマウントする。
10. NFS サーバー・ノードで HACMP を始動する。 /tmp/hacmp.out を調べて、HACMP の始動が成功したことを確かめてください。 **ha_mon** コマンドは、このファイルを作成されたとおりにモニターするために使用できます。
11. 他のノードを一度に 1 つずつ立ち上げる。 /tmp/hacmp.out を調べて、各ノードが正常に完了したことを確かめます。 **ha_mon** コマンドは、このファイルを作成されたとおりにモニターするために使用できます。
12. 透視図および問題管理を使用して、任意のモニターをセットアップする。
13. 各ノードで並行保守アクションをシミュレートして、各ノードでフェイルオーバーが機能するかどうかを検証する。 (TAKE オプションを指定して) **ha_cmd** コマンドを使用すれば、引き受けで HACMP を正常に停止できます。 /tmp/hacmp.out を調べ、モニター・ツールを使用して、引き受けおよび再統合が成功したことを検証します。

DB2 SP HACMP ES ワークシート

以下に示すワークシートは、外部 SSA ディスク構成の準備で記入した HACMP ワークシートと一緒に使用します (HACMP のワークシートは、 HACMP の計画、インストール、および管理の手引きにあります)。それぞれに、完全な例とブランク・ワークシートが準備されています。

次の図には、最初のサンプル・ワークシートに記されている外部ディスクに対するデータベース構成が示されています。このデータベースの作成で使用したステートメントは、次のとおりです。

```
db2 create database pwq on /newdata
```

障害点を持たない論理ボリュームでは、 SSA 外部アダプターと外部 SSA ディスクの両方がミラーリングされています。この図では、**maymap** コマンドの出力に似た構成が説明されています。 Maymap は (AIXTOOLS で使用できる) ユーティリティーで、外部 SSA ディスク構成を表示できます。このユーティリティーをセットアップの計画時に使用することをお勧めします。

DB2 4 ノード・データベースの外部ディスクのセットアップ例

– 高可用性のための兄弟接続を示しています。

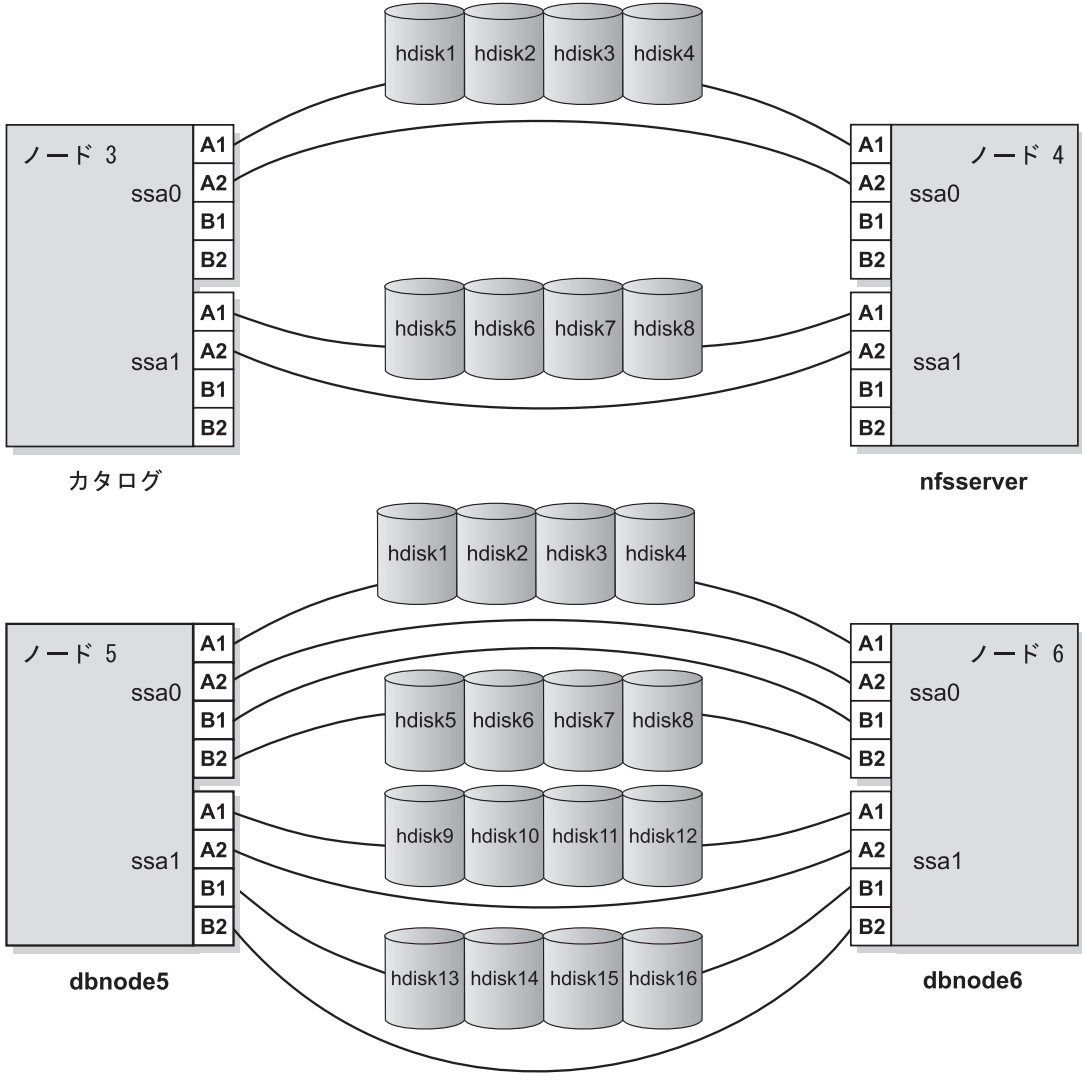


図 27. DB2 4 ノード・データベースの外部ディスクのセットアップ例

以下の表を検討する前に、ボリューム・グループに対する定足数設定、および論理ボリュームに対するミラーリング書き込みの整合性設定に関して、 HACMP 資料を読んでおくことをお勧めします。両者で使用する設定は、可用性とパフォーマンスに直接影響を与えます。必ず、両方の設定を検討し、その意味を理解してください。「定足数」と「ミラーリング書き込みの整合性」の両方で使用する一般的な設定値は、「off」です。

表 12. HACMP ボリューム・グループ、論理ボリューム、およびファイル・システム

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者
3	havg3	8	hlv300	10	2	hdisk1 hdisk5	/newdata /pwq /NODE0003	hlog301	Catalognode マウント・ポイント ; ノード 4	root *
3	havg3	8	hlog301	1	2	hdisk1 hdisk5	適用外	適用外	Catalognode jfslog; ノード 4	root *
3	havg3	8	hlv301	10	2	hdisk2 hdisk6	適用外	適用外	Catalognode rawtemp スペース ; ノード 4	pwq **
4	havg4	8	hlv400	10	2	hdisk3 hdisk7	/dbmnt	hlog401	nfsserver pwq ホーム ; ノード 3	root *
4	havg4	8	hlog401	1	2	hdisk3 hdisk7	適用外	適用外	nfsserver jfslog; ノード 3	root *
5	havg5	8	hlv500	10	2	hdisk1 hdisk9	/newdata/ pwq/ NODE0005	HLOG501	Dbnode5 マウント・ポイント; ノード 6	root *
5	havg5	8	hlog501	1	2	hdisk1 hdisk9	適用外	適用外	Dbnode5 jfslog; ノード 6	root *
5	havg5	8	hlv501	10	2	hdisk2 hdisk10	適用外	適用外	Dbnode5 raw temp スペース ; ノード 6	pwq **
5	havg5	8	hlv502	100	2	hdisk2 hdisk10	適用外	適用外	Dbnode5 raw 表スペース ; ノード 6	pwq **
5	havg5	8	halv503	100	2	hdisk3 hdisk11	適用外	適用外	Dbnode5 raw 表スペース ; ノード 6	pwq **
5	havg5	8	halv504	100	2	hdisk3 hdisk11	適用外	適用外	Dbnode5 raw 表スペース ; ノード 6	pwq **
5	havg5	8	halv505	100	2	hdisk4 hdisk12	/dbdata5	hlog501	Dbnode6 システム表スペース; ノード 6	root *
6	havg6	8	hlv600	10	2	hdisk5 hdisk13	/newdata/ pwq/ NODE0006	hlog601	Dbnode6 マウント・ポイント; ノード 5	root *
6	havg6	8	hlog601	1	2	hdisk5 hdisk13	適用外	適用外	Dbnode6 jfslog; ノード 5	root *
6	havg6	8	hlv601	10	2	hdisk6 hdisk14	適用外	適用外	Dbnode6 raw temp スペース; ノード 5	pwq **
6	havg6	8	hlv602	100	2	hdisk6 hdisk14	適用外	適用外	Dbnode6 raw 表スペース; ノード 5	pwq **
6	havg6	8	hlv603	100	2	hdisk7 hdisk15	適用外	適用外	Dbnode6 raw 表スペース; ノード 5	pwq **
6	havg6	8	hlv604	100	2	hdisk7 hdisk15	適用外	適用外	Dbnode6 raw 表スペース; ノード 5	pwq **

表 12. HACMP ボリューム・グループ、論理ボリューム、およびファイル・システム (続き)

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者
6	havg6	8	hlv605	100	2	hdisk8 hdisk16	/dbdata6	hlog601	Dbnode6 システム表スペース; ノード 5	root *

注:

1. * jfs ファイル・システムの論理ボリュームおよびログはルート許可を保持します。
2. ** ロー・データベース・スペースは、 /dev ロー・ファイル項目 (/dev/rxxxx) に対するデータベース・ユーザー許可を取得します。

表 13. HACMP ボリューム・グループ、論理ボリューム、およびファイル・システム (ブランク)

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者

表 13. HACMP ボリューム・グループ、論理ボリューム、およびファイル・システム (ブランク) (続き)

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者

表 13. HACMP ボリューム・グループ、論理ボリューム、およびファイル・システム (ブランク) (続き)

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者

表 14. HACMP NFS サーバーの計画

SP ノード	外部ファイル・システム	バックアップ・ノード	SP スイッチ・ブートおよびサービス IP 別名のペア	マウントするファイル・システム (/etc/filesystems)	データベース・ホーム・ディレクトリーとして指定するファイル・システム	ファイル・システムをエクスポートする宛先アドレス (/etc/exports)
3	/dbmnt	4	nfs_boot_3 nfs_client_3	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
4	/dbmnt	3	nfs_server_boot nfs_server	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
5	適用外	適用外	nfs_boot_5 nfs_client_5	nfs_server:/ dbmnt as /dbi	/dbi/pwq	適用外
6	適用外	適用外	nfs_boot_6 nfs_client_6	nfs_server:/ dbmnt as /dbi	/dbi/pwq	適用外

表 14. HACMP NFS サーバーの計画 (続き)

SP ノード	外部ファイル・システム	バックアップ・ノード	SP スイッチ・ブートおよびサービス IP 別名のペア	マウントするファイル・システム (/etc/filesystems)	データベース・ホーム・ディレクトリーとして指定するファイル・システム	ファイル・システムをエクスポートする宛先アドレス (/etc/exports)
注: 1. /etc/passwd は必ず全ノードで同じものを指定する。これは、コントロール・ワークステーションから同期化できます。 2. 外部ファイル・システムにデータベース・インスタンス所有者の許可があることを確かめる。 3. /etc/filesystems には、マウント・パラメーター、つまり、hard、bg、intr、および rw を指定しなければならない。 4. /etc/exports に指定される `-root=ip1:ip2:ip3` は、サーバーとそのバックアップに限られる。						

表 15. HACMP NFS サーバーの計画 (ブランク)

SP ノード	外部ファイル・システム	バックアップ・ノード	SP スイッチ・ブートおよびサービス IP 別名のペア	マウントするファイル・システム (/etc/filesystems)	データベース・ホーム・ディレクトリーとして指定するファイル・システム	ファイル・システムをエクスポートする宛先アドレス (/etc/exports)

表 15. HACMP NFS サーバーの計画 (ブランク) (続き)

SP ノード	外部ファイル・システム	バックアップ・ノード	SP スイッチ・ブートおよびサービス IP 別名のペア	マウントするファイル・システム (/etc/filesystems)	データベース・ホーム・ディレクトリーとして指定するファイル・システム	ファイル・システムをエクスポートする宛先アドレス (/etc/exports)

表 15. HACMP NFS サーバーの計画 (ブランク) (続き)

SP ノード	外部ファイル・システム	バックアップ・ノード	SP スイッチ・ブートおよびサービス IP 別名のペア	マウントするファイル・システム (/etc/filesystems)	データベース・ホーム・ディレクトリーとして指定するファイル・システム	ファイル・システムをエクスポートする宛先アドレス (/etc/exports)

第7章 Windows オペレーティング・システムでの高可用性

データベース・システムは、マシンが故障した場合に、その故障したマシン上のデータベース・サーバーが別のマシンで実行できるようにセットアップすることができます。Windows NT では、 Microsoft Cluster Server (MSCS) でフェイルオーバーのサポートを実装できます。 MSCS を使用するには、 MSCS 機能を持つ Windows NT Version 4.0 Enterprise Edition をインストールする必要があります。

MSCS は、物理ディスクと IP アドレスのフェイルオーバー・サポートなど、クラスター環境で障害検出およびリソースの再始動の両方を実行することができます。 (失敗したマシンが再びオンラインになるとき、リソースがフォールバックするように前もって構成しておかない限り、自動的にフォールバックされることはありません。詳細については 239ページの『フォールバックの考慮事項』を参照してください。)

フェイルオーバーをサポートするように DB2 インスタンスを使用可能にする前に、以下の計画段階のステップを実行してください。

1. データ・ストレージに使用したいディスクを決定します。各データベース・サーバーは、使用するために最低 1 つのディスクに割り当てられる必要があります。データを保管するのに使用するデータは共用ディスク・サブシステムに接続されているべきであり、さらに MSCS ディスク・リソースとして構成される必要があります。
2. リモート要求をサポートするのに使用したいデータベース・サーバーごとに、必ず 1 つの IP アドレスがあるようにします。

フェイルオーバー・サポートを設定する場合、既存のインスタンスに設定したり、フェイルオーバーの実装時に新しいインスタンスを作成したりすることができます。

フェイルオーバー・サポートを使用可能にするには、以下のステップを実行します。

1. DB2MSCS ユーティリティーに入力ファイルを作成します。
2. **db2mscs** コマンドを呼び出します。
3. 区分データベース・システムを使用中である場合、データベース・ドライブ・マッピングを登録して相互引き受けを使用可能にします。 240ページの『区分データベース環境で相互引き受け構成にデータベース・ドライブ・マッピングを登録する』を参照してください。

フェイルオーバー・サポート用のインスタンスを使用可能にした後、構成は、228ページの図28 のようになります。

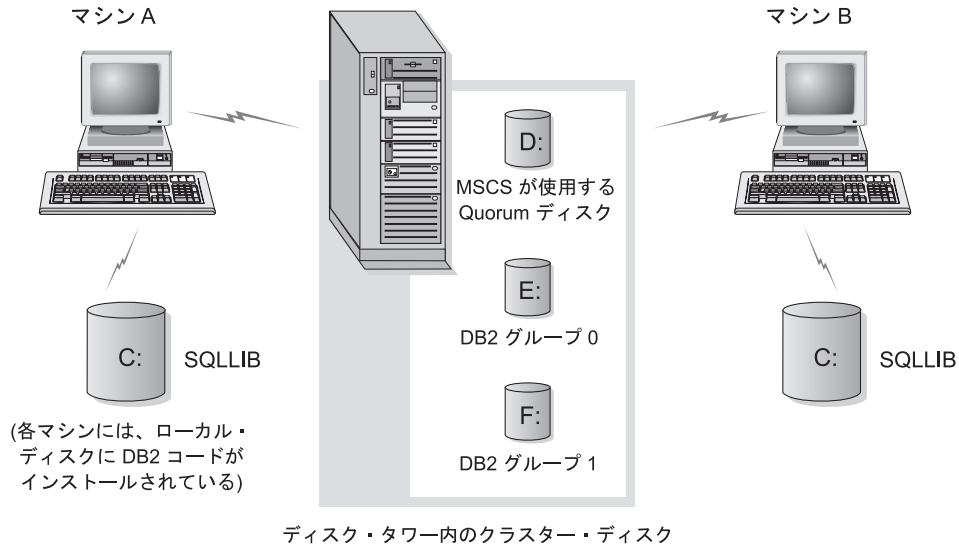


図 28. MSCS 構成の例

以下のセクションでは、フェイルオーバー・サポートの様々なタイプ、およびそれらを実装する方法を説明します。以下で説明されるステップのいずれかを実行する前に、MSCS ソフトウェアを MSCS クラスターで使用したいすべてのマシン上にインストールしておく必要があります。さらに、DB2 もすべてのマシンにインストールしてください。

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・スタンドバイ
- 相互引き受け

現在、MSCS は 2 つのマシンのクラスターをサポートしています。

区分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・スタンド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、相互引き受け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 つを相互引き受けの構成を使用するため、 2 つをホット・スタンド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・スタンドバイ構成

ホット・スタンドバイ構成では、 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。区分データベース・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。 図29 にホット・スタンドバイ構成の例を示します。

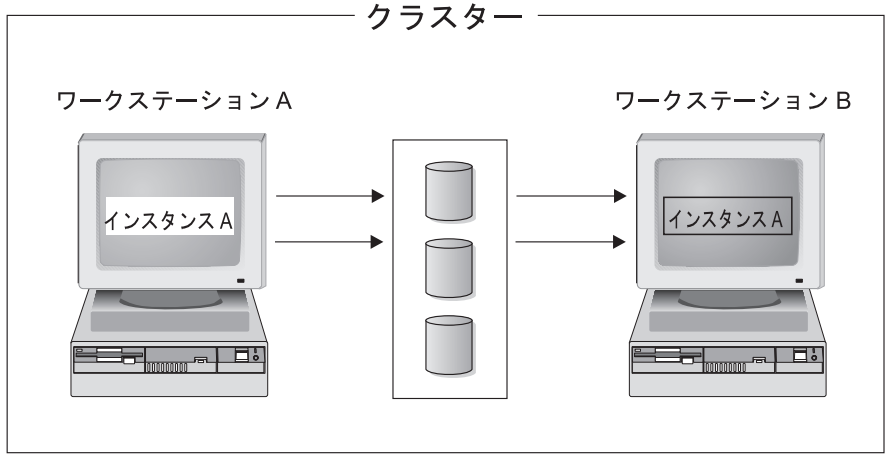


図 29. ホット・スタンドバイ構成

相互引き受け構成

相互引き受け構成では、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。相互引き受け構成では、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活動状態であることができます。 230ページの図30 に相互引き受け構成の例を示します。

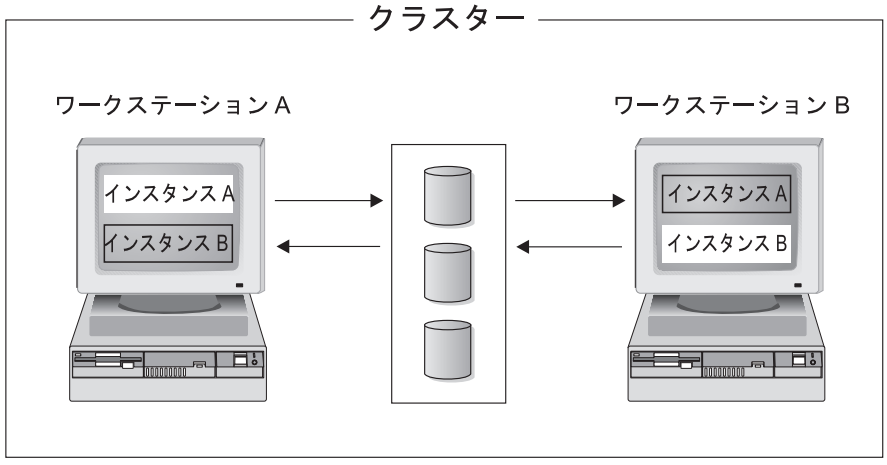


図 30. 相互引き受け構成

DB2MSCS ユーティリティーの使用方法

DB2MSCS ユーティリティーは、DB2 用のインフラストラクチャーを作成し、Microsoft Cluster Service (MSCS) サポートを使用して Windows NT 環境でフェイルオーバーをサポートするのに使用します。このユーティリティーを使用して、単一区画環境、および区分データベース環境の両方でフェイルオーバーを使用可能にできます。

DB2MSCS ユーティリティーを正常に実行するには、 Cluster Service がリソース DLL (`db2wolf.dll`) を見付けることができなければなりません。これは、`%ProgramFiles%¥SQLLIB¥bin` ディレクトリーにあります。 DB2 UDB バージョン 7 インストール・プログラムでは、 PATH システム環境変数が `%ProgramFiles%¥SQLLIB¥bin` ディレクトリーを指すように設定されます。ただし、Windows 2000 オペレーティング・システム上で実行している場合は、インストール後にマシンをリブートする必要はありません。 DB2MSCS ユーティリティーを実行したい場合は、マシンをリブートして PATH 環境変数を Cluster Service 用に更新する必要があります。

db2mscs コマンドは、インスタンスが所有するマシン上の、各インスタンスごとに 1 回呼び出します。 MSCS クラスター中のあるマシンで実行する DB2 インスタンスが 1 つだけである場合、このユーティリティーはホット・スタンドバイ構成をセットアップします。 MSCS クラスター中の各マシンで実行する 1 つのインスタンスがある場合、インスタンスが所有する各マシンで DB2MSCS を 1 回実行し、相互引き受け構成をセットアップします。

DB2MSCS ユーティリティーは以下の処理を実行します。

1. DB2MSCS.CFG と呼ばれる入力ファイルから、必要な MSCS および DB2 パラメーターを読み取ります。入力パラメーターのフル・セットに関する情報は、『DB2MSCS.CFG ファイルの指定』を参照してください。
2. 入力ファイル中のパラメーターの妥当性検査をします。
3. DB2 リソース・タイプを登録します。
4. MSCS グループ (複数の場合もある) を作成して MSCS および DB2 リソースを入れます。
5. IP リソースを作成します。
6. Network Name リソースを作成します。
7. MSCS ディスクをグループに移動します。
8. DB2 リソース (複数の場合もある) を作成します。
9. DB2 リソースに必要なすべての従属関係を追加します。
10. クラスター化されていない DB2 インスタンスを、クラスター化されたインスタンスに変換します。
11. すべてのリソースをオンライン上に出します。

コマンド構文は、以下のとおりです。

ここで、

-f:*input_file*

MSCS ユーティリティーが使用する 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と、 DB2MSCS ユーティリティーは現行ディレクトリーにある DB2MSCS.CFG ファイルを読み取ります。

DB2MSCS.CFG ファイルの指定

DB2MSCS.CFG ファイルは、 DB2MSCS ユーティリティーが読み取るパラメーターを含む ASCII テキスト・ファイルです。書式 PARAMETER_KEYWORD=*parameter_value* を使用して、個別の行で各入力パラメーターを指定します。次に例を示します。

```
CLUSTER_NAME=WOLFPACK
GROUP_NAME=DB2 Group
IP_ADDRESS=9.21.22.89
```

2 つの構成ファイルのサンプルが、 /SQLLIB ディレクトリーの /CFG サブディレクトリーに入っています。 1 つ目の DB2MSCS.EE は、単一区画データベース環境の例です。 2 つ目の DB2MSCS.EEE は、区分データベース環境の例です。

DB2MSCS.CFG ファイルの 2 つのパラメーターは次のとおりです。

DB2_INSTANCE

DB2 インスタンスの名前。インスタンス名が指定されていない 場合、デフォルトのインスタンス (DB2INSTANCE 環境変数の値) が使用されます。

このパラメーターにはグローバルな効力範囲があり、 DB2MSCS.CFG ファイル中で 1 度だけ指定します。

このパラメーターは任意指定です。

例:

```
DB2_INSTANCE=DB2
```

インスタンスはすでに存在しているはずです。インスタンスの作成についての詳細は、*DB2 エンタープライズ拡張エディション (Windows 版) 概説およびインストール* を参照してください。

DB2_LOGON_USERNAME

DB2 サービスのログオン・アカウントの名前。

このパラメーターにはグローバルな効力範囲があり、 DB2MSCS.CFG ファイル中で 1 度だけ指定します。

このパラメーターは、 DB2 エンタープライズ拡張エディション インスタンスにのみ必要です。

例:

```
DB2_LOGON_USERNAME=db2user
```

DB2_LOGON_PASSWORD

DB2 サービスのログオン・アカウントのパスワード。
DB2_LOGON_USERNAME パラメーターが提供されたのに DB2_LOGON_PASSWORD パラメーターが提供されない場合、 DB2MSCS ユーティリティーはパスワードを要求します。パスワードは、コマンド行に入力されても表示されません。

このパラメーターにはグローバルな効力範囲があり、 DB2MSCS.CFG ファイル中で 1 度だけ指定します。

このパラメーターは、 DB2 エンタープライズ拡張エディション インスタンスにのみ必要です。

例:

```
DB2_LOGON_PASSWORD=xxxxxx
```

CLUSTER_NAME

MSCS クラスターの名前。この行に続いて指定されるすべてのリソースは、別の CLUSTER_NAME タグが指定されるまでこのクラスターで作成されます。

このパラメーターは、各クラスターごとに 1 度指定します。

このパラメーターは任意指定です。指定されないと、ローカル・マシン上の MSCS クラスター名が使用されます。

例:

```
CLUSTER_NAME=WOLFPACK
```

GROUP_NAME

MSCS グループの名前。このパラメーターが指定されると、 MSCS グループがなかった場合に新しいグループが作成されます。グループがあった場合、ターゲット・グループとして使用されます。この行に続いて作成される MSCS リソースはすべて、別の GROUP_NAME キーワードが指定されるまでこのグループに作成されます。

このパラメーターは、各グループごとに 1 度指定します。

このパラメーターは必須です。

例:

```
GROUP_NAME=DB2 Group
```

DB2_NODE

現行の MSCS グループに含まれるデータベース区画サーバー (ノード) のノード番号。複数の論理ノードが同じマシン上に存在する場合、各ノードは個別の DB2_NODE キーワードを必要とします。

このパラメーターは、DB2 リソースが正しい MSCS グループで作成されるように、 GROUP_NAME パラメーターの後で指定します。

このパラメーターは、 DB2 エンタープライズ拡張エディション インスタンスにのみ必要です。

例:

```
DB2_NODE=0
```

IP_NAME

IP アドレス・リソースの名前。 IP_NAME の名前は任意ですが、固有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 接続に必要です。このパラメーターを、区分データベース環境でインスタンス所有のマシンに指定する必要があります。このパラメーターは、単一区画データベース環境では任意指定です。

例:

```
IP_NAME=IP Address for DB2
```

注: DB2 クライアントは、この IP リソースの TCP/IP アドレスを使用して、TCP/IP ノード・エントリーのカタログを作成する必要があります。MSCS IP アドレスを使用することにより、データベース・サーバーが他の

マシンに対して障害を起こした場合でも、 IP アドレスが故障したマシン上で使用可能なので、 DB2 クライアントはデータベース・サーバーに接続できます。

IP リソースの属性は次のとおりです。

IP_ADDRESS

IP リソースの TCP/IP アドレス。このキーワードは、先行する IP リソースに TCP/IP アドレスを設定するのに使用します。

このパラメーターは、IP_NAME パラメーターが指定される場合に必要です。

例:

```
IP_ADDRESS=9.21.22.34
```

IP_SUBNET

先行する IP リソースのサブネット・マスク。

このパラメーターは、IP_NAME パラメーターが指定される場合に必要です。

例:

```
IP_SUBNET=255.255.255.0
```

IP_NETWORK

先行する IP リソースが所属している MSCS ネットワークの名前。このパラメーターが指定されていないと、システムが検出する最初の MSCS ネットワークが使用されます。

このパラメーターは任意指定です。

例:

```
IP_NETWORK=Token Ring
```

NETNAME_NAME

Network Name リソース名。 Network Name リソースを作成するのに指定します。

このパラメーターは、単一区画データベース環境の場合、任意指定です。区分データベース環境の場合は必須です。

例:

```
NETNAME_NAME=Network name for DB2
```

Network Name リソースの属性は次のとおりです。

NETNAME_VALUE

Network Name の値。

このパラメーターは、NETNAME_NAME パラメーターが指定される場合に必要です。

例:

```
NETNAME_VALUE=DB2SRV
```

NETNAME_DEPENDENCY

Network Name リソースの従属関係リスト。各 Network Name リソースには、IP アドレス・リソース上に従属関係がなければなりません。このパラメーターが指定されないと、 Network Name リソースはグループで最初の IP リソースとの従属関係になります。

このパラメーターは任意指定です。

例:

```
NETNAME_DEPENDENCY=IP Address for DB2
```

DISK_NAME

現行のグループに移動される物理ディスク・リソース名。必要なだけディスク・リソースを指定してください。

注:

1. ディスク・リソースはすでに存在しているはずです。
2. DB2MSCS ユーティリティーが MSCS サポートに DB2 インスタンスを構成すると、インスタンス・ディレクトリーがグループで*最初の* MSCS ディスクにコピーされます。インスタンス・ディレクトリーに異なる MSCS ディスクを指定するには、 INSTPROF_DISK パラメーターを使用します。

例:

```
DISK_NAME=Disk E:
DISK_NAME=Disk F:
```

INSTPROF_DISK

MSCS ディスクに DB2 インスタンス・ディレクトリーを入れるように指定する任意指定パラメーター。このパラメーターが指定されていない と、DB2MSCS ユーティリティーは、インスタンス・ディレクトリーと同じグループに属する*最初の* MSCS ディスクを使用します。

DB2 インスタンス・ディレクトリーは、 X:¥DB2PROFS ディレクトリー (X は MSCS ディスクのドライブ文字) の下の MSCS ディスクに作成されます。

例:

```
INSTPROF_DISK=Disk E:
```

TARGET_DRVMAP_DISK

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任意指定パラメーター。ノードと同じグループに属さない MSCS ディスク上にデータベースが作成された場合、データベース区画はターゲット・ドライ

ブ・マップ・ディスクに含まれます。このパラメーターが指定されない場合、データベース・ドライブ・マッピングは、 DB2DRVMP ユーティリティーを使用して、手作業で登録しなければなりません。

例:

```
TARGET_DRVMAP_DISK = Disk E:
```

単一区画データベース・システムにフェイルオーバーをセットアップする

DB2MSCS ユーティリティーを単一区画データベース・システムに対して実行すると、1 つの MSCS グループに DB2 およびすべての従属する MSCS リソース (IP アドレス、 Network Name、およびディスク) が入れられます。たとえば、単一区画データベース・システムの DB2MSCS.CFG ファイルの内容は次のようになります。

```
#
#  DB2MSCS.CFG for a single-partition database system
#
DB2_INSTANCE=DB2
CLUSTER_NAME=MSCS
GROUP_NAME=DB2 Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

2 つの単一区画データベース・システムに相互引き受け構成をセットアップする

2 つの単一区画データベース・システムを個別のマシンのそれぞれにセットアップして、片方のマシン上のデータベース・システムが障害を起こしても、もう一方の MSCS ノードで再開するようにできます。

この構成にフェイルオーバー・サポートをセットアップするには、インスタンスが所有する各マシン上で DB2MSCS ユーティリティーを 1 度実行する必要があります。各データベース・システムごとに、構成ファイルを調整してください。

DB2 インスタンスの名前を DB2A および DB2B と想定すると、 DB2A インスタンスの DB2MSCS.CFG ファイルは次のようになります。

```
#
#  DB2MSCS.CFG for first single-partition database system
#
DB2_INSTANCE=DB2A
CLUSTER_NAME=MSCS
GROUP_NAME=DB2A Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
```

```
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

DB2B インスタンスの DB2MSCS.CFG ファイルは次のようになります。

```
#
#  DB2MSCS.CFG for second single-partition database system
#
DB2_INSTANCE=DB2B
CLUSTER_NAME=MSCS
GROUP_NAME=DB2B Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk F:
```

完全な例は、242ページの『例 - 相互引き受けに 2 つの単一区画インスタンスをセットアップする』を参照してください。

区分データベース・システムに複数の MSCS クラスターをセットアップする

複数の区分データベース・システムに対して DB2MSCS ユーティリティーを実行している場合、 1 つの MSCS グループが、システムに参加する各物理マシンごとに作成されます。 DB2MSCS.CFG ファイルには複数のセクションが含まれていなければならず、各セクションには GROUP_NAME パラメーターおよびそのグループに必要なすべての従属リソースに、異なる値がなければなりません。

さらに、各 MSCS グループのデータベース区画サーバーごとに、 DB2_NODE パラメーターを指定する必要があります。複数の論理ノードがある場合、各論理ノードには個別の DB2_NODE キーワードが必要です。

たとえば、 4 つのマシン上で 4 つのデータベース区画サーバーから構成される複数区画データベース・システムがあり、相互引き受け構成を使用して 2 つの MSCS クラスターを構成すると仮定します。 DB2MSCS.CFG 構成ファイルを次のようにセットアップします。

```
#
#  DB2MSCS.CFG for one partitioned database system with
#  multiple clusters
DB2_INSTANCE=DB2MPP
DB2_LOGON_USERNAME=db2user
DB2_LOGON_PASSWORD=xxxxxx
CLUSTER_NAME=MSCS1
# Group 1
GROUP_NAME=DB2 Group 1
DB2_NODE=0
```

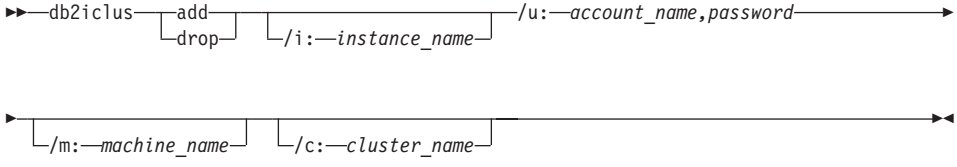
```
IP_NAME=...

...
# Group 2
GROUP_NAME=DB2 Group 2
DB2_NODE=1
IP_NAME=...

...

CLUSTER_NAME=MSCS2
# Group 3
GROUP_NAME=DB2 Group 3
DB2_NODE=2
IP_NAME=...

...
# Group 4
GROUP_NAME=DB2 Group 4
DB2_NODE=3
IP_NAME=...

...
```

完全な例は、245ページの『例 - 相互引き受けに 4 つのノードを持つ区分データベース・システムをセットアップする』を参照してください。

MSCS システムの保守

DB2MSCS ユーティリティーを実行すると、 MSCS クラスターのすべてのマシンにフェイルオーバー・サポート用のインフラストラクチャーが作成されます。マシンからサポートを除去するには、 drop オプションを付けて **db2iclus** コマンドを使用します。マシンでのサポートを再度使用可能にしたい場合、 add オプションを使用します。

コマンド構文は、以下のとおりです。



ここで、

add

MSCS クラスターにフェイルオーバー・サポートを追加して、使用可能にします。すると、DB2 リソース (データベース・サーバー) がこのマシンでフェイルオーバーできます。

drop	MSCS クラスターからフェイルオーバー・サポートを除去して、マシンから除去します。
/i: *instance_name*	インスタンスの名前。 (このパラメーターは、DB2INSTANCE 環境変数の設定をオーバーライドします。)
/u: *account_name*, *password*	DB2 サービスのログオン・アカウント名として使用されるドメイン・アカウント。たとえば、次のようなものがあります。 `/u:domainA¥db2nt,password` このパラメーターは、 add オプションにのみ必要です。
/m:*machine_name*	MSCS クラスターに追加したい、または MSCS クラスターから除去したいコンピューター名。フェイルオーバー・サポートを修正しているマシン以外のマシンからコマンドを実行する場合、このオプションを指定します。
/c: *cluster_name*	LAN と呼ばれる MSCS クラスター名。この名前は、MSCS クラスターが最初に作成されるときに指定されます。

フォールバックの考慮事項

デフォルトでは、グループは元の (障害を起こした) マシンにフォールバックするようには設定されません。フェイルオーバー後にフォールバックするように手作業で DB2 グループを構成しない限り、フェイルオーバーの原因が解決された後は代替 MSCS ノードで実行し続けます。

DB2 グループを、元のマシンに自動的にフォールバックするように構成する場合、DB2 リソースを含む DB2 グループのすべてのリソースは、元のマシンが使用可能になるとすぐにフォールバックします。フォールバックの間にデータベース接続が存在している場合、 DB2 リソースをオフラインにすることはできず、フォールバック処理は失敗します。

フォールバック処理中にすべてのデータベース接続をデータベースから強制的にオフにしたい場合、 DB2_FALLBACK レジストリー変数を ON に設定します。この変数は、次のように設定してください。

```
db2set DB2_FALLBACK=ON
```

このレジストリー変数の設定後は、クラスター・サービスをリブートしたり再始動したりする必要はありません。

区分データベース環境で相互引き受け構成にデータベース・ドライブ・マッピングを登録する

区分データベース環境でデータベースを作成する場合、データベースの作成位置を示すドライブ文字を指定することができます。

注: 単一区画データベース環境にはデータベース・ドライブ・マッピングを設定しません。

CREATE DATABASE コマンドを実行するとき、指定されるドライブが、インスタンスに参加するすべてのマシンで同時に使用可能になっていることが期待されます。これは不可能なので、DB2 はデータベース・ドライブ・マッピングを使用して、各マシンごとに同じドライブに別の名前を割り当てます。

たとえば、DB2 という名前の DB2 インスタンスに、 2 つのデータベース区画サーバーが入っているとします。

```
NODE0 がマシン WOLF_NODE_0 で活動状態
NODE1 がマシン WOLF_NODE_1 で活動状態
```

また、共用ディスク F: が NODE0 として同じグループに所属しており、共用ディスク F: が NODE1 として同じグループに所属していると仮定します。

共用ディスク E にデータベースを作成するには、コマンドは次のようになります。

```
db2 create database mppdb on e:
```

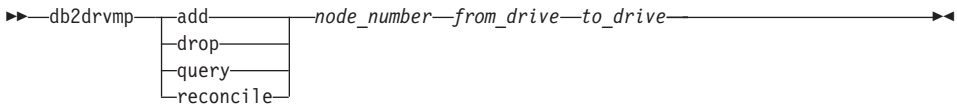
コマンドを成功させるために、ドライブ E: は両方のマシンで使用可能でなければなりません。相互引き受け構成では、各データベース区画サーバーが異なるマシン上で活動状態であり、クラスター・ディスク E: は 1 つのマシンにだけ使用可能です。この状態では、 CREATE DATABASE コマンドは常に失敗します。

この問題を解決するには、データベース・ドライブを次のようにマップしてください。

```
NODE0 については、マッピングはドライブ F: からドライブ E: へ
NODE1 については、マッピングはドライブ E: からドライブ F: へ
```

こうすると、 NODE0 のドライブ F: へのデータベース・アクセスがドライブ E: にマップされ、 NODE1 のドライブ E: へのデータベース・アクセスがドライブ F: にマップされます。ドライブ・マッピングを使用して、 CREATE DATABASE コマンドは NODE0 にドライブ E: で、および NODE1 にドライブ F: でデータベース・ファイルを作成します。

db2drvmp コマンドを使用してドライブ・マッピングをセットアップします。コマンド構文は、以下のとおりです。



パラメーターは、以下のとおりです。

add 新しいデータベース・ドライブ・マップを割り当てます。

drop 既存のデータベース・ドライブ・マップを除去します。

query データベース・マップの照会。

reconcile 登録内容が損傷を受けたときにデータベース・マップ・ドライブを修理します。詳細については、242ページの『データベース・ドライブ・マッピングの調整』を参照してください。

node_number ノード番号。このパラメーターは、add および drop 操作に必要です。

from_drive マップ元のドライブ文字。このパラメーターは、add および drop 操作に必要です。

to_drive マップ先のドライブ文字。このパラメーターは、add 操作に必要です。他の操作には適用しません。

F: から E: のデータベース・ドライブ・マッピングを NODE0 に設定したい場合、次のコマンドを使用できます。

```
db2drvmp add 0 F E
```

注: データベース・ドライブ・マッピングは表スペース、コンテナー、または他のデータベース・ストレージ・オブジェクトには適用されません。

同様に、E: から F: のデータベース・ドライブ・マッピングを NODE1 に設定したい場合、次のコマンドを出します。

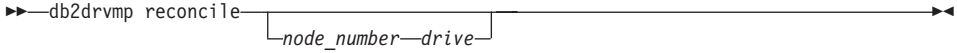
```
db2drvmp add 1 E F
```

注: データベース・ドライブ・マッピングのセットアップ、または変更の効果はすぐに出るわけではありません。データベース・ドライブ・マッピングを活動化するには、 Cluster Administrator ツールを使用して DB2 リソースをオフラインにしてからオンラインにします。

DB2MSCS.CFG ファイルで TARGET_DRVMAP_DISK キーワードを使用すると、ドライブ・マッピングを自動的に行うようにすることができます。

データベース・ドライブ・マッピングの調整

データベースが実際にデータベース・ドライブ・マッピングを持つマシン上で作成されると、マップが隠しファイル中のドライブで保管されます。これはデータベース・ドライブが、データベースの作成後に除去されることのないようにするためのものです。たとえば、間違えてデータベース・ドライブ・マッピングを除去してしまう場合、データベース・ドライブ・マッピングを*調整* します。マップを調整するには、データベースを含む各データベース区画サーバーごとに **db2drvmp reconcile** コマンドを実行します。コマンド構文は、以下のとおりです。



パラメーターは、以下のとおりです。

node_number
: 修理されるノードのノード番号。 *node_number* が指定されない場合、コマンドはすべてのノードへのマッピングを調整します。

drive
: 調整するドライブ。ドライブが指定されない場合、コマンドはすべてのドライブへのマッピングを調整します。

db2drvmp コマンドは、データベース区画サーバーが管理するデータベース区画でマシン上のすべてのドライブを走査し、データベース・ドライブ・マッピングを必要に応じてレジストリーに再度適用します。

例 - 相互引き受けに 2 つの単一区画インスタンスをセットアップする

この例の目的は、相互引き受け構成でフェイルオーバー・サポートを使用し、 2 つの単一区画データベース・インスタンスをセットアップすることです。この例では、4 つのサーバーが 2 つの MSCS クラスターに構成されます。相互引き受け構成を使用することにより、マシンのいずれかが障害を起こした場合、そのマシン用に構成されたデータベース・サーバーは、 MSCS ソフトウェアを使用して構成されたとおり、代替マシンにフェイルオーバーし、その代替マシン上で実行します。

結果の構成には 2 つの MSCS クラスターがあります。各クラスターには、次のものがあります。

- 2 つのサーバー。それぞれ 64 MB のメモリーおよび 2 GB のローカル SCSI ディスクが 1 つあります。
- それぞれが 2 GB の 3 つの SCSI ディスクを持つ 1 つの SCSI ディスク・タワー。

加えて、各マシンには 1 つの 100X イーサネット・アダプター・カードがインストールされています。

各マシンには次のソフトウェアがインストールされています。

- MSCS 機能を持つ Windows NT Version 4.0 Enterprise Edition
- DB2 ユニバーサル・データベース エンタープライズ・エディション バージョン 7

結果のネットワーク構成は次のとおりです。

サーバー 1:	サーバー 2:
• マシン名: db2test1	• マシン名: db2test2
• TCP/IP ホスト名: db2test1	• TCP/IP ホスト名: db2test2
• IP アドレス: 9.9.9.1 (サブネット・マスク: 255.255.255.0	• IP アドレス: 9.9.9.2 (サブネット・マスク: 255.255.255.0
• MSCS クラスター名: ClusterA	• MSCS クラスター名: ClusterA

ネットワーク中の両方のマシンが TCP/IP で構成され、イーサネット 100 T-base Hub を使用して私用の LAN に接続されます。ドメイン・ネーム・サーバー (DNS) はありませんが、すべてのマシンにはローカル TCP/IP hosts ファイルがあります。このファイルには以下のエントリーが含まれています。

```
9.9.9.1 db2test1 # for Server 1
9.9.9.2 db2test2 # for Server 2
9.9.9.3 ClusterA # for MSCS ClusterA
9.9.9.4 db2tcp1 # for DB2 remote client connection to Server 1
9.9.9.5 db2tcp2 # for DB2 remote client connection to Server 2
```

仮のタスク

以下のタスクを実行する前に、両方のマシンが同じドメイン、 DB2NTD に所属していると仮定します。

1. ローカル管理者グループのメンバーである DB2 のドメイン・アカウントを、 DB2 が実行する予定のマシン上で作成します。すべてのタスクを実行するためのアカウントを使用します。
   - ユーザー名を db2nt に設定します。
   - パスワードを db2nt に設定します。
2. MSCS 機能をマシン db2test1 および db2test2 にインストールします。
   - MSCS クラスターに ClusterA という名前を付けます。
   - クラスター IP アドレスは 9.9.9.3 です。
   - 共用ディスク D: は MSCS ソフトウェアによって使用されます。
   - 共用ディスク E: および F: は DB2 によって使用されます。
3. マシン db2test1 に、 DB2 ユニバーサル・データベース エンタープライズ・エディション バージョン 7 をインストールします。ローカル・ドライブである C:¥SQLLIB に、ソフトウェアをインストールします。

4. マシン db2test2 に、 DB2 ユニバーサル・データベース エンタープライズ・エディション バージョン 7 をインストールします。ローカル・ドライブである C:¥SQLLIB に、ソフトウェアをインストールします。

次のステップは、各インスタンスごとに DB2MSCS.CFG ファイルをセットアップしてから、各インスタンスごとに DB2MSCS ユーティリティーを実行することです。

DB2MSCS ユーティリティーを実行する

db2test1 マシンをセットアップするには、以下のタスクを実行してください。

1. マシン db2test1 では、ユーザー db2nt としてログオンします。パスワードは db2nt です。
2. まだ存在しなければ、DB2 インスタンス DB2A を作成します。インスタンスを作成するコマンドは次のとおりです。

   ```
   db2icrt DB2A
   ```

3. マシン db2test1 で DB2 インスタンスに DB2MSCS.CFG ファイルを設定します。

   ```
   #
   #  DB2MSCS.CFG for database system
   #  on machine db2test1
   DB2_INSTANCE=DB2A
   CLUSTER_NAME=ClusterA
   #
   # Group 1
   GROUP_NAME=DB2A Group
   IP_NAME=IP Address for DB2A
   IP_ADDRESS=9.9.9.4
   IP_SUBNET=255.255.255.0
   IP_NETWORK=ClusterA
   NETNAME_NAME=Network name for DB2A
   NETNAME_VALUE=DB2SRV1
   NETNAME_DEPENDENCY=IP Address for DB2A
   DISK_NAME=Disk E:
   INSTPROF_DISK=Disk E:
   ```

4. 以下のように DB2MSCS ユーティリティーを実行してください。

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

5. db2nt アカウントからログアウトします。
6. マシン db2test2 でユーザー db2nt としてログオンします。このユーザーはローカル管理者グループに属しています。パスワードは db2nt です。
7. まだ存在しなければ、DB2 インスタンス DB2B を作成します。インスタンスを作成するコマンドは次のとおりです。

   ```
   db2icrt DB2B
   ```

8. マシン db2test2 で DB2 インスタンスに DB2MSCS.CFG ファイルを設定します。

   ```
   #
   #  DB2MSCS.CFG for database system
   #  on machine db2test2
   ```

```
DB2_INSTANCE=DB2B
CLUSTER_NAME=ClusterA
#
# Group 1
GROUP_NAME=DB2B Group
IP_NAME=IP Address for DB2B
IP_ADDRESS=9.9.9.5
IP_SUBNET=255.255.255.0
IP_NETWORK=ClusterA
NETNAME_NAME=Network name for DB2B
NETNAME_VALUE=DB2SRV2
NETNAME_DEPENDENCY=IP Address for DB2B
DISK_NAME=Disk F:
INSTPROF_DISK=Disk F:
```

9. 以下のように DB2MSCS ユーティリティーを実行してください。

```
db2mscs -f:DB2MSCS.CFG
```

10. db2nt アカウントからログアウトします。

例 - 相互引き受けに 4 つのノードを持つ区分データベース・システムをセットアップする

この例の目的は、相互引き受け構成でフェイルオーバー・サポートを使用し、 4 つのノードを持つ単一区画データベース・インスタンスをセットアップすることです。この例では、4 つのサーバーが 2 つの MSCS クラスターに構成されます。相互引き受け構成を使用することにより、マシンのいずれかが障害を起こした場合、そのマシン用に構成されたデータベース区画サーバーは、 MSCS ソフトウェアを使用して構成されたとおり、代替マシンにフェイルオーバーし、その代替マシン上で論理ノードとして実行します。

結果の構成には 2 つの MSCS クラスターがあります。各クラスターには、次のものがあります。

- 2 つのサーバー。それぞれ 64 MB のメモリーおよび 2 GB のローカル SCSI ディスクが 1 つあります。
- それぞれが 2 GB の 3 つの SCSI ディスクを持つ 1 つの SCSI ディスク・タワー。

加えて、各マシンには 1 つの 100X イーサネット・アダプター・カードがインストールされています。

各マシンには次のソフトウェアがインストールされています。

- MSCS 機能を持つ Windows NT Version 4.0 Enterprise Edition
- DB2 ユニバーサル・データベース エンタープライズ拡張エディション バージョン 7

結果のネットワーク構成は次のとおりです。

サーバー 1: • マシン名: db2test1 • TCP/IP ホスト名: db2test1 • IP アドレス: 9.9.9.1 (サブネット・マスク: 255.255.255.0 • MSCS クラスター名: ClusterA	サーバー 2: • マシン名: db2test2 • TCP/IP ホスト名: db2test2 • IP アドレス: 9.9.9.2 (サブネット・マスク: 255.255.255.0 • MSCS クラスター名: ClusterA
サーバー 3: • マシン名: db2test3 • TCP/IP ホスト名: db2test3 • IP アドレス: 9.9.9.3 (サブネット・マスク: 255.255.255.0 • MSCS クラスター名: ClusterB	サーバー 4: • マシン名: db2test4 • TCP/IP ホスト名: db2test4 • IP アドレス: 9.9.9.4 (サブネット・マスク: 255.255.255.0 • MSCS クラスター名: ClusterB

ネットワーク中のすべてのマシンが TCP/IP で構成され、イーサネット 100 T-base Hub を使用して私用の LAN に接続されます。ドメイン・ネーム・サーバー (DNS) はありませんが、すべてのマシンにはローカル TCP/IP `hosts` ファイルがあります。このファイルには以下のエントリーが含まれています。

```
9.9.9.1 db2test1 # for Server 1
9.9.9.2 db2test2 # for Server 2
9.9.9.3 db2test3 # for Server 3
9.9.9.4 db2test4 # for Server 4
9.9.9.5 ClusterA # for MSCS Cluster 1
9.9.9.6 ClusterB # for MSCS Cluster 2
9.9.9.7 db2tcp # for DB2 remote client connection
```

仮のタスク

次のタスクを実行する前に、4 つのすべてのマシンが同じドメイン、 `DB2NTD` に所属していると仮定します。

1. ローカル管理者グループのメンバーである DB2 のドメイン・アカウントを、 DB2 が実行する予定のマシン上で作成します。すべてのタスクを実行するためのアカウントを使用します。
   - ユーザー名を `db2nt` に設定します。
   - パスワードを `db2nt` に設定します。
2. *"password never expires"* 特性で、 2 番目のドメイン・アカウントを作成します。このアカウントは、DB2 サービスと関連付けられます。
   - ユーザー名を `db2mpp` に設定します。
   - パスワードを `db2mpp` に設定します。

3. MSCS 機能をマシン db2test1 および db2test2 にインストールします。
   - MSCS クラスターに ClusterA という名前を付けます。
   - クラスター IP アドレスは 9.9.9.5 です。
   - 共用ディスク D: は MSCS ソフトウェアによって使用されます。
   - 共用ディスク E: および F: は DB2 によって使用されます。
4. MSCS 機能をマシン db2test3 および db2test4 にインストールします。
   - MSCS クラスターに ClusterB という名前を付けます。
   - クラスター IP アドレスは 9.9.9.6 です。
   - 共用ディスク D: は MSCS ソフトウェアによって使用されます。
   - 共用ディスク E: および F: は DB2 によって使用されます。
5. DB2 エンタープライズ拡張エディションをマシン db2test1 にインストールします。
   - *"This machine will be the instance-owning database partition server"* オプションを選択します。
   - DB2 サービスのアカウントは db2mpp です。パスワードは db2mpp です。
   - ローカル・ドライブである C:¥SQLLIB に、ソフトウェアをインストールします。
6. DB2 エンタープライズ拡張エディションをマシン db2test2、db2test3、および db2test4 にインストールします。
   - *"This machine will be a new node on an existing partitioned database system"* オプションを選択します。
   - db2test1 をインスタンス所有のマシンとして選択します。
   - DB2 サービスのアカウントは db2mpp です。パスワードは db2mpp です。
   - ローカル・ドライブである C:¥SQLLIB に、ソフトウェアをインストールします。

次のステップは、DB2MSCS.CFG ファイルをセットアップし、 DB2MSCS ユーティリティーを実行することです。

DB2MSCS ユーティリティーを実行する

db2test1 マシンをセットアップするには、以下のタスクを実行してください。

1. ユーザー db2nt としてログオンします。このユーザーはローカル管理者グループに属しています。パスワードは db2nt です。
2. DB2MSCS.CFG ファイルをセットアップします。

```
#
#  DB2MSCS.CFG for one partitioned database system with
#  multiple MSCS clusters
DB2_INSTANCE=DB2MPP
CLUSTER_NAME=ClusterA
DB2_LOGON_USERNAME=db2mpp
DB2_LOGON_PASSWORD=db2mpp
```

```
# Group 1
# for DB2 node 0
GROUP_NAME=DB2NODE0
DB2_NODE=0
IP_NAME=IP Address for DB2
IP_ADDRESS=9.9.9.7
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
NETNAME_NAME=Network name for DB2
NETNAME_VALUE=DB2WOLF
NETNAME_DEPENDENCY=IP Address for DB2
DISK_NAME=Disk E:
INSTPROF_DISK=Disk E:
#

# Group 2
# for DB2 node 1
GROUP_NAME=DB2NODE1
DB2_NODE=1
DISK_NAME=Disk F:
#

CLUSTER_NAME=ClusterB
# Group 3
# for DB2 node 2
GROUP_NAME=DB2NODE2
DB2_NODE=2
DISK_NAME=Disk E:

#
# Group 4
# for DB2 node 3
GROUP_NAME=DB2NODE3
DB2_NODE=3
DISK_NAME=Disk F:
```

3. 以下のように DB2MSCS ユーティリティーを実行してください。

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

4. db2nt アカウントからログアウトします。

最終ステップは 2 つの MSCS クラスターに、データベース・ドライブ・マッピングを登録することです。

ClusterA にデータベース・ドライブ・マッピングを登録する

MSCS クラスター ClusterA にデータベース・ドライブ・マッピングを登録するには、以下のタスクを実行します。

1. マシン db2test1 では、ユーザー db2mpp としてログオンします。これは DB2 サービスと関連付けられるアカウントです。パスワードは db2mpp です。
2. データベース・ドライブ・マッピングを登録するには、次のコマンドを入力します。

```
db2drvmp add 0 F E

db2drvmp add 1 E F
```

3. すべての DB2 リソースをオフラインにしてからオンラインにします。

ClusterB にデータベース・ドライブ・マッピングを登録する

MSCS クラスター ClusterB にデータベース・ドライブ・マッピングを登録するには、以下のタスクを実行します。

1. マシン db2test3 では、ユーザー db2mpp としてログオンします。これは DB2 サービスと関連付けられるアカウントです。パスワードは db2mpp です。
2. データベース・ドライブ・マッピングを登録するには、次のコマンドを入力します。

```
db2drvmp add 2 F E
db2drvmp add 3 E F
```

3. すべての DB2 リソースをオフラインにしてからオンラインにします。

MSCS 環境で DB2 を管理する

MSCS クラスターを使用している場合、DB2 インスタンスには毎日の操作、データベースの展開、およびデータベース構成に関係する付加的な計画を必要とします。 DB2 を MSCS ノードで透過的に実行するには、付加的な管理用タスクを実行する必要があります。すべての DB2 従属のオペレーティング・システム・リソースは、すべての MSCS ノードで使用可能でなければなりません。これらのオペレーティング・システムのいくつかは、MSCS の効力範囲外です。つまり、MSCS リソースとして定義できません。同じオペレーティング・システム・リソースがすべての MSCS ノードで使用可能であるように、各システムを確実に構成することが必要です。以下のセクションでは、行う必要がある付加的な作業を説明します。

DB2 リソースの開始および停止

DB2 リソースは、 Cluster Administrator ツールから開始したり停止したりするべきです。いくつかのメカニズムは **db2start** コマンド、および「コントロール パネル」からの**「サービス」**オプションなどの DB2 インスタンスを開始するのに使用できます。しかし、DB2 が Cluster Administrator から開始しない場合、 MSCS ソフトウェアは DB2 インスタンスの状態を感知しません。 DB2 インスタンスが Cluster Administrator を使用して開始され、 **db2stop** コマンドを使用して停止される場合、 MSCS ソフトウェアは **db2stop** コマンドをソフトウェア障害として解釈し、 DB2 を再始動しようとします。 (現在の MSCS インターフェースは、*リソース状態* の通知をサポートしません。)

同様に、**db2start** を使用して DB2 インスタンスを開始する場合、 MSCS はリソースがオンライン上にあるかどうか検出できません。データベース・サーバーが失敗すると、 MSCS は DB2 リソースをクラスター中のフェイルオーバー・マシン上でオンラインにすることはありません。

3 つの操作が DB2 インスタンスに適用します。

オンライン
この操作は、**db2start** コマンドの使用と同じです。 DB2 がすでに活動状態である場合、この操作を使用して、 MSCS に DB2 が活動状態であることを単に通知することができます。この操作中にエラーが起きると、Windows NT イベント・ログに書き込まれます。

オフライン
この操作は、**db2stop** コマンドの使用と同じです。インスタンスに活動状態の接続がある場合、この操作は失敗します。これは **db2stop** の動作と一貫します。

リソースの失敗
この操作は、**force** オプションを指定した **db2stop** コマンドの使用と同じです。 DB2 はすべてのアプリケーションを DB2 システムから切断し、すべてのデータベース・サーバーを停止します。

スクリプトの実行

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。これらのスクリプトは、DB2INSTPROF 環境変数に指定される、インスタンス・プロファイル・ディレクトリーの中になければなりません。このディレクトリーは、 **db2icrt** コマンドの -p パラメーターによって指定されるディレクトリー・パスです。次のコマンドを発行して、この値を得ることができます。

```
db2set -i:instance_name DB2INSTPROF
```

このファイル・パスは、インスタンス・ディレクトリーがすべてのクラスター・ノードで使用可能になるように、クラスター化ディスク上になければなりません。

これらのスクリプト・ファイルは必須ではなく、インスタンス・ディレクトリーで検出される場合にのみ実行されます。これらはバックグラウンドで MSCS クラスター・サービスによって立ち上げられます。スクリプト・ファイルは、スクリプト・ファイル内のコマンドから戻される情報を取り込むために標準出力をリダイレクトする必要があります。出力は画面には表示されません。

区分データベース環境では、デフォルトで、インスタンス中のすべてのデータベース区画サーバーが同じスクリプトを使用します。インスタンス中の異なるデータベース区画サーバーを見分ける必要がある場合、 DB2NODE 環境変数の異なる割り当てをターゲット・ノード番号に使用してください (たとえば、 `db2cpre.bat` および `db2cpost.bat` ファイルで IF ステートメントを使用する)。

DB2 リソースをオンラインにする前にスクリプトを実行する

DB2 リソースをオンラインにする*前に* スクリプトを実行したい場合、スクリプトの名前を `db2cpre.bat` にしなければなりません。 DB2 は Windows NT コマンド行プロセッサー (CLP) からこのバッチ・ファイルを立ち上げる関数を呼び出し、 DB2 リソース

がオンラインになる前に CLP が実行を完了するのを待機します。このバッチ・ファイルは、 DB2 データベース・マネージャー構成の修正などのタスクをこのバッチ・ファイルに使用することができます。フェイルオーバー・システムが無理である場合、データベース・マネージャーのパラメーター値をいくつか変更する必要があるかもしれませんし、 DB2 が使用するシステム・リソースを削減することが必要です。

db2cpre.bat スクリプトに配置されたコマンドは、同期で実行されるべきです。そうしないと、DB2 リソースは、スクリプト中のすべてのタスクが完了する前にオンラインになり、その結果予期しない動作が起きる場合があります。特に、**db2cmd** を db2cpre.bat スクリプトで呼び出すべきではありません。これは、 DB2 コマンドを非同期に **db2cmd** プログラムに対して実行する他のコマンド・プロセッサーを代わりに立ち上げてしまうからです。

db2cpre.bat スクリプトで DB2 CLP コマンドを使用したい場合、コマンドはファイル中に配置し、 DB2 コマンド行プロセッサー用に DB2 環境を初期化するプログラム内から CLP バッチ・ファイルとして実行する必要があり、次に DB2 コマンド行プロセッサーの完了を待機します。次に例を示します。

```
#include <windows.h>

int WINAPI DB2SetCLPEnv_api(DWORD pid);

void main ( int argc, char *argv [ ] )
{
      STARTUPINFO         startInfo   = {0};
      PROCESS_INFORMATION pidInfo     = {0};
      char     title  [32]   = "Run Synchronously";
      char     runCmd [64]  =
                              "DB2 -z c:¥¥run.out -tvf c:¥¥run.clp";

/* Invoke API to set up a CLP Environment */
      if ( DB2SetCLPEnv_api (GetCurrentProcessId ()) == 0 ) 1
      {
         startInfo.cb          = sizeof(STARTUPINFO);
         startInfo.lpReserved  = NULL;
         startInfo.lpTitle     = title;
         startInfo.lpDesktop   = NULL;
         startInfo.dwX         = 0;
         startInfo.dwY         = 0;
         startInfo.dwXSize     = 0;
         startInfo.dwYSize     = 0;
         startInfo.dwFlags     = 0L;
         startInfo.wShowWindow = SW_HIDE;
         startInfo.lpReserved2 = NULL;
         startInfo.cbReserved2 = 0;
               if ( CreateProcessA( NULL,
                              runCmd, 2
                              NULL,
                              NULL,
                              FALSE,
                              NORMAL_PRIORITY_CLASS CREATE_NEW_CONSOLE,
```

```
                              NULL,
                              NULL,
                              &startInfo,
                              &pidInfo ) )
            {
               WaitForSingleObject (pidInfo.hProcess, INFINITE);
               CloseHandle (pidInfo.hProcess);
               CloseHandle (pidInfo.hThread);
            }
         }
         return;
}
```

1 API DB2SetCLPEnv_api はインポート・ライブラリー DB2API.LIB によって解決されます。この API は、呼び出される CLP コマンドを許可する環境を設定します。このプログラムが db2cpre.bat スクリプトから呼び出される場合、コマンド・プロセッサーは CLP コマンドの完了を待機します。

2 runCmd は DB2 CLP コマンドを含むスクリプト・ファイルの名前です。

db2clpex.exe という名前のサンプル・プログラムが、 DB2 インストール・パスの MISC サブディレクトリーに見つかることがあります。この実行可能ファイルは提供される例に類似していますが、 DB2 CLP コマンドをコマンド行引き数として受け入れます。このサンプル・プログラムを使用したい場合、 BIN サブディレクトリーにコピーしてください。この実行可能ファイルは、 db2cpre.bat スクリプトで次のように使用できます (INSTHOME はインスタンス・ディレクトリー)。

```
db2clpex "DB2 -Z INSTHOME¥pre.log -tvf INSTHOME¥pre.clp"
```

すべての DB2 ATTACH コマンドまたは CONNECT ステートメントは、明示的にユーザーを指定する必要があります。そうしないと、クラスター・サービスと関連するユーザー・アカウントのもとで実行されます。 CLP スクリプトはまた TERMINATE コマンドで完了して、 CLP バックグラウンド・プロセスを終了する必要があります。

以下に、db2cpre.bat ファイルの例を示します。

```
db2cpre.bat : 1
------------------------
db2clpex "db2 -z INSTHOME¥pre-%DB2NODE%.log -tvf INSTHOME¥pre.clp" 2 - 5
------------------------

PRE.CLP 6
------------------------
update dbm cfg using MAXAGENTS 200;
get dbm cfg;
terminate;
------------------------
```

1 db2cpre.bat スクリプトは、クラスター・サービスと関連するユーザー・アカ

ウントの下で実行します。 DB2 処置が必要な場合、クラスター・サービスと関連するユーザー・アカウントは、 DB2 で定義されているように有効な SQL ID でなければなりません。

2 INSTHOME はインスタンス・ディレクトリーです。

3 ログ・ファイルの名前は、両方の論理ノードが同時にオンラインになったときにファイルの競合を避けるため、各ノードごとに異なっていなければなりません。

4 db2clpex.exe は、コマンド行引き数を使用して、呼び出す CLP コマンドを指定するサンプル・プログラムです。

5 db2clpex.exe サンプル・プログラムは、すべての MSCS クラスター・ノードで使用可能にされなければなりません。

6 この例の CLP コマンドは、エージェントの数に制限を設定します。

DB2 リソースをオンラインにした後にスクリプトを実行する

DB2 リソースをオンラインにした後に スクリプトを実行したい場合、スクリプトの名前を db2cpost.bat にしなければなりません。スクリプトは、DB2 リソースが正常にオンラインになってから、 MSCS から非同期で実行されます。 **db2cmd** コマンドをこのスクリプトで使用し、 DB2 CLP スクリプト・ファイルを実行できます。 **db2cmd** コマンドの -c パラメーターを使用して、ユーティリティーがタスクの完了時にすべてのウィンドウをクローズするように指定します。次に例を示します。

```
db2cmd -c db2 -tvf mycmds.clp
```

-c パラメーターは、 **db2cmd** コマンドに対する最初の引き数でなければなりません。これにより、バックグラウンドのコマンド・プロセッサーを孤立しないようにできます。

db2cpost.bat スクリプトは、 DB2 リソースがフェイルオーバーし、活動状態になった直後にデータベース活動を実行したい場合に役立ちます。たとえば、ユーザー・アクセス用にプライム状態になるように、インスタンス中のデータベースを再始動または活動化することができます。

以下に、db2cpost.bat スクリプトの例を示します。

```
db2cpost.bat 1
------------------------
db2cmd -c db2 -z INSTHOME¥post-%DB2NODE%.log -tvf INSTHOME¥post.clp 2 - 4
------------------------

POST.CLP 5
------------------------
restart database SAMPLE;
connect reset;
activate database SAMPLE;
terminate;
------------------------
```

1. db2cpost.bat スクリプトは、クラスター・サービスと関連するユーザー・アカウントの下で実行します。 DB2 処置が必要な場合、クラスター・サービスと関連するユーザー・アカウントは、 DB2 で定義されているように有効な SQL ID でなければなりません。
2. INSTHOME はインスタンス・ディレクトリーです。
3. ログ・ファイルの名前は、両方の論理ノードが同時にオンラインになったときにファイルの競合を避けるため、各ノードごとに異なっていなければなりません。
4. **db2cmd** コマンドを使用できます。 db2cpost.bat スクリプトが非同期に実行できるためです。 -c パラメーターを使用して、コマンド・プロセッサーを終了する必要があります。
5. この例の CLP スクリプトには、データベースを再始動および活動化させるコマンドが含まれています。このスクリプトは、データベース・マネージャーの起動直後、データベースを活動状態に戻します。区分データベース・システムでは、複数の DB2 リソースが同時にオンラインになるので、 ACTIVATE DATABASE コマンドを除去する必要があります。 RESTART DATABASE コマンドは、別のノードがデータベースを活動化するために失敗することがあります。これが起きた場合、スクリプトを再実行してデータベースが確実に、正しく再始動するようにしてください。

データベースの考慮事項

データベースの作成時、データベース・パスが共用ディスクを必ず参照するようにしてください。これによって、データベースがすべての MSCS ノードで参照可能になります。すべてのログと他のデータベース・ファイルも、DB2 が正常にフェイルオーバーするように、クラスター・ディスクを参照することが必要です。これらのステップを実行しないと、ファイルが削除されたまたは使用不可能になった DB2 を参照するので、DB2 システム障害が発生します。

また、データベース・マネージャーおよびデータベース構成パラメーターの設定が、必ず DB2 の使用するシステム・リソース量がどの MSCS ノードでもサポートされるようにしてください。 *autorestart* データベース構成パラメーターは、フェイルオーバー上の最初のデータベース接続がデータベースを一貫した状態にするように、 ON に設定するべきです。 (*autorestart* のデフォルト設定は ON です。) また、 db2cpost.bat スクリプトを使用してデータベースを再始動および活動化することによっても、データベースを作動可能状態にすることができます。この方法をお勧めします。*autorestart* には従属関係がなく、データベースはユーザー接続要求とは関係なく作動可能状態になるためです。

ユーザーおよびグループ・サポート

DB2 はユーザー認証およびグループ・サポートに関して、 Windows NT に依存しています。 DB2 インスタンスをシームレス (直接) に 1 つの MSCS ノードから別のノー

ドにフェイルオーバーするには、各 MSCS ノードを同じ Windows NT セキュリティー・データベースにアクセスさせる必要があります。これは、Windows NT ドメイン・セキュリティーを使用して達成できます。

すべての DB2 ユーザーとグループをドメイン・セキュリティー・データベースで定義してください。 MSCS ノードは、このドメインのメンバーでなければならないか、またはドメインがトラステッド・ドメインであることが必要です。その場合、DB2 はドメイン・セキュリティー・データベースを、 DB2 を実行しているノードに関係なく認証およびグループ・サポートに使用します。

ローカル・アカウントを使用している場合、アカウントは各 MSCS ノードで複製される必要があります。このアプローチは、エラーを起こしやすく、二重のメンテナンスが必要なので、あまりお勧めできません。

すべての MSCS ノードが同じ DCE セルのクライアントである場合、 DCE セキュリティーもサポートされる認証モードです。

MSCS サービスを、DB2 命名規則に従うユーザー・アカウントと関連付ける必要があります。これによって MSCS サービスは、DB2 に対して処置をとることができます。このことは `db2cpre.bat` および `db2cpost.bat` スクリプトで必要になることがあります。

Windows NT ユーザーおよびグループ・サポートについての詳細は、 *管理の手引き: インプリメンテーション* の『DB2 (Windows NT 版) での DB2 ユーザー認証』を参照してください。

通信に関する考慮事項

DB2 は、MSCS 環境で 2 つの LAN プロトコルをサポートします。

- TCP/IP
- NetBIOS

TCP/IP は、クラスター・リソース・タイプであるためサポートされます。 DB2 が区分データベース・システムの通信プロトコルとして TCP/IP を使用できるようにするには、 IP アドレス・リソースを作成して、リモート・アプリケーションに調整プログラム・ノードとして使用したいデータベース区画サーバーを表す DB2 リソースとして同じグループに配置してください。次に Cluster Administrator ツールを使用して従属関係を作成し、 DB2 リソースが開始される前に確実に IP リソースがオンラインになるようにします。これで DB2 クライアントは TCP/IP ノード・ディレクトリー・エントリーのカタログを作成し、この TCP/IP アドレスを使用できます。

svcename データベース・マネージャー構成パラメーターと関連する TCP/IP ポートは、インスタンスに参加するすべてのマシン上で DB2 インスタンスが使用するために予約

しておく必要があります。ポート番号と関連したサービス名も、すべてのマシン上の services ファイルで同じでなければなりません。

NetBIOS はクラスター・リソースによってサポートされませんが、 NetBIOS を LAN プロトコルとして使用できます。これはプロトコルが LAN 上で NetBIOS 名を確実に固有のものにするからです。 DB2 が NetBIOS 名を登録すると、NetBIOS は名前が LAN で使用されていないことを確かめます。フェイルオーバーのシナリオでは、DB2 がシステムからシステムへと移動されると、 DB2 が使用する *nname* は、 MSCS クラスターで 1 つのパートナー・マシンから登録解除され、別のマシンに登録されます。

DB2 NetBIOS サポートは NetBIOS Frames (NBF) を使用します。このプロトコル・スタックは、異なる論理アダプター番号 (LANA) と関連付けることができます。サーバーに一貫した NetBIOS アクセスを保証するために、 NBF プロトコル・スタックと関連した LANA はすべてのクラスター・ノードで同じでなければなりません。これは、「コントロール パネル」の**「ネットワーク」**オプションを使用して構成できます。NBF を LANA 0 と関連付けることは DB2 のデフォルト設定で予期されているので、そうする必要があります。

システム時刻の考慮事項

DB2 はシステム時刻を使用して特定の操作にタイム・スタンプを付けます。 DB2 フェイルオーバーに参加するすべての MSCS ノードには、システム時刻ゾーン、および DB2 がすべてのマシンで一貫した動作をするように同期化されたシステム時刻が必要です。

「コントロール パネル」ダイアログの**「日付と時刻」**オプションを使用してシステム時刻を設定します。 MSCS には、 MSCS ノードがクラスターを形式化するために結合する時に日時を同期化するタイム・サービスがあります。しかし、タイム・サービスは単に 12 時間ごとに時刻を同期化するだけなので、時刻が 1 つのシステム上で変更され、時刻が同期化される前に DB2 がフェイルオーバーする場合に問題が起きることがあります。

MSCS クラスター・ノードの 1 つで時刻が変更される場合、次のコマンドを使用して他のクラスター・ノードで手作業で同期化されなければなりません。

```
net time /set /y ¥¥remote_node
```

remote_node には、クラスター・ノードのマシン名が入ります。

区分データベース環境の管理サーバーとコントロール・センターに関する考慮事項

DB2 管理サーバーは、 DB2 ユニバーサル・データベースのインストール中に (オプションで) 作成されます。これは区分データベース・システムではありません。コントロール・センターは、管理サーバーが提供するサービスを使用して、 DB2 インスタンスおよびデータベースを管理します。

区分データベース・システムでは、 1 つの DB2 インスタンスが複数の MSCS ノードに常駐できます。これは、どの MSCS ノードで DB2 インスタンスが活動状態になっているかに関係なく、インスタンスがずっとアクセス可能になっているように、コントロール・センターの下の複数のシステム上で DB2 インスタンスのカタログが作成されなければならないということを暗示しています。

管理サーバーのインスタンス・ディレクトリーは共用ではありません。管理サーバーのディレクトリーのすべてのユーザー定義のファイルを、すべての MSCS ノードに対して二重化して、すべての MSCS ノードに同じレベルの管理を提供しなければなりません。特に、ユーザー・スクリプトおよびスケジュールされた実行可能ファイルを、すべてのノードで使用可能にしなければなりません。また、スケジュールされた活動を MSCS クラスターのすべてのマシン上でもスケジュールしなければなりません。

別の方法として、すべてのマシン上で管理サーバーを重複する代わりに、管理サーバーをフェイルオーバーさせることも可能です。次の例の目的では、クラスター中に 2 つの MSCS ノードがあり、 MACH0 および MACH1 という名前だと仮定します。 MACH0 は、管理サーバーが使用するクラスター・ディスクにアクセスできます。さらに、MACH0 および MACH1 両方に管理サーバーがあるとします。以下のステップを実行して、管理サーバーを高度に使用可能にします。

1. 各マシンで **db2admin stop** コマンドを呼び出し、両方のマシンで管理サーバーを停止します。
2. すべての管理クライアント・マシンで、 UNCATALOG NODE コマンドを使用し、MACH0 および MACH1 で管理サーバーへのすべての参照のカタログ化を解除します。 (LIST NODE DIRECTORY コマンドをクライアント・マシン上で使用して、管理サーバーへの参照が残っていないか判別できます。)
3. MACH1 から **db2admin drop** コマンドを呼び出し、 MACH1 から管理サーバーを除去します。 (このステップは、両方のマシン上に管理サーバーがある場合にのみ実行します。)
4. MACH0 から **db2admin** コマンドを発行し、管理サーバーの名前を決定します。(デフォルトの名前は `DB2DAS00` です。)
5. DB2MSCS ユーティリティーを使用して管理サーバーにフェイルオーバー・サポートをセットアップします。これには、IP およびディスク・リソースで従属関係を持つ `DB2DAS00` という名前の MSCS で DB2 リソースを作成することが必要です。(相互引き受け構成がある場合、 NODE0 のための DB2 リソースを保持するグループにリソースを入れます。) このリソースは、管理サーバーをサポートする MSCS リソースとして使用されます。 `DB2MSCS.ADMIN` ファイルは次のとおりです。

```
#
# db2mscs.admin for Administration Server
# run db2mscs -f:db2mscs.admin
#
DB2_INSTANCE=DB2DAS00
CLUSTER_NAME=CLUSTERA
DB2_LOGON_USERNAME=db2admin
```

```
DB2_LOGON_PASSWORD=db2admin
# put Administration server in the same group as DB2 Node 0
GROUP_NAME=DB2NODE0 1
DISK_NAME=DISK E:
INSTPROF_DISK=DISK E:
IP_NAME= IP Address for Administration Server
IP_ADDRESS=9.9.9.8
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
```

1 このグループは、既存のグループと同じものかもしれません。このように、インスタンス・プロファイル・ディレクトリーには付加的なディスクは必要ありません。

6. MACH1 で、次のコマンドを呼び出し、DB2DAS00 を管理サーバーとして設定します。

   ```
   db2set -g db2adminserver=DB2DAS00
   ```

7. MACH0 で、サービス・プログラムを介して DB2DAS00 の開始プロパティーを変更し、自動でなく手作業で立ち上げることができるようにします。これは、DB2DAS00 がこの時点で MSCS に制御されているためです。

管理サーバーがフェイルオーバー用に使用可能になると、すべてのリモート・アクセスは管理サーバーとの通信に MSCS IP リソースを使用しなければなりません。管理サーバーは、これで次の特性を持つことになります。

- 管理サーバーのインスタンス・ディレクトリーは管理サーバーでフェイルオーバーします。
- クライアントは、活動状態の MSCS ノードに関係なく、単一ノードのカタログを作成して管理サーバーと通信するだけですみます。
- 管理サーバーに対するジョブのスケジュールは 1 度だけですみます。
- ローカル・インスタンスは、管理サーバーがローカル・インスタンスと同じ MSCS ノードで活動状態のときにだけ管理サーバーによって制御されます。
- 管理サーバーは、クラスター・サービスが活動状態でない場合はアクセスできません。

制限と制約事項

MSCS 環境で DB2 を実行するには、以下の制限があります。

- 共用ディスクに、両方の MSCS ノードで通用する同じ物理ディスク番号がない限り、共用ディスクで物理入出力を使用することはできません。ディスクは区画 ID を使用してアクセスされるので、論理入出力は使用できます。
- MSCS サポートにすべての DB2 リソースを構成することが必要です。そうしないと、 DB2 実行時にシステム・エラーが発生します (DB2 はシステム・リソースがないと適切に操作しません)。たとえば、データベース・ログが MSCS 共用ディスク上にないと、 DB2 はデータベースを再始動できません。

- Cluster Administrator ツールから DB2 インスタンスを管理する必要があります。MSCS はデータベース・マネージャーをソフトウェア矛盾として開始、または停止するのに使用される他のメカニズムを表示します。たとえば、DB2 を開始するのに MSCS を、停止するのに **db2stop** コマンドを使用すると、 MSCS はこれをソフトウェア障害として検出し、インスタンスを再始動します。これは、コントロール・センターを使用して DB2 を開始および停止してはならないということも示唆しています。
- DB2 をアンインストールするには、まず MSCS を停止する必要があります。

第8章 Solaris 実行環境での高可用性

Solaris 実行環境での高可用性は、DB2 が Sun Cluster 2.x (SC2.x)、Sun Cluster 3.0 (SC3.0)、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 3.0 についての情報は、 『DB2 and High Availability on Sun Cluster 3.0』 というタイトルのホワイトペーパー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 についての情報は、 『DB2 and High Availability on VERITAS Cluster Server』 というタイトルのホワイトペーパーを参照してください。これも 『DB2 UDB and DB2 Connect Online Support』 Web サイトから入手できます。

この章では、 DB2 が Sun Cluster 2.x (SC2.x) と協働して高可用性を実現する方法について詳しく説明しています。また、これらの 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31 を参照してください)。

図 31. DB2、Sun Cluster 2.x、および高可用性

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 万時間の次数 (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、図32 で図示されています。

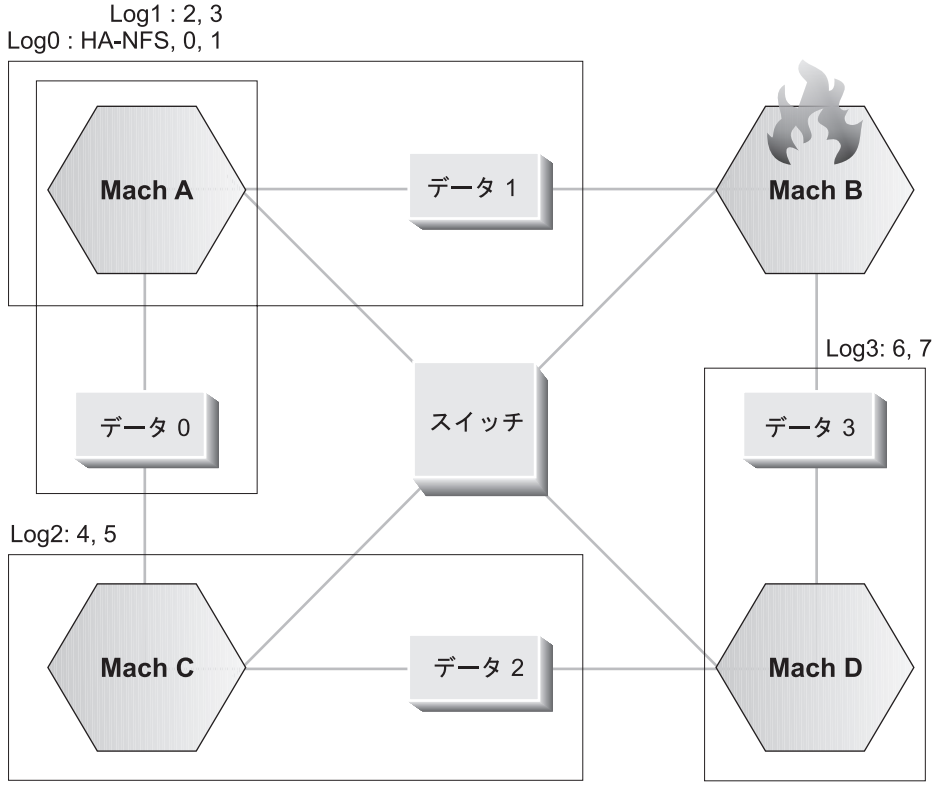


図 32. フェイルオーバー

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・サービスを回復できることと、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。

高可用性 Web サーバーにアクセスする Web ブラウザーを考慮してみましょう。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す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、その理由を付きとめ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作業負荷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) を検査し、問題が起きる前に置き換えます。

フォールト・トレランスおよび連続可用性

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、隙間産業で最も良く使用され、通常、このマシンを実装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、回復することができるという利点があります。

連続可用性 は、高可用性の上位ステップです。これにより、計画済みおよび非計画のダウン時間からクライアントが保護されます。連続可用性が構成されていると、データ・サービスをホストするマシンの 1 つで障害が起きたり、または保守のためにダウンしたりした場合に、クライアントはまったく影響を受けません。連続可用性の構成は複雑で、実装にさらに費用がかかります。

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、そして比較的実装に費用がかからないからです。

Sun Cluster 2.2

Sun Cluster 2.2 (SC2.2) は、 Sun Microsystems の高可用性クラスター化製品です。SC2.2 は単一のクラスターで最大 4 台のマシンをサポートします。 4 台の Ultra Enterprise 10000 を使用すると、クラスターは最大で 256 個の CPU と 256 GB の RAM を持つことができます。

サポートされるシステム

システム	UltraSPARC	メモリー容量	入出力
Ultra Enterprise 1	1	64MB ～ 1GB	3 SBus
Ultra Enterprise 2	1 ～ 2	64MB ～ 2GB	4 SBus
Ultra Enterprise 450	1 ～ 4	32MB ～ 4GB	10 PCI
Ultra Enterprise 3000	1 ～ 6	64MB ～ 6GB	9 SBus
Ultra Enterprise 4000	1 ～ 14	64MB ～ 14GB	21 SBus
Ultra Enterprise 5000	1 ～ 14	64MB ～ 14GB	21 SBus
Ultra Enterprise 6000	1 ～ 30	64MB ～ 30GB	45 SBus
Ultra Enterprise 10000	1 ～ 64	512MB ～ 64GB	64 SBus

エージェント

Sun Cluster ソフトウェアには、 SC2.2 製品に付属してサポートされる、多数の高可用性エージェントが含まれています。他の HA エージェント (たとえば、DB2 のエージェント) は、Sun の外部で開発されており、 Sun Cluster ソフトウェアには付属していません。 DB2 の HA エージェントは DB2 に付属しており、 IBM によってサポートされ、DB2 に付属して無料で提供されています。

Sun Cluster ソフトウェアは、 Sun Cluster ソフトウェアの各種コンポーネントに対応するメソッド (スクリプトまたはプログラム) を登録する機会を提供することにより、高可用性データ・サービスを処理します。これらのメソッドを使用して、 SC2.2 ソフトウェアでは、詳しい知識がなくてもデータ・サービスを制御できます。このメソッドには、以下のものがあります。

START 論理ネットワーク・インターフェースがオンラインになる前に、データ・サービスの一部を開始します。

START_NET
論理ネットワーク・インターフェースがオンラインになった後に、データ・サービスの一部を開始します。

STOP 論理ネットワーク・インターフェースがオフラインになった後に、データ・サービスの一部を停止します。

STOP_NET
論理ネットワーク・インターフェースがオフラインになる前に、データ・サービスの一部を停止します。

ABORT
STOP メソッドに似ていますが、クラスター・ソフトウェアによりマシンがダウン状態になる直前に実行されるという点が異なります。この場合、マシンの「健康」が重要であるため、データ・サービスでは、マシンがダウン状態になる前に「最後のお願い」要求を実行することができます。このメソッドは、論理ネットワーク・インターフェースがオフラインになった後に実行されます。

ABORT_NET
ABORT メソッドに似ていますが、論理ネットワーク・インターフェースがオフラインになる前に実行されるという点が異なります。

FM_INIT
障害モニターを初期化します。

FM_START
障害モニターを開始します。

FM_STOP
障害モニターを停止します。

FM_CHECK

hactl コマンドにより呼び出されます。対応するデータ・サービスの現行の状態を戻します。

DB2 エージェントは、スクリプト START_NET、STOP_NET、FM_START、および FM_STOP で構成されています。スクリプト ABORT、ABORT_NET、および FM_CHECK はクラスターの再構成時に実行されません。

高可用性エージェントは、これらのメソッドの 1 つまたは複数で構成されます。メソッドは、 **hareg** コマンドにより SC2.2 に登録されます。メソッドが登録されると、 Sun Cluster は対応するメソッドを呼び出して、データ・サービスを制御します。

サービスの ABORT および STOP メソッドは呼び出されない場合があることを覚えておくことは大切です。これらのメソッドはデータ・サービスの制御シャットダウンを目的としており、データ・サービスは、これらのメソッドを呼び出さずにマシンに障害が起きても回復できなければなりません。

詳細については、 Sun Cluster の資料を参照してください。

論理ホスト

SC2.2 ソフトウェアは、論理ホストの概念を使用します。*論理ホスト* は、一連のディスクと 1 つ以上の論理公衆ネットワーク・インターフェースで構成されます。高可用性データ・サービスは論理ホストと関連しており、論理ホストのディスク・グループにあるディスクを必要とします。論理ホストは、クラスター内のさまざまなマシンによってホスト可能で、実行されているマシンの CPU およびメモリーを「借りる」ことができます。

論理ネットワーク・インターフェース

他の UNIX ベースのオペレーティング・システムと同様、 Solaris にも、ネットワーク・インターフェースの 1 次 IP アドレスに加えて、追加 IP アドレスを持つ機能があります。追加 IP アドレスは、 1 次 IP アドレスが物理ネットワーク・インターフェースに常駐するのと同じ方法で、論理インターフェースに常駐します。以下は、クラスター内の 2 つのマシン上にある論理インターフェースの例です。 2 つのホストがあり、両方とも現在、マシン「thrash」上にあります。

```
scadmin@crackle(202)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 289966 0 289966 0 0 0
hme0 1500 9.21.55.0 9.21.55.98 121657 6098 764122 0 0 0
scid0 16321 204.152.65.0 204.152.65.1 489307 0 476479 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.33 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.17 347317 0 348073 0 0 0
```

1. lo0 はループバック・インターフェースです。
2. hme0 は公衆ネットワーク・インターフェース (イーサネット) です。
3. scid0 は最初の私設ネットワーク・インターフェース (SCI

(スケーラブル・コヒーレント・インターフェース)) です。
4. scid0:1 は Sun Cluster ソフトウェアが内部で使用する
論理ネットワーク・インターフェースです。
5. scid1 は 2 番目の私設ネットワーク・インターフェースです。

```
scadmin@thrash(203)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 1128780 0 118780 0 0 0
hme0 1500 9.21.55.0 9.21.55.92 1741422 5692 757127 0 0 0
hme0:1 1500 9.21.55.0 9.21.55.109 0 0 0 0 0 0
hme0:2 1500 9.21.55.0 9.21.55.110 0 0 0 0 0 0
scid0 16321 204.152.65.0 204.152.65.2 476641 0 489476 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.34 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.18 348199 0 347444 0 0 0
```

1. hme0:1 は論理ホストの論理ネットワーク・インターフェースです。
2. hme0:2 は別の論理ホストの論理ネットワーク・インターフェースです。

論理ホストには、 1 つまたは複数の論理インターフェースが関連付けられています。これらの論理インターフェースは、論理ホストとともにマシン間を移動し、論理ホストと関連したデータ・サービスにアクセスするために使用されます。これらの論理インターフェースは論理ホストとともに移動するので、クライアントは、データ・サービスが常駐するマシンとは無関係に、そのデータ・サービスにアクセスすることができます。

高可用性データ・サービスは、 TCP/IP アドレス INADDR_ANY にバインドされます。これにより、システム上の各 IP アドレスは、データ・サービスのための接続を確実に受け取ることができます。代わりに、データ・サービスが特定の IP アドレスにバインドされる場合、データ・サービスをホストしている論理ホストに関連した論理インターフェースをバインドしなければなりません。 INADDR_ANY にバインドしておけば、データ・サービスの必要とする新規 IP アドレスがシステムに到着した場合に、その IP アドレスに再バインドする必要がありません。

注: HA インスタンスのクライアントは、論理ホストの論理 IP アドレスのホスト名を使用して、データベースのカタログを作成します。ここではマシンの 1 次ホスト名は使用すべきではありません。 DB2 がそのマシン上で実行されているという保証はないからです。

ディスク・グループとファイル・システム

データ・サービス用のディスクは、グループ単位 (またはセット単位) で論理ホストと関連付けられます。クラスターが Sun StorEdge Volume Manager (Veritas) を実行中の場合、 Sun Cluster ソフトウェアは、 Veritas の「vxdg」ユーティリティーを使用して、各論理ホストのディスク・グループをインポートおよびデポート (追放) します。以下に示すのは、2 つの論理ホスト「log0」および「log1」のディスク・グループの例です。これらの論理ホストは「thrash」というマシンによってホストされています。「crackle」というマシンは、現在どの論理ホストのホストにもなっていません。

```
scadmin@crackle(206)# vxdg list
NAME STATE ID
rootdg enabled 899825206.1025.crackle

scadmin@thrash(205)# vxdg list
NAME STATE ID
rootdg enabled 924176206.1025.thrash
data0 enabled 925142028.1157.crackle=
data1 enabled 899826248.1108.crackle
```

ディスク・グループ「data0」および「data1」はそれぞれ、論理ホスト「log0」および「log1」と対応します。ディスク・グループ「data0」は、以下のコマンドを実行することによって「thrash」からデポートすることができます。

```
vxdg deport data0
```

また、以下のコマンドを実行して「crackle」にインポートできます。

```
vxdg import data1
```

これは Sun Cluster ソフトウェアによって自動的に行われるもので、活動状態のクラスター上で手作業で行ってはなりません。

それぞれのディスク・グループには、クラスター内の 2 つ以上のマシン間で共有可能な多数のディスクが含まれています。論理ホストは、自身が属するディスク・グループ内のディスクに物理アクセス可能なマシンにのみ移動できます。

各論理ホストのファイル・システムを制御する 2 つのファイルがあります。

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logical_host>
/etc/opt/SUNWcluster/conf/hanfs/dfstab.<logical_host>
```

logical_host は、関連する論理ホスト名です。

`vfstab` ファイルは `/etc/vfstab` ファイルと類似していますが、ディスク・グループが論理ホスト用にインポートされた後でマウントされるファイル・システムの項目を含むという点が異なります。 `dfstab` ファイルは `/etc/dfs/dfstab` ファイルと類似していますが、 HA-NFS を介して論理ホスト用にエクスポートされるファイル・システムの項目を含むという点が異なります。各マシンにはこれらのファイルの独自コピーがあるため、クラスター内の各マシンで同じ内容になるように注意を払う必要があります。

注: 論理ホストの `vfstab` ファイルおよび `dfstab` ファイルのパスは、 `hanfs` ディレクトリーが含まれているため、間違いやすくなっています。論理ホストの `dfstab` ファイルだけが HA-NFS で使用されます。 `vfstab` ファイルは、 HA-NFS が構成されていなくても使用されます。

以下に示すのは、相互引き受け構成の DB2 ユニバーサル・データベース エンタープライズ拡張エディション (EEE) を実行しているクラスターの例です。

```
scadmin@thrash(217)# ls -l /etc/opt/SUNWcluster/conf/hanfs
total 8
-rw-r--r-- 1 root build 173 Apr 14 15:01 dfstab.log0
-rw-r--r-- 1 root build 316 Apr 26 12:07 vfstab.log0
-rw-r--r-- 1 root build 389 Apr 13 21:04 vfstab.log1

scadmin@thrash(218)# cat dfstab.log0
share -F nfs -o root=crackle:thrash:¥
jolt:bump:crackle.torolab.ibm.com:thrash.torolab.ibm.com:¥
jolt.torolab.ibm.com:bump.torolab.ibm.com /log0/home
```

ファイル・システム /log0/home をマウントする許可が与えられているホストは、クラスター内の各マシン上のすべてのネットワーク・インターフェース (論理および物理) から来ています。ファイル・システムはルート許可とともにエクスポートされます。

```
scadmin@thrash(220)# cat vfstab.log0
#device to mount             device  to fsck               mount
#                                                         point

/dev/vx/dsk/data0/data1-stat /dev/vx/rdsk/data0/data1-stat /log0
/dev/vx/dsk/data0/vol01      /dev/vx/rdsk/data0/vol01      /log0/home
/dev/vx/dsk/data0/vol02      /dev/vx/rdsk/data0/vol02      /log0/data

scadmin@thrash(221)# cat vfstab.log1
#device to mount             device  to fsck               mount
#                                                         point
/dev/vx/dsk/data1/data1-stat /dev/vx/rdsk/data1/data1-stat /log1
/dev/vx/dsk/data1/vol01      /dev/vx/rdsk/data1/vol01      /log1/home
/dev/vx/dsk/data1/vol02      /dev/vx/rdsk/data1/vol02      /log1/data
/dev/vx/dsk/data1/vol03      /dev/vx/rdsk/data1/vol03      /log1/data1



FS   fsck mount   options
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -


FS   fsck mount   options
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -
ufs  2    no      -
```

vfstab.log0 ファイルには、 /log0 ディレクトリーの下にあるファイル・システム用に、 3 つの有効な項目が含まれます。論理ホスト log0 のファイル・システムが論理ボリューム装置を使用していることに注意してください。これは、論理ホストに関連したディスク・グループ data0 の一部です。

vfstab ファイル内のファイル・システムは上から下の順にマウントされるので、ファイル・システムが正しい順序でリストされているか確認することは重要です。特定のファイル・システムの下にマウントされるファイル・システムは、上位のファイル・システムより下にリストしなければなりません。論理ホストで必要な実際のファイル・システムは、データ・サービスの必要に依存しており、これらの例とは大きく異なります。

フェイルオーバー時に、SC2.2 ソフトウェアは、論理ホストに関連するディスク・グループおよび論理インターフェースが、クラスター内のマシンの間で論理ホストとともに移動することを保証します。高可用性データ・サービスは、フェイルオーバーの後に、少なくともこれらのリソースが新規システム上で使用可能にされることを予期します。実際には、多くのデータ・サービスはそれらが高可用性であることに気付いてさえおらず、これらのリソースをフェイルオーバー後もまったく同じように「認識する」でしょう。

制御メソッド

制御メソッドは以下を使用して登録されます。

```
hareg(1m)
```

HA サービスが登録されると、 SC 2.2 はクラスターの再構成またはフェイルオーバー中の適切な時刻に、 HA サービスに登録されたメソッドを呼び出します。

クラスターの再構成 (制御フェイルオーバー) 時には、以下の処理が (指定された順序で) 行われます。マシンがクラッシュした場合、ステップ 5c より前のアクションは行われません。 (クラスターの再構成についての詳細は、 SC2.2 の資料を参照してください。)

1. FM_STOP メソッドが実行されます。
2. STOP_NET メソッドが実行されます。
3. 論理ホストの論理インターフェースがオフラインになります。
   - ifconfig hme0:1 0.0.0.0 down
4. STOP メソッドが実行されます。
5. ディスク・グループとファイル・システムが移動されます。
   a. 論理ホストのファイル・システムをアンマウントします。
   b. vxdg は 1 つのマシン上にディスク・グループをデポートします。

   - - 以下のステップはマシンがクラッシュした場合にのみ実行されます - -

   c. vxdg は 別のマシン上にディスク・グループをインポートします。
   d. 論理ホスト・ファイル・システムに対して fsck を実行します。
   e. 論理ホスト・ファイル・システムをマウントします。
6. START メソッドが実行されます。
7. 論理ホストの論理インターフェースがオンラインになります。
   - ifconfig hme0:1 <ip address> up
8. START_NET メソッドが実行されます。
9. FM_INIT メソッドが実行されます。
10. FM_START メソッドが実行されます。

制御メソッドは、以下のコマンド行引き数を使用して実行されます。

```
METHOD <logical hosts being hosted> <logical hosts not being hosted> <time-out>
```

最初の引き数は、現在ホストされている論理ホストのコンマ区切りリストで、 2 番目は、現在ホストされていない論理ホストのコンマ区切りリストです。最後の引き数は、メソッドのタイムアウト、つまり、SC2.2 ソフトウェアが打ちきるまでにメソッドを実行できる時間の合計です。

ディスクとファイル・システムの構成

SC2.2 は 2 つのボリューム・マネージャー Sun StorEdge Volume Manager (Veritas) および Solstice Disk Suite をサポートしています。両方とも正常に機能しますが、StorEdge Volume Manager は、クラスター環境でいくらか有利です。クラスター構成によっては、ディスク格納装置のコントローラー番号が、クラスター内のマシンごとに異なる場合があります。コントローラー番号が違うと、コントローラーのディスク装置へのパスも異なります。 Disk Suite はディスク装置へのパスを直接使用するので、この状況では十分機能しません。 StorEdge Volume Manager は、コントローラー番号に関係なく、ディスクそのものを使用するので、コントローラー番号が違っても影響を受けません。

HA の目標はデータ・サービスの可用性を上げることなので、すべてのファイル・システムおよびディスク装置がミラーリングされているか、または RAID 構成になっていることを確認することが重要です。これにより、ディスクの障害のために起きるフェイルオーバーは防止され、クラスターの安定度が増します。

HA-NFS

DB2 UDB EEE では、 1 つのインスタンスが複数のマシンにわたって構成されるときに、ファイル共用システムが必要です。一般的な DB2 UDB EEE 構成では、ホーム・ディレクトリーが NFS を介して 1 つのマシンからエクスポートされ、 EEE インスタンスに関与しているすべてのマシン上でマウントされています。相互引き受け構成では、 DB2 UDB EEE は HA-NFS を使用して、高可用性ファイル共用システムを提供します。論理ホストの 1 つが HA-NFS を介してファイル・システムをエクスポートしたら、クラスター内の各マシンは、そのファイル・システムを、 EEE インスタンスのホーム・ディレクトリーとしてマウントします。 HA-NFS についての詳細は、 Sun Cluster の資料を参照してください。

cconsole および ctelnet ユーティリティー

SC2.2 には、2 つの役立つユーティリティー `cconsole` および `ctelnet` が付属しています。これらのユーティリティーを使用すれば、単一のコマンドをクラスター内の複数のマシンに同時に発行することができます。これらのユーティリティーを使用して構成ファイルを編集すれば、確実にクラスター内の全マシンで構成ファイルを等しい状態に保つことができます。また、これらのユーティリティーを使用すれば、各マシンでまった

く同じ方法によりソフトウェアをインストールすることもできます。これらのユーティリティーについての詳細は、 Sun Cluster の資料を参照してください。

キャンパス・クラスタリングとコンチネンタル・クラスタリング

クラスター内のマシンが同じ建物にない場合、そのクラスターはキャンパス・クラスター と呼ばれます。キャンパス・クラスターは、単一の障害ポイントとして建物自体を除去する場合に役に立ちます。たとえば、クラスター内のマシンがすべて同じ建物にあり、それが焼け落ちた場合、クラスター全体に影響が及びます。しかし、マシンが複数の建物にあれば、建物の 1 つが燃えても、クラスターは存続します。

コンチネンタル・クラスター は、マシンが別々の都市に分散しているクラスターです。このクラスターの目標は、単一の障害ポイントとして地域を除去することです。このタイプのクラスターは、地震や津波のような災害からの保護を提供します。

現在、Sun Cluster は、10 km (約 6 マイル) の範囲内にあるマシンをサポートできます。このため、 2 つの異なるサイトの間での高速接続を必要とする人にとって、キャンパス・クラスタリングは実際的なオプションとなります。クラスターには、2 つの私用内部接続と、共用ディスクのための多数の光ファイバー・ケーブルが必要です。 2 つのサイトの間の高速接続にかかるコストは、その利点によって相殺されます。

一般的な問題

SC2.2 ソフトウェアは Cluster Configuration Database (CCD(4)) を使用して、クラスター構成用にクラスター全体の単一リポジトリーを提供します。 CCD には専用 API があり、 `/etc/opt/SUNWcluster/conf` ディレクトリーの下に保管されます。まれに、CCD の同期がとれなくなり、修正が必要となる場合があります。この状況で CCD を修正する最善の方法は、バックアップ・コピーから復元することです。

CCD をバックアップするには、クラスター内の全マシンでクラスター・ソフトウェアをシャットダウンし、 `/etc/opt/SUNWcluster/conf` ディレクトリーを tar 圧縮して、tar ファイルを安全な場所に保管します。バックアップを作成したときにクラスター・ソフトウェアがシャットダウンされていないと、 CCD を復元するときに問題が起きる可能性があります。クラスター構成が変更されたときに新規バックアップをとることにより、バックアップ・コピーが最新の状態に保たれるようにしてください。 CCD を復元するには、クラスター内の全マシン上でクラスター・ソフトウェアをシャットダウンし、 `conf` ディレクトリーを `conf.old` に移動して、バックアップ・コピーを tar 解凍します。クラスターは、新規の CCD を使用して開始できます。

DB2 に関する考慮事項

このセクションでは、以下のトピックについて説明します。

- 273ページの『HA インスタンスに接続するアプリケーション』
- 274ページの『EE および EEE インスタンスのディスクのレイアウト』

HA インスタンスに接続するアプリケーション

高可用性 DB2 インスタンスに依存するアプリケーションは、フェイルオーバー時に再接続が可能でなければなりません。論理ホストのホスト名および IP アドレスは同じままなので、別のホスト名に接続したり、データベースを再カタログする必要はありません。

2 つのマシンと 1 つの DB2 ユニバーサル・データベース エンタープライズ・エディションのインスタンスを持つクラスターを考慮してみましょう。 EE インスタンスは通常、クラスター内のマシンの 1 つに常駐します。 HA インスタンスのクライアントは、 HA インスタンスに関連した論理ホストの論理 IP アドレス (またはホスト名) に接続します。

HA クライアントによれば、 2 つのタイプのフェイルオーバーがあります。 1 つのタイプは、 HA インスタンスをホストしているマシンがクラッシュしたときに起こります。別のタイプは、 HA インスタンスを正常にシャットダウンする機会が与えられたときに起こります。

マシンがクラッシュして HA インスタンスをダウンした場合、データベースへの既存の接続および新規の接続は両方ともハングします。接続がハングするのは、ネットワーク上に、クライアントがデータベースに使用していた IP アドレスを持つマシンがないためです。データベースが正常にシャットダウンされると、 `db2stop force` はデータベースへの既存の接続を中断し、エラー・メッセージが戻されます。

フェイルオーバー時に、データベースに関連した論理 IP アドレスはオフラインになります。これは、SC2.2 ソフトウェアによってオフラインにされたためか、または論理ホストをホストしていたマシンがクラッシュしたためです。このとき、データベースへの新規の接続は、少しの間ハングします。

データベースに関連した論理 IP アドレスは、最終的に、DB2 が開始される前に別のマシンに移動されます。この段階では、データベースへの接続はハングしませんが、通信エラーを受け取ります。これは、DB2 がまだシステム上で開始されていないためです。データベースに接続されていた DB2 クライアントも、通信エラーを受け取り始めます。クライアントはまだ接続状態にあると考えますが、論理 IP アドレスをホストし始

めたマシンは、既存の接続を識別しません。接続は単純にリセットされ、 DB2 クライアントは通信エラーを受け取ります。少しした後、DB2 はマシン上で開始され、 DB2 への接続が正常に確立されます。この時点で、データベースが不整合になり、クライアントはそれが回復するのを待たなければならない場合があります。

HA 環境用のアプリケーションを設計する際、データベース接続がハングする場合のために特別なコードを書く必要はありません。接続は、 Sun Cluster ソフトウェアが論理 IP アドレスを移動させる少しの間だけハングします。 Sun Cluster 上で実行されるどのデータ・サービスにも、この段階で同様のハング接続があります。どのようにデータベースがダウンするかに関係なく、クライアントはエラー・メッセージを受け取り、正常に行われるまで再接続を試行しなければなりません。クライアントから見ると、 HA インスタンスがダウンしてから同じマシンに戻されたように見えます。制御されるフェイルオーバーでは、クライアントには、強制的に切断されて、後に同じマシンでデータベースへの再接続が可能であるように見えます。制御されないフェイルオーバーでは、クライアントには、データベース・サーバーがクラッシュしてから、すぐに同じマシンに戻されたように見えます。

EE および EEE インスタンスのディスクのレイアウト

DB2 は、必要とするディスク装置またはファイル・システムが、クラスター内の各マシン上で同じように見えることを予期します。これを確実にするためには、必要なディスクまたはファイル・システムを、 HA インスタンスに関連した論理ホストに従い、クラスター内の各マシンで同じパス名になるように構成する必要があります。

DMS および SMS 表スペースの両方とも、 HA 環境でサポートされています。 DMS 表スペースの装置コンテナーでは、ボリューム・マネージャーにより作成されたロー・デバイスを使用する必要があります。これらの装置は、ミラーリングされているか、または RAID 構成になっています。 `/dev/rdsk/c20t0d0s0` のような正規のディスク装置は、以下の理由から、使用すべきではありません。

- 装置に、同時に複数のマシンから書き込まれる可能性が高くなる。
- コントローラー番号が別のマシンで異なる可能性がある。

DB2 がこの状況でフェイルオーバーされると、必要なディスク装置は別のマシンの場合と同じように見えず、DB2 は始動しません。 DMS 表スペースのファイル・コンテナー、および SMS 表スペースのファイル・コンテナーは、マウントされたファイル・システムに常駐する必要があります。論理ホストのファイル・システムは、論理ホストの `vfstab` ファイルに組み込まれていると、自動的にマウントされます。

論理ホストの `vfstab` ファイルは以下のパスにあります。

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logical_host>
```

logical_host は、 `vfstab` ファイルに関連した論理ホストの名前です。

各論理ホストには独自の vfstab ファイルがあり、このファイルには、論理ホストのディスク・グループが現行のマシンに移された後で、 HA サービスが開始される前にマウントされるファイル・システムが含まれます。 Sun Cluster ソフトウェアは、ファイル・システムの正常性を保証するために、 **fsck** (ファイル・システム検査) を実行した後で適切に定義されたファイル・システムをマウントしようと試みます。 **fsck** が失敗すると、ファイル・システムはマウントされず、エラー・メッセージがログに記録されます。

注: プロセスにオープン・ファイルが含まれる場合、または現行の作業ディレクトリーがマウント・ポイントの下にある場合、マウントは失敗します。これを防ぐには、論理ホストの vfstab ファイルに含まれるマウント・ポイントの下に、プロセスが残されていないことを確認してください。

SMS 表スペースを使用するときは、任意の規則を EEE インスタンスのファイル・システムのレイアウトで使用することができます。以下に示すのは、hadb2_setup ユーティリティーで使用される規則です。

```
scadmin@crackle(190)# pwd
/export/ha_home/db2eee/db2eee
scadmin@crackle(191)# ls -l
total 18
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0000 -> /log0/disks/db2eee/NODE0000
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0001 -> /log0/disks/db2eee/NODE0001
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0002 -> /log0/disks/db2eee/NODE0002
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0003 -> /log0/disks/db2eee/NODE0003
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0004 -> /log0/disks/db2eee/NODE0004
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0005 -> /log1/disks/db2eee/NODE0005
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0006 -> /log1/disks/db2eee/NODE0006
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0007 -> /log1/disks/db2eee/NODE0007
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0008 -> /log1/disks/db2eee/NODE0008
scadmin@crackle(192)#
```

インスタンス所有者は db2eee で、 db2eee インスタンスのデフォルト・データベース・ディレクトリーは /export/ha_home/db2eee です。論理ホスト log0 は、データベース区画 0、1、2、および 3 をホストしており、論理ホスト log1 は、データベース区画 4、5、6、7、および 8 をホストしています。

各データベース区画ごとに、対応する NODE*xxxx* ディレクトリーがあります。データベース区画のノード・ディレクトリーは、関連する論理ホストのファイル・システムの下にあるディレクトリーを指しています。

パスの表記規則を選択するときは、以下のことを確かめてください。

1. ファイル・システムのディスクが、それを必要とするデータベース区画を担当する論理ホストのディスク・グループにある。
2. コンテナーを保持するファイル・システムが、論理ホストの vfstab ファイルによってマウントされている。

EE および EEE インスタンスのホーム・ディレクトリーのレイアウト

EE インスタンスの場合、ホーム・ディレクトリーは、論理ホストの vfstab ファイルで定義されたファイル・システムである必要があります。このディレクトリーは DB2 が開始される前に使用可能になり、クラスター内で論理ホストが移動される場所に DB2 とともに移されます。各マシンには vfstab ファイルの独自コピーがあるため、各マシンで同じ内容になるように注意を払う必要があります。以下に示すのは EE インスタンスのホーム・ディレクトリーの例です。

```
/log0/home/db2ee
```

/log0 は論理ホスト log0 の論理ホスト・ファイル・システムで、 db2ee は DB2 インスタンスの名前です。このホーム・ディレクトリーのパスは、「db2ee」インスタンスをホストするクラスター内の各マシンにある /etc/passwd ファイルに含める必要があります。

EEE インスタンスの場合、ホーム・ディレクトリーをセットアップする方法は 2 つあります。ホット・スタンドバイ構成の場合、ホーム・ディレクトリーは EE インスタンスと同じ方法でセットアップすることができます。相互引き受け構成の場合は、HA-NFS をホーム・ディレクトリーで使用し、 EEE インスタンスをセットアップする *前*に、適切に構成しなければなりません。

クラスター内のマシンのうちの 1 つは、選択した論理ホストの dfstab ファイルを使用して、 EEE インスタンスのファイル・システムをエクスポートする必要があります。dfstab ファイルには、マシンが論理ホストをホストしているときに NFS を介してエクスポートされるファイル・システムが含まれています。各マシンには dfstab ファイルの独自コピーがあるため、各マシンで同じ内容になるように注意を払う必要があります。

HA-NFS ファイル・システムに関する情報は、 (hadb2_setup プログラムによって) hadb2tab ファイルに置かれています。 HA エージェントがインスタンスに関する情報を読み取ると、インスタンスの HA-NFS ファイル・システムが自動的にマウントされます (280ページの『hadb2tab ファイル』を参照してください)。

HA-NFS ファイル・システムのマウント・ポイントは、通常、 /export/ha_home です。クラスター内の各マシンで、このマウント・ポイントは HA-NFS ディレクトリーをエクスポートする論理ホストから NFS マウントされます。 EEE インスタンス所有者のホーム・ディレクトリーは、以下のディレクトリーに置かれており、呼び出されます。

```
/export/ha_home/<instance>
```

instance は、インスタンス所有者の名前です。

ホーム・ディレクトリーをマウントしたりアンマウントしないですむように、各マシン上のインスタンスごとにホーム・ディレクトリーを持つこともできます。これを行うに

は、確実にホーム・ディレクトリーが各マシン上で同一になるように、追加の管理オーバーヘッドが必要になります。これが失敗すると、DB2 は正常に始動しないか、また別の構成で DB2 が始動されることになります。これは、サポートされている構成ではありません。

論理ホストと DB2 UDB EEE

論理ホストは通常、1 つまたは複数のデータベース区画をホストしたり、 HA-NFS ファイル・システムをエクスポートしたりするために選択されます。たとえば、クラスター内に 4 つのデータベース区画と 2 つのマシンがある場合、各マシンごとに 1 つの論理ホストが存在しなければなりません (278ページの図33)。一方の論理ホストで 2 つのデータベース区画のホストと HA-NFS ファイル・システムのエクスポートを行い、他方の論理ホストで残りの 2 つのデータベース区画のホストを行うことができます。

デフォルトでは、 DB2 UDB EEE インスタンスは、そのインスタンス用に活動状態にあるデータベース区画がすでに 1 つ以上存在しているマシンに、最大で 2 つのデータベース区画を正常に追加するのに十分のリソースを割り当てます。たとえば、クラスター上の単一インスタンス用に 4 つのデータベース区画がある場合は、論理ホストごとに 1 つのデータベース区画があるか、または 1 つの論理ホストが 3 つのデータベース区画をホストしている場合にのみ、このことが問題となります。どちらの場合も、同じインスタンスのデータベース区画をすでにホストしているマシンに対して、 3 つのデータベース区画をフェイルオーバーすることが可能です。

DB2_NUM_FAILOVER_NODES レジストリー変数を使用すれば、フェイルオーバーされるデータベース区画のために予約されたリソースの量を増やすことができます。

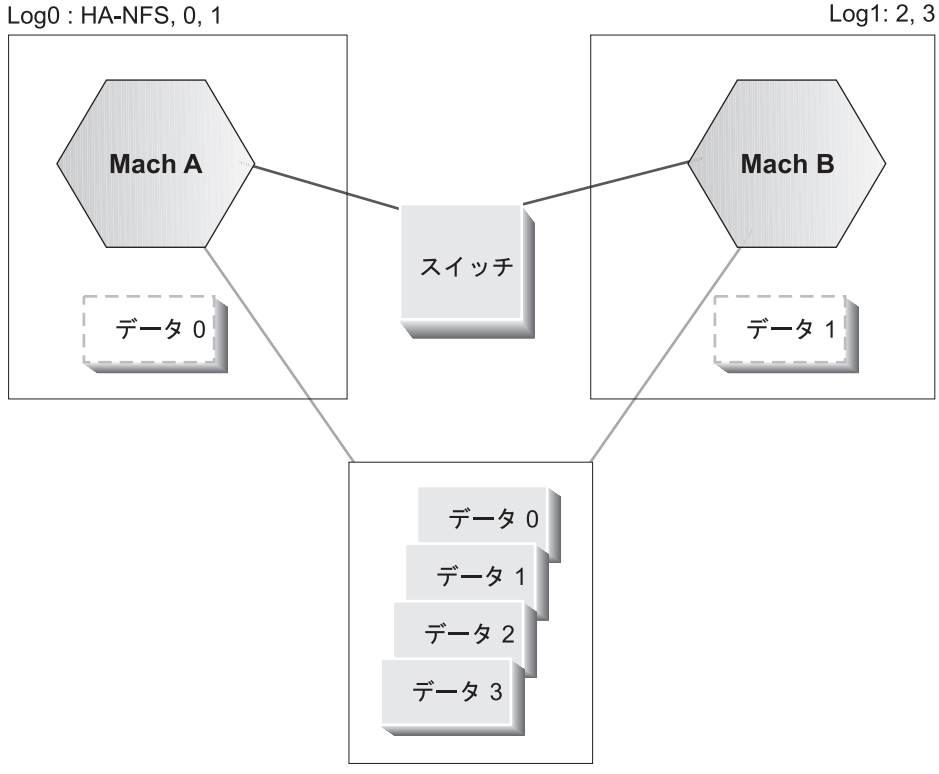


図 33. 各マシンごとに 1 つの論理ホストがある

DB2 のインストール場所およびオプション

DB2 がインストールされるファイル・システムは、ミラーリングされているか、または少なくとも RAID 構成にある必要があります。 DB2 が通常のディスクにインストールされていると、ディスク障害は起こりやすくなります。その結果起こるフェイルオーバーは防止可能なものと見なされるため、クラスターの安定度は減少します。

DB2 は、論理ホストのディスク・グループにあるディスクにはインストールできません。これは、HA エージェントが常に、DB2 ライブラリーへのアクセスを必要とするからです。 DB2 ライブラリーにアクセスできないと、HA エージェントには障害が起こります。 DB2 は通常、クラスター内の各マシンでインストールしなければなりません。

データベースおよびデータベース・マネージャー構成パラメーター

データベース・マネージャー構成パラメーターは、フェイルオーバーの後、 DB2 が始動される前に、 `pre_db2start` スクリプトを使用して変更することができます

(283ページの『ユーザー・スクリプト』を参照)。この実行可能スクリプトは、 (存在する場合に) インスタンス所有者のホーム・ディレクトリーの sqllib/ha ディレクトリーの下で実行されます。名前から分かるように、これは **db2start** の直前に実行されます。インスタンスが EEE インスタンスでない限り、 pre_db2start スクリプトには、制御メソッドに渡されたのと同じ引き数が渡されます。 EEE インスタンスの場合は、pre_db2start スクリプトにも、 **db2start** コマンドのノード番号が渡されます。

破損回復

HA 環境での破損回復は、通常の環境での破損回復と同じです。 HA インスタンスが、クラッシュしたマシンとは別のマシンで再開されたとしても、インスタンスのファイルおよびディスク装置は同じように見え、データベースを回復するのに必要なアクションも異なりません。破損回復や他の形式のデータベース回復についての詳細は、 3ページの『第1章 適切なバックアップと回復の方針の開発』 を参照してください。

データベースは手作業で (または、ユーザー・スクリプトの 1 つによって) 再始動できますが、特に EEE インスタンスの場合には、 *autorestart* データベース構成パラメーターを ON に設定するようお勧めします。これにより、データベースが不整合の状態になる時間を最小限にとどめることができます。

データ複製による高可用性

データの可用性は、複製によっても強化することができます。 2 つのサーバー間でデータを複写することにより、 1 つの形式の高可用性が実現します。サーバーの 1 つがダウンすると、別のサーバーがそれを引き継ぎ、データ・サービスを提供し続けることができます。

ただし、複製は非同期に行われるので、サーバーがダウンしたときに一部の変更内容が別のサーバーに伝えられない場合もあります。

Sun Cluster 2.2 での DB2 コネクトの前提条件

Sun Cluster 2.2 で DB2 コネクトがサポートされるのは以下の場合です。

- ホストへのプロトコルが TCP/IP である (SNA ではない)
- 2 フェーズ・コミットが使用されていない。ユーザーが、SPM ログを共用ディスク上にあるように構成すると、 (*spm_log_path* データベース・マネージャー構成パラメーターを使用して可能) この制約事項は緩和されます。また、フェイルオーバー・マシンが同一の TCP/IP 構成 (同一のホスト名、 IP アドレス、など) を持っている場合も同様です。

DB2 高可用性エージェント

DB2 高可用性エージェントは、 DB2 と SC2.x の間の仲介者として機能します。このエージェントにより、DB2 についての詳しい知識がなくても、 Sun Cluster 2.2 ソフトウェアはクラスター環境で DB2 を制御することができます。 EE および EEE 両インスタンスのための 1 つのエージェントがあります。このエージェントは管理インスタンスとデータベース・インスタンスの両方をサポートします。

hadb2 サービスの登録

SC2.2 を使用して作業するには、 DB2 HA エージェントが登録されていなければなりません。データ・サービスを登録すると、使用可能な制御メソッドとそれらが常駐するディレクトリーが SC2.2 に知らされます。 HA エージェントに付属している特殊なスクリプト hadb2_reg は、 hadb2 サービスを EE および EEE インスタンスの両方に対して登録できます。 hadb2_reg スクリプトは、クラスター全体に対して 1 度だけ実行する必要があります。

DB2 HA エージェントの制御メソッドのセットは 1 つだけですが、それらが登録される方法は、 EEE インスタンスが相互引き受け構成で使用されるかどうかによって異なります。ホット・スタンドバイ構成の EE インスタンスまたは EEE インスタンスの場合、 HA-NFS は使用されません。したがって、 hadb2 サービスが HA-NFS に依存することを SC2.2 ソフトウェアに知らせる -d nfs スイッチは必要ありません。

DB2 V7.1 制御メソッドを EEE インスタンス用に登録するために hadb2_reg が使用する実際のコマンドは以下の通りです。

```
hareg -r hadb2 -b /opt/IBMdb2/V7.1/ha -m
START=hadb2_start,START_NET=hadb2_startnet,STOP_NET=hadb2_stopnet,
FM_START=hadb2_fmstart,FM_STOP=hadb2_fmstop
-t START_NET=$TIMEOUT,STOP_NET=$TIMEOUT -d nfs
```

-b スイッチは、すべての制御メソッドを opt/IBMdb2/V7.1/ha ディレクトリーで探すように SC2.x に指示します。 -m スイッチは、 hadb2 サービス用の実際の制御メソッドを定義します。 -t スイッチは、 START_NET および STOP_NET 制御メソッドのタイムアウトを定義します。各制御メソッドについての詳細は、 Sun Cluster の資料を参照してください。

hadb2_unreg スクリプトを使用すると、 hadb2 サービスを登録解除することができます。これは、hadb2_reg と同様、クラスターで 1 度だけ実行する必要があります。

hadb2tab ファイル

hadb2tab ファイルは、 DB2 HA エージェント用の主な構成ファイルです。各制御メソッドはこのファイルを調べて、可用性の高いインスタンスを見つけ出します。 hadb2tab ファイルは、 DB2 UDB バージョン 7.1 の /var/db2/v71/ ディレクトリーの下にあります。このファイルは複数インスタンスをサポートしており、それぞれの非コメント行は、それぞれの HA インスタンスを表します。以下は、hadb2tab ファイルの例です。

```
<scadmin@thrash(203)# cat hadb2tab
EEE DATA db2eee jolt ON /export/ha_home /log0/home #Added by DB2 HA software
EE ADMIN db2ee log1 ON  -               -          #Added by DB2 HA software
```

最初のフィールドは、インスタンスが EE インスタンスであるか、または EEE インスタンスであるかを DB2 HA エージェントに示します。 2 番目のフィールドは、インスタンスがデータ・インスタンスであるか、または管理インスタンスであるかを示します。 3 番目のフィールドには、 HA インスタンスのユーザー名が含まれています。 4 番目のフィールドは、インスタンスの論理ホストまたは HA-NFS ホストです。これは、インスタンスが EE インスタンスと EEE インスタンスのいずれであるかに依存します。 5 番目のフィールドは、インスタンスの障害モニターのオン / オフを示します。最後の 2 つのフィールドは、それぞれ、ローカル・マウント・ポイントとリモート HA-NFS ディレクトリーです。これらのフィールドは、使用されない場合は - (ハイフン) に設定され、 EEE 相互引き受け構成でのみ使用されます。行で「#」マーカーより前にある情報が、長さ 0 であるか、またはインスタンスの有効な定義であれば、 `hadb2tab` ファイルでコメントが使用できます。

制御メソッド

SC2.2 エージェントの制御メソッドは、一連のスクリプトまたはプログラムです。 DB2 (Solaris 版) のエージェントは、以下のメソッドを含む一連のプログラムです。

START_NET
: hadb2_startnet が、DB2 を始動するのに使用されます。

STOP_NET
: hadb2_stopnet が、DB2 を停止するのに使用されます。

FM_START
: hadb2_fmstart が、 DB2 の障害モニターを始動するのに使用されます。

FM_STOP
: hadb2_fmstop が、 DB2 の障害モニターを停止するのに使用されます。

これらの制御メソッドについての詳細は、 Sun Cluster の資料を参照してください。

EE インスタンスの場合、インスタンスに関連する論理ホストは、 `hadb2tab` ファイルで定義されます。ただし、EEE インスタンスの場合、制御メソッドは以下のファイルも参照しなければなりません。

```
~<instance>/sqllib/ha/hadb2-eee.cfg
```

`~<instance>` は、インスタンス所有者のホーム・ディレクトリーです。このファイルには各データベース区画ごとに 1 つの行が含まれ、データベース区画と論理ホストを関連付けるために使用されます。以下に示すのは、有効な `hadb2-eee.cfg` ファイルの例です。

```
crackle % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

インスタンスまたはデータベース区画は、クラスター内で対応する論理ホストについていきます。論理ホストは、基礎となるハードウェアおよび SC2.2 によってサポートされるクラスター内のマシンならどれにでも移動できます。構成が適切にセットアップされていれば、 DB2 は SC2.2 ソフトウェアによってサポートされるどのトポロジーもサポートします。

インスタンスの全情報を読み取った後、制御メソッドは、インスタンスと関連する論理ホストを認識しています。コマンド行引き数を解析した後、制御メソッドは、現行のマシンによりホストされる論理ホストとホストされない論理ホストを認識しています。

以下の表は、実行されている制御メソッドと、データベース区画またはインスタンスと関連した論理ホストが現行のマシンでホストされているかどうかによって行われるアクションを示しています。

制御メソッド	関連した論理ホストがホストされている場合	関連した論理ホストがホストされていない場合
START_NET	DB2 インスタンスまたはデータベース区画を開始する	処置なし
STOP_NET	処置なし	DB2 インスタンスまたはデータベース区画を停止する
FM_START	インスタンスの障害モニターを開始する	処置なし
FM_STOP	処置なし	インスタンスの障害モニターを停止する

開始アクションを実行する制御メソッドは、現在ホストされている論理ホストだけを処理し、停止アクションを実行する制御メソッドは、現在ホストされていない論理ホストだけを処理します。

HA-NFS が使用されている場合、制御メソッドは、特別な方法で HA-NFS ディレクトリーをマウントする必要もあります。 HA-NFS のローカル・マウント・ポイントおよびディレクトリーが - (ハイフン) として定義されていない場合、制御メソッドはローカル・マウント・ポイントで `statvfs(2)` を実行します。ローカル・マウント・ポイントのファイル・システムのタイプが `nfs` でない場合、エージェントは、 `hadb2tab` 行の情報を使用してファイル・システムのマウントを試みます。 HA-NFS のマウント・ポイントおよびディレクトリーが - (ハイフン) として定義されていない場合、対応する論理ホストの `vfstab` ファイルが、インスタンスのホーム・ディレクトリーを含むファ

イル・システムをマウントするために必要となります。 HA-NFS のローカル・マウント・ポイントおよびリモート・ディレクトリーは、 EE および EEE のホット・スタンドバイ構成の場合、 - (ハイフン) としてのみ定義できます。

ユーザー・スクリプト

このスクリプトは制御メソッドから実行され、機能を追加します。このスクリプトには制御メソッドと同じコマンド行引き数が渡され、システム管理者またはデータベース管理者によって書き込まれます。

バックグラウンドで実行されないスクリプト内からプログラムを実行しなければならない場合、 nohup(1) を使用してそのプログラムをバックグラウンドで実行することを考慮してください。 nohup プログラムは、実行されているプログラムを、 SIGHUP (またはハングアップ) シグナルから保護します。 nohup を使用しないと、バックグラウンドでスクリプトから実行されているプログラムは、スクリプトの終了時に SIGHUP シグナルの結果として強制終了される場合があります。

制御メソッドは以下のスクリプトを実行します。

- /var/db2/v61/failover
- ~<instance>/sqllib/ha/pre_db2start
- ~<instance>/sqllib/ha/post_db2start
- ~<instance>%s/sqllib/ha/post_failover
- ~<instance>/sqllib/ha/pre_db2stop
- ~<instance>/sqllib/ha/fm_warning

~*instance* は、 HA インスタンスのホーム・ディレクトリーです。

fm_warning スクリプトを除き、各ユーザー・スクリプトは、そのスクリプトを呼び出した制御メソッドと同じ引き数を使用して実行されます。 EEE インスタンスを使用している場合、データベース区画番号も (最後の引き数として) ユーザー・スクリプトに渡されます。

/var/db2/v71/failover スクリプトは、START_NET メソッドの最初に呼び出され、バックグラウンドで実行されます。このスクリプトを使用すると、たとえばフェイルオーバーが起きたときにサポート・スタッフに電子メールを出すことができます。以下に、フェイルオーバー・スクリプトの例を示します。

```
#!/bin/ksh

# E-mail or page support staff to notify them that a failover has occurred.

echo "Failover occurred on machine 'hostname':Running $0!"
   |/bin/mail admin@sphere.torolab.ibm.com
```

スクリプトから正常に電子メールを出すためには、 sendmail(1m) が適切にシステムで構成されていなければなりません。

名前から分かるように、 pre_db2start スクリプトは、 **db2start** が呼び出される直前に実行されます。このスクリプトは、データベース・マネージャー構成パラメーターの変更などのタスクで使用することができます。完了までに最大で 20 秒が与えられています。 EEE インスタンスの場合、このスクリプトは **db2start** が各データベース区画で呼び出される前に実行されます。このスクリプトは、データ・インスタンスにのみ適用でき、管理インスタンスには適用できません。

同様に、post_db2start スクリプトは、 **db2start** が呼び出された*直後に* 実行されます。このスクリプトは、データベースの再始動のようなタスクで使用することができます。これは、バックグラウンドで実行されるので、その実行時間が他のインスタンスに支障をきたすことはありません。このスクリプトは、データ・インスタンスにのみ適用でき、管理インスタンスには適用できません。

インスタンス所有者のホーム・ディレクトリーの下にある post_failover スクリプトは、インスタンスの処理の後で実行されます。このスクリプトを使用すると、 DB2 が機能していることをクライアント・アプリケーションに知らせたり、データベースを活動化したり、状況ファイルを管理者に送信したりすることができます。これは、バックグラウンドで実行されるので、その実行時に別の HA インスタンスに対するアクションが遅れることはありません。以下に、フェイルオーバー後処理スクリプトの例を示します。

```
#!/bin/ksh
#

# Send the status file to the administrato-r.
mail admin@sphere.torolab.ibm.com </tmp/HA.info.db2eee
```

DB2 HA エージェントの START_NET および STOP_NET メソッドは両方とも、各インスタンスの処理の後に状況ファイルを作成します。状況ファイルの名前は以下の通りです。

```
/tmp/HA.info.<instance>
```

instance は、インスタンス所有者のユーザー名です。状況ファイルには、インスタンスの開始および停止レポートと、制御メソッドを実行するのにかかった時間が含まれます。以下に、状況ファイルの例を示します。

```
scadmin@crackle(173)# cat /tmp/HA.info.db2eee
-----  Elapsed Time: 00:00:18           -----
-----  Elapsed Time: 00:00:00  (HA-NFS) -----

NODE     ACTION     RESULT     TRIES     RC
----     ------     ------     -----     --
   4     stop       success        3    1064
   5     stop       success        1    1064
   6     stop       success        2    1064
```

```
   7     stop       success        2    1064
   8     stop       success        1    1064
-------------------------------------------
```

pre_db2stop スクリプトは、 **db2stop** が呼び出される直前に実行されます。このスクリプトを使用すると、 DB2 が停止しようとしていることをクライアント・アプリケーションに通知することができます。完了までに最大で 20 秒が与えられます。このスクリプトは、データ・インスタンスにのみ適用でき、管理インスタンスには適用できません。

障害モニターも、予期せぬシャットダウンのために DB2 が再始動されたときに、ユーザー・スクリプトを実行します。以下のスクリプトが呼び出されます。

```
~<instance>/sqllib/ha/fm_warning
```

fm_warning スクリプトを使用すると、 DB2 が障害モニターによって再始動されたことをシステム管理者に通知することができます。システム管理者は、 DB2 が突然シャットダウンした理由を突き止め、これが再び起きないように適切な処置を取る必要があります。 fm_warning スクリプトはバックグラウンドで実行されます。

他の考慮事項

HA データ・サービスがオフにされると、フェイルオーバーまたはクラスターの再構成時には停止メソッドだけが実行されます。他のメソッドは、HA データ・サービスが適切に登録され、オンにされた場合にのみ実行されます。

クラスター内の各マシンに、そのマシンが担当できるすべてのデータ・サービスを実行するのに十分なリソースがあることを確認してください。 CPU のロード、メモリー、スワップ、およびカーネル・パラメーターなどのリソースは、クラスターが実働する前に考慮しなければなりません。たとえば、クラスター内のマシンが 2 つの DB2 インスタンスを実行する必要がある場合、そのマシンのカーネル・パラメーターは、各インスタンスで必要とされるものの合計でなければなりません。

障害モニター

障害モニターがオンにされると、障害モニターはクラスターの再構成時またはフェイルオーバー時に始動されます。 DB2 が START_NET スクリプトによって始動されていない場合、障害モニター自身が DB2 を始動します。障害モニターは、 DB2 が始動しなかったかどうか、または不明な理由でシャットダウンされたかどうかを検出できます。このため、障害モニターがオンになっているときは、DB2 を手作業でシャットダウンしないことが重要です。障害モニターはこれを予期しないシャットダウンと見なし、 DB2 を再始動します。これがあまりにも多く起きる場合、適切な論理ホストがフェイルオーバーされます。

障害モニターがインスタンスで使用可能になっているとき、インスタンスを手作業で開始または停止する正しい方法は、最初に障害モニターまたは hadb2 サービスをオフにす

ることです。これらのアクションの両方とも、 -f および -s スイッチを使用した **hadb2_setup** コマンドによって開始できます (290ページの『hadb2_setup コマンド』を参照)。

注: 同じ論理ホストについて複数のインスタンスを使用しないでください。複数のインスタンスが 1 つの論理ホストと関連付けられていると、良好な状態にあるインスタンスが、不良の状態にあるインスタンスとともにフェイルオーバーされる可能性があります。

EEE に関する考慮事項

論理ホストと関連させるデータベース区画を決定するときは、それらのデータベース区画がどのようにフェイルオーバーするかを考慮することが重要です。 2 つのマシン間にある 4 つのデータベース区画で使用される 2 マシン・クラスターを考慮してみましょう。これは、図34 に示されています。

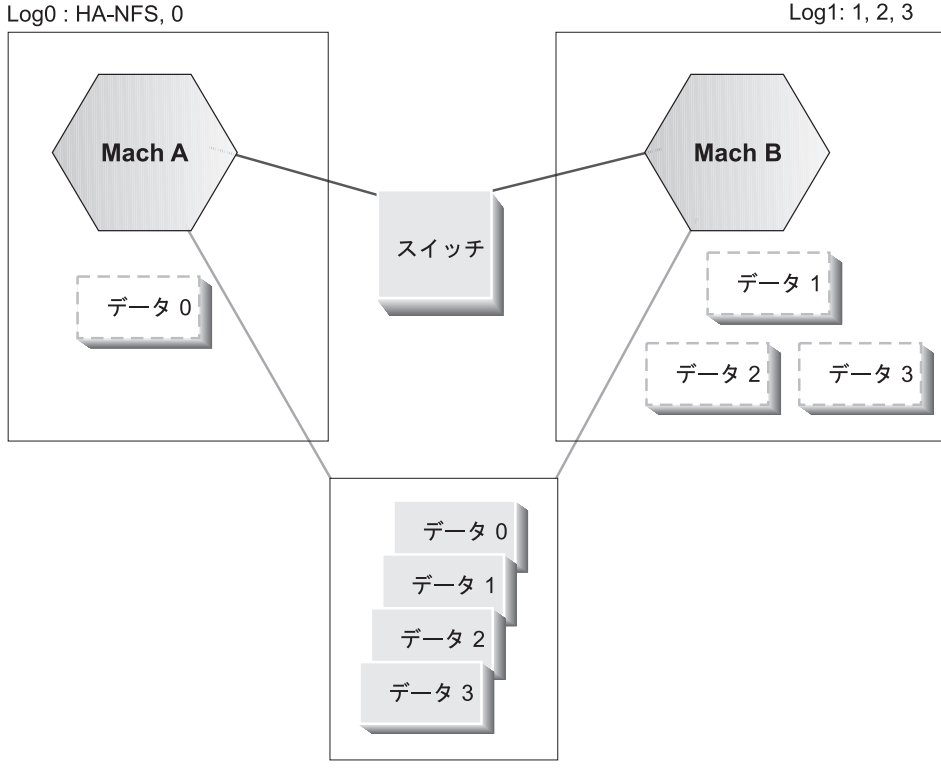


図 34. 4 つのデータベース区画を持つ 2 マシン・クラスター

一方の論理ホストを各データベース区画と関連付け、他方を HA-NFS に関連付けます。この場合、すべての論理ホストが 1 つのシステムによってホストされていると、問題が生じる可能性があります。システムに障害が起きる場合、すべての論理ホストを同

時にシステムから移動しなければなりません。不運にも、 Sun Cluster は予測可能な順序で論理ホストを移動しないので、データベース区画が関連付けられている論理ホストが HA-NFS の論理ホストの前に移動する可能性があります。単一のシステムでホストされるものごとにデータベース区画をグループ分けするのは良いアイデアです。このとき、通常 1 つのマシンでホストされる 2 つのデータベース区画が、単一の論理ホストと関連付けられています。

EEE インスタンスによって使用される db2nodes.cfg ファイルは、データベース区画が常駐するマシンを示すように更新されます。たとえば、すべてのデータベース区画が「crackle」というマシンにある場合、 db2nodes.cfg ファイルは以下のようになります。

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 crackle 4 204.152.65.33
5 crackle 5 204.152.65.33
6 crackle 6 204.152.65.33
7 crackle 7 204.152.65.33
8 crackle 8 204.152.65.33
```

これらのデータベース区画の一部が「thrash」というマシンに移動すると、 db2nodes.cfg ファイルは以下のように更新されます。

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 thrash 0 204.152.65.34
5 thrash 1 204.152.65.34
6 thrash 2 204.152.65.34
7 thrash 3 204.152.65.34
8 thrash 4 204.152.65.34
```

マシン名「thrash」を反映するようにホスト名とスイッチ名の両方が変更されることと、ポート番号も異なることに注意してください。

HA.config ファイル

存在する場合、 /etc/HA.config ファイルには、いくつかの構成オプションが含まれます。これには、以下のものが含まれます。

```
scadmin@thrash(204)# cat /etc/HA.config
SYSLOG_FACILITY=LOG_LOCAL3
SYSLOG_LPRIORITY=LOG_INFO
SYSLOG_EPRIORITY=LOG_ERR
USE_INTERCONNECT=auto
SWITCH_NAME=204.152.65.18
DEBUG_LEVEL=2
```

```
FAILS_PER_HOUR=2
FAILS_PER_DAY=4
FAILS_PER_WEEK=10
FM_FAIL_SEV=soft
DB2START_TIMEOUT=60
DB2STOP_TIMEOUT=500
SCRIPT_USER=bin
```

注: HA.config ファイルが存在しない場合、デフォルト値が使用されます。

SYSLOG_FACILITY 変数は、メッセージおよびエラーの両方をログに記録するための SYSLOG 機能を設定します。 SYSLOG_LPRIORITY および SYSLOG_EPRIORITY 変数は、情報メッセージおよびエラー・メッセージのそれぞれをログに記録するための SYSLOG 優先順位を設定します。

SYSLOG デーモンが DB2 HA エージェントからの情報をログに記録できるようにするために、いくらかの変更が必要となる場合もあります。たとえば、 /etc/syslog.conf ファイルに追加された以下の 2 行のうちの 1 つは、 SYSLOG デーモンに、情報をログ・ファイルに書き込むよう指示します。

```
*.notice                                        /var/adm/SC.x
local3.info                                     /var/adm/SC.LOG_LOCAL3
```

Sun Cluster は通常、高速内部接続を行います。 DB2 との間で高速内部接続を使用するには、 USE_INTERCONNECT を auto または override に設定します。 auto 設定 (デフォルト) では、 Sun 内部論理ネットワーク・インターフェースを使用します。最初のインターフェースに障害が起きると、このインターフェースは別の物理インターフェースに移されます。 USE_INTERCONNECT が override に設定されている場合、スイッチ名は SWITCH_NAME 変数からとられます。あるいは、 USE_INTERCONNECT を no に設定することができます。これは高速内部接続を使用しないことを指定します。

DEBUG_LEVEL は、フェイルオーバー時にログに記録される情報の量を指定します。これは 0 ～ 10 の数字であり、10 が最高のデバッグ・レベルです。情報は、指定した SYSLOG 優先順位および機能でログに記録されます。問題が生じた場合は、デバッグ・レベルを最大レベルに設定し、 SYSLOG が HA エージェントからの出力をログを記録するように構成してから、 SYSLOG 出力を IBM サービスに送ってください。

変数 FAILS_PER_HOUR、FAILS_PER_DAY、および FAILS_PER_WEEK は、 DB2 障害モニターが論理ホストをフェイルオーバーするときを決定するのに役立ちます。各 HA 環境は異なっています。許容できる DB2 障害の数を決めなければなりません。「許容できる」障害が起きるたびに、 DB2 は同じマシン上で再始動されます。これら 3 つの障害しきい値のうちの 1 つを超えると、インスタンスまたはデータベース区画に関連した論理ホストがフェイルオーバーされます。

FM_FAIL_SEV 変数は、フェイルオーバーが「ソフト」であるか「ハード」であるかを指定します。詳細については、 Sun Cluster の資料で hactl(1m) を参照してください。

DB2START_TIMEOUT および DB2STOP_TIMEOUT 変数は、 **db2start** および **db2stop** が実行可能な最大秒数を指定します。指定した間隔が過ぎると、 HA エージェントは操作が失敗したと見なし、インスタンスの再始動を試みます。

特定のインスタンスと関連していないユーザー・スクリプトがあります。通常、これらのスクリプトはルートとして実行されますが、これは、SCRIPT_USER 変数によりオーバーライド可能です。 SCRIPT_USER 変数を設定すれば、これらのスクリプトを実行できるユーザー ID を指定できます。

制御メソッドが DB2 コマンドを実行する方法

DB2 HA エージェントは、 **su** コマンドを使用して、インスタンス所有者としてコマンドを実行します。実際のコマンドは以下のようなものです。

```
su - <instance> -c "db2stop"
```

instance は、インスタンスのユーザー名です。

インスタンス所有者の `.profile` ファイルが **su** に適したものであるようにすることは重要です。そうでないと、 **su** コマンドは正常に機能しない可能性があります。 **su** コマンドを手作業で、またはスクリプトから呼び出して、コマンドを正常に実行できることを確認してください。

セットアップ

このセクションを読む前に、必ず SC2.2 ソフトウェアに精通しておいてください。このセクションでは、読者が SC2.2 と HA-NFS のセットアップ方法を理解しており、ボリューム・マネージャーの使用方法を理解しているものと想定しています。 DB2 で必要な他のパッチに加えて、 HA エージェントでは以下のパッチが必要です。

```
Solaris 2.6:
   105210-17 (またはそれ以降)
   105786-05 (またはそれ以降)
```

注: Solaris 7 (Solaris 2.7) ではパッチは必要ありません。

一般的なインストール・ステップ

1. SC2.2 をクラスター内の全マシンにインストールする。インストール時に、 SC2.2 はインストールするエージェントを尋ねてきます。 DB2 は SC2.2 に付属していないので、これはエージェントのリストにはありません。 DB2 のエージェントは DB2 とともにインストールされ、 **hadb2_reg** コマンドによって登録されます。
2. 論理ホストをディスク・グループおよび論理 IP アドレスと共に構成する。

DB2 UDB エンタープライズ・エディションでのセットアップ

1. 論理ホストの論理ホスト・ファイル・システムの下に、インスタンスのホーム・ディレクトリーを作成する。
2. DB2 をクラスター内の全マシンにインストールする。
3. インスタンスのホーム・ディレクトリーが現在あるクラスター内のマシンに、インスタンスを作成する。
4. クラスター内の別のマシンにインスタンスのユーザーを追加し、数値ユーザー ID が同じであることを確認する。
5. **hadb2_reg** コマンドを使用して、 hadb2 サービスを登録する。
6. **hadb2_setup** コマンドを実行して、インスタンスの HA をセットアップする。

DB2 UDB エンタープライズ拡張エディションでのセットアップ

1. HA インスタンス所有者のホーム・ディレクトリーを作成する。
   a. ホット・スタンドバイの場合、論理ホストの論理ホスト・ファイル・システムの下に、インスタンスのホーム・ディレクトリーを作成する。
   b. 相互引き受けの場合、 HA-NFS を構成し、論理ホストの 1 つからホーム・ディレクトリーをエクスポートする。マシンの 1 つで、選択したマウント・ポイントの下に HA-NFS ディレクトリーをマウントする。
2. DB2 をクラスター内の全マシンにインストールする。
3. HA-NFS ファイル・システムをマウントしたマシンに、インスタンスを作成する。
4. クラスター内の別のマシンにインスタンスのユーザーを追加し、数値ユーザー ID が同じであることを確認する。
5. **hadb2_reg** コマンドを使用して、 hadb2 サービスを登録する。
6. **hadb2_setup** コマンドを実行して、インスタンスの HA をセットアップする。

注: NIS を使用して HA インスタンスの情報を定義することはお勧めしません。これは、NIS によって単一の障害ポイントが生じる可能性があるためです。

hadb2_setup コマンド

hadb2_setup コマンドは、 DB2 HA エージェントに付属するプログラムの中心をなすものです。これを使用すると、インスタンスのセットアップ、修正、または削除を行えます。また、 hadb2_setup サービスのオン / オフも行えます。このコマンドを使用すれば、 hadb2tab ファイルを手作業で編集する必要はありません。

注: **hadb2_setup** コマンドは、実行されるマシン上でのみ、アクションを実行します。1 つのマシンに対して与えられる変更は、クラスター内の別のマシンにも行われます。

以下の引き数がサポートされています。

```
EE インスタンスを追加する場合:
---------------------
   hadb2_setup -a -i <instance> -f [on|off] -h <logical_host> -p [DATA|ADMIN] -t EE

   たとえば、以下のようになります。
      hadb2_setup -a -i db2ee -f off -h log1 -p DATA -t EE


EEE インスタンスを追加する場合:
----------------------
   hadb2_setup -a -i <instance> -f [on|off] -h <nfs_host> -l <mount_point> ¥
      -r <ha-nfs_dir> -p [DATA|ADMIN] -t EEE -n "<node_info>"

   たとえば、以下のようになります。
      hadb2_setup -a -i db2eee -f off -h ha-sun1 -l /export/ha_home ¥
         -r /log0/home -p DATA -t EEE -n "log0[0,10,20],log1[30,40,50]"


インスタンスを削除する場合:
---------------------
   hadb2_setup -d -i <instance>


インスタンスを修正する場合:
---------------------
   hadb2_setup -m -i <instance> [-f [on|off] | -l <mount_point> | ¥
      -h <host> | -p [DATA|ADMIN] | -r <ha-nfs_dir> | -t [EE|EEE] ]


他のオプション:
-------------
   -s   <on|off>             (すべての HA インスタンスで) hadb2 のオン / オフを行う
   -y                        安全チェックで「はい (yes)」を想定する
```

hadb2 サービスをオンまたはオフにするには、 -s スイッチを指定してください。これは、-n および -y スイッチを指定して **hareg** コマンドを実行し、 hadb2 サービスを指定することに相当します。 **hareg(1m)** コマンドについての詳細は、 Sun Cluster の資料を参照してください。

インスタンスの障害モニターは、 -f スイッチを使用してオフにすることができます。これは、ローカル・マシン上にあるインスタンスの障害モニターを停止させる効果と、hadb2tab ファイルを修正して障害モニターが停止したことを反映させる効果があります。

EE インスタンスの場合、インスタンスがフェイルオーバーされるときに全マシンの障害モニターをオフにすることをお勧めします。 EEE インスタンスの場合は、手作業でシャットダウンする前に、インスタンスのデータベース区画をホストしている全マシン上で障害モニターをオフにしなければなりません。

インスタンスを削除するには、 -d スイッチを使用します。これは、 hadb2tab ファイルからインスタンスを除去するだけです。他のファイルやディレクトリーを除去したり修正したりすることはありません。 hadb2tab ファイルは HA-DB2 エージェントの主な構成ファイルなので、このファイルからインスタンスを除去すると、制御メソッドはその存在を認識しなくなります。

インスタンスを修正するには、 -m スイッチを使用します。これは、 hadb2tab ファイルの情報を変更するだけです。他のファイルやディレクトリーを除去したり修正したりすることはありません。 -m スイッチは、 hadb2tab ファイルにある情報に関する任意のスイッチと共に使用することができます。 db2nodes.cfg ファイルおよび hadb2-eee.cfg ファイルは、最初のセットアップの後に手作業で変更しなければなりません。これは、**hadb2_setup** コマンドがこれらのファイルの修正をサポートしていないためです。

インスタンスの追加には、さらに多くのことが関係しています。

EE インスタンスの場合、以下の引き数が必要です。

```
hadb2_setup -a -i <instance> -f <fm> -h <logical_host> -t <EEE_or_EE>
   -p <purpose>
```

instance は追加されるインスタンスの名前です。 *fm* は障害モニターを最初にオンにするかオフにするかを指定します。 *logical_host* は関連する論理ホストです。 *EEE_or_EE* は EE に設定されます。 *purpose* は DATA または ADMIN のいずれかになります。

EEE インスタンスの場合、以下の引き数が必要です。

```
hadb2_setup -a -i <instance> -f <fm> -h <nfs_host> -t <EEE_or_EE> -p
   <purpose> -l <mount_point> -r <HA-NFS_directory> -n <node_info>
```

instance は追加されるインスタンスの名前です。 *fm* は障害モニターを最初にオンにするかオフにするかを指定します。 *nfs_host* は HA-NFS ファイル・システムをエクスポートする論理ホストのホスト名です。 *EEE_or_EE* は EEE に設定されます。 *purpose* は DATA または ADMIN のいずれかになります。 *mount_point* は HA-NFS ディレクトリーのローカル・マウント・ポイントです。 *HA-NFS_directory* は HA-NFS ディレクトリーです。 *node_info* はデータベース区画と論理ホストを関連させる情報です。たとえば、以下のようになります。

```
hadb2_setup -a -i db2eee -f on -h jolt -l /export/ha_home -p DATA -t EEE -r
   /log1/home -n "log0[0,1],log1[2,3]"
```

EEE インスタンスを追加するとき、ノード情報は引用符で閉じる必要があります。この例では、インスタンス「db2eee」は 2 つの論理ホスト「log0」および「log1」と関連付けられます。「db2eee」インスタンスのデータベース区画「0」および「1」は論理ホスト「log0」と関連付けられ、データベース区画「2」および「3」は論理ホスト「log1」と関連付けられます。

hadb2_setup コマンドを使用すると、インスタンスをクラスター内の全マシンに追加できます。インスタンスはその後、強制的にクラスターの再構成を行うか、または hadb2 サービスをオフにしてからオンにすることによって始動させることができます。これを行うには、**hareg** コマンドを実行するか、 -s スイッチを指定して **hadb2_setup** コマンドを実行します。インスタンスが始動しない場合は、 296ページの『トラブルシューティング』を参照してください。

hadb2_setup コマンドで EEE インスタンスを追加すると、以下のアクションが透過的に実行されます。

- 指定した情報の検査。これには、ユーザーがシステムに存在することと、 HA-NFS が実行されていることを確認することが含まれます。
- db2nodes.cfg ファイルの作成。
- hadb2-eee.cfg ファイルの作成。
- EEE インスタンスの .rhosts ファイルの作成。
- デフォルトのデータベース・パスから関連する論理ホストのデータ・ディレクトリーへのシンボリック・パスの作成。
- hadb2tab ファイルへの行の追加。

構成エラーを防止し、 HA インスタンスを **hadb2_setup** コマンドの実行後に始動できるようにするために、コマンドは新規インスタンスが追加される前にかなりの量のテストを実行します。

db2nodes.cfg ファイルが作成され、現行のクラスターの状態に対応する情報が入れられます。たとえば、論理ホスト「log0」がマシン「crackle」によってホストされている場合、「log0」に関連したデータベース区画の項目には、マシン名「crackle」および「crackle」の高速内部接続が含まれます。

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 thrash 0 204.152.65.34
3 thrash 1 204.152.65.34
```

hadb2-eee.cfg ファイルは、コマンドで指定したノード情報のみに基づいて作成されます。データベース区画ごとに 1 つの行があります。

```
sphere % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

.rhost ファイルは DB2 UDB EEE で必要とされ、このファイルにはクラスター内の各マシンの全ホスト名 (または IP アドレス) が含まれていなければなりません。たとえば、以下の通りです。

```
crackle db2eee
204.152.65.1 db2eee
204.152.65.17 db2eee
thrash db2eee
204.152.65.2 db2eee
204.152.65.18 db2eee
crackle db2eee
jolt db2eee
bump db2eee
thrash.torolab.ibm.com db2eee
crackle.torolab.ibm.com db2eee
```

SMS 表スペースのファイル・システム・レイアウトに従い、 **hadb2_setup** コマンドは複数のディレクトリーやシンボリック・リンクをセットアップします。これには以下のものが含まれます。

- 各論理ホストの論理ホスト・ファイル・システムの下の「data」ディレクトリー。
- 論理ホストに関連した各データベース区画の (「data」ディレクトリーの下にある) ノード・ディレクトリー。
- デフォルトのデータベース・パスのシンボリック・リンクは、 `~<instance>` の下にあります。 *~instance* はインスタンスのホーム・ディレクトリーです。各データベース区画ごとに、対応するノード・ディレクトリーを指す 1 つのシンボリック・リンクがあります。詳しくは、274ページの『EE および EEE インスタンスのディスクのレイアウト』を参照してください。

フェイルオーバー時間

フェイルオーバー時間は、データが最初に使用不可になったときから再び使用可能になったときまでの時間により計測されます。フェイルオーバー中に以下のような複数のイベントが起きると、フェイルオーバー時間に大きく影響する可能性があります。

- ディスクのデポートとインポート。

  ディスクのデポートとインポートには、通常、他のイベントに比べて非常に長い時間がかかります。ただし、これは全体のダウン時間には影響しません。フェイルオーバー時に 1 つのマシンから別のマシンに移動する必要のあるディスクが多いほど、プロセスにかかる時間は長くなります。欠陥のあるディスクがある場合、プロセスはさらに長くなる可能性があります。

- 論理ホスト用にマウントされているファイル・システムの Fsck。

  論理ホストのファイル・システムをマウントする前に、 `fsck` を実行してファイル・システムの正常性を確認しなければなりません。ファイル・システムが大きいほど、このプロセスには時間がかかります。ジャーナル・ファイル・システムを使用すると、この時間を大幅に減少させることができます。ジャーナル・ファイル・システムは通常、HA 環境で使用されるので、 `fsck` 時間は問題になりません。

- HA エージェントから呼び出されるユーザー・スクリプト。

HA エージェントは、存在して実行可能なユーザー・スクリプトを呼び出します。これらのスクリプトのいくつかは同期的に実行され、 HA インスタンスを立ち上げるのにかかる時間を増す可能性があります。これらのスクリプトができるだけ素早く実行されるようにしてください。これらのスクリプトにより呼び出される外部プログラムをバックグラウンドで実行することを考慮できます。

- HA-NFS。

  単一の EEE インスタンスが相互引き受け構成にある場合、 HA-NFS をインスタンス所有者のホーム・ディレクトリーで使用しなければなりません。 HA-NFS は、 *lockd* (HA-NFS の HA エージェントで定義される) の猶予期間のために、フェイルオーバー時間を増します。 HA-NFS の実行時におけるこの時間は 90 秒です。フェイルオーバー後に HA-NFS ファイル・システム上でファイルをロックするプロセスが、猶予期間の終了を待たなければならないため、これはフェイルオーバー時間に影響を与えます。 DB2 の HA エージェントは、フェイルオーバー後にインスタンス所有者のホーム・ディレクトリーの下でファイルをロックする最初のプロセスであり、最初のロックを取得するのにかかる時間を記録します。この時間は、フェイルオーバー後の状況レポートに表示されます。

- DB2 の開始。

  DB2 の開始は、少ししかフェイルオーバー時間に影響しません。 EE インスタンスの場合、平均で約 5 〜 15 秒影響します。 EEE インスタンスの場合、約 10 秒に加え、フェイルオーバーされるデータベース区画ごとに約 5 秒影響します。たとえば、3 つのデータベース区画がフェイルオーバーされている場合、これら 3 つのデータベース区画の開始によって影響するフェイルオーバー時間は、約 25 秒です。これには、インスタンスのデータベースの破損回復は含まれません。

- データベースの破損回復。

  破損回復はしばしば、フェイルオーバーに関連するダウン時間の大部分に影響します。データベースの回復にかかる時間は、以下のいくつかの要因に依存します。

  – クライアントの作業負荷。データベースの変更だけがトランザクション・ログに記録されます。クライアントの作業負荷のほとんどが読み取り専用の操作である場合、破損回復時にデータベースに適用しなければならないトランザクションは比較的少なくてすみます。

  – ディスクとマシンの速度。ディスクおよび HA インスタンスをホストしているマシンの速度も、データベースを回復するためにかかる時間に貢献します。システムが速いほど、破損回復時間は短くなります。

  – *softmax* データベース構成パラメーターの値。 *softmax* の値は、ログ・ファイル・サイズのうち、ソフト・チェックポイントが取られ、ログ制御ファイルが書き込まれる部分のパーセンテージです。ログ制御ファイルは、破損回復時に使用され、データベースを一貫した状態に復元するために実際に必要なログ・レコードを判別します。この値を小さくすると、データベース・マネージャーがページ・クリーナーを起動する頻度が上がり、ソフト・チェックポイントを作成する頻度も上がります。パフォーマンスは落ちますが、データベースの回復は速くなります。

– インスタンスが EE と EEE のどちらであるか。インスタンスが EEE インスタンスである場合、データベースの再始動操作は、並列に実行されます。各データベース区画は、データベースの各部分の再始動を担当します。データベースに 50 GB のデータがある場合、 4 つのデータベース区画を持つインスタンスは、 1 つの EE インスタンスよりも約 4 倍速くデータベースを回復させることができます。

トラブルシューティング

以下の表には、起こり得る問題、その推定原因、および解決するために取ることのできるアクションが示されています。

表 16. Sun Cluster 2.2 での高可用性のトラブルシューティング

症状	考えられる原因	アクション
論理ホストのファイル・システムをマウントできない	論理ホストのファイル・システムは通常、論理ホストのフェイルオーバー時にマウントおよびアンマウントされます。フェイルオーバーのときに、論理ホストのファイル・システムの下に活動状態のプロセスまたはオープン・ファイルがあってはなりません。まれに、強制終了できないプロセスが、その現行作業ディレクトリーを論理ホストのファイル・システムの下に保持していることがあります。プロセスがマウント・ポイントの下にあるかどうかを知るには、`fuser(1m)` か、 GNU ユーティリティー `lsof` を使用します。論理ホストのファイル・システムをマウントできない場合、エラー・メッセージが出されます。[a]	システムをリブートするか、または論理ホストのファイル・システムを別の名前にして再作成してください。これを行うことにより、凍結したプロセスをディレクトリーの下に残すことができ (強制終了ができないため)、マウントができるようになります。 [b]

表 16. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
db2start または db2stop のタイムアウトが機能しない	SIGALRM シグナルがブロック化システム呼び出しから出されない場合があります。代わりに、SA_RESTART フラグが `sigaction()` を使用して設定されたかのように、このシステム呼び出しが再始動されます。これにより、DB2 HA エージェントのタイムアウトは無視され、エージェント・メソッドはハングし、ハングした **db2start** または **db2stop** コマンドから回復されません。	Solaris 2.6 用の必要なパッチ 105210-17 (またはそれ以降) を適用してください。
インスタンスにログインするとハングする	これが起きる理由はたくさんありますが、最も一般的な理由には、 NFS の問題および `/usr/sbin/quota` プログラムが関係します。	NFS マウントを検査してそれが良好な状態であることを確認してください。また、インスタンス所有者が所有する割り当て量のプロセスも確認してください。システム管理者の判断により、割り当て量プログラムを `/bin/true` へのシンボリック・リンクに変更すると、この問題を解決できる可能性があります。ただし、これは推奨されている解決方法ではありません。
EEE インスタンスをセットアップしたが、始動しない	**hadb2_setup** コマンドが、ポートを `/etc/services` ファイルに追加しません。管理者が手作業で追加すると予期されています。エラー・メッセージが戻されます。[c]	`/etc/services` ファイルで適切なポートを指定したことを確認してください。

表 16. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
START_NET メソッドが DB2 を開始できない		障害モニターをオフにして、インスタンスがフェイルオーバーされていないことを確認してください。インスタンス所有者としてログインして、 DB2 を手作業で開始してください。 1. `hadb2tab` 構成ファイルで適切なインスタンス・タイプが指定されていることを確認します。たとえば、EE 管理インスタンス用の `db2nodes.cfg` ファイルがあると、問題が起こります。 HA エージェント・メソッドは、この問題から回復することができません。 2. `.rhosts` ファイルが存在し、その中に有効な項目があることを確認します。 3. HA-NFS ファイル・システムが、クラスター内の全マシンのルート許可によって共用されていることを確認します。 4. カーネル・パラメーターを検査し、正しいことを確認します。 5. `/etc/services` ファイルにインスタンスの項目が含まれることを確認します。

表 16. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
インスタンスが 1 つのマシンでしか機能しない	• インスタンスの *uid* 数値が、クラスター内の各マシンで同じでない可能性があります。 • カーネル・パラメーターが、クラスター内の各マシンで有効でない可能性があります。 • `hadb2tab` ファイルが、クラスター内の各マシンで同じでない可能性があります。 • 論理ホストの `vfstab` ファイルのような別の構成ファイルが、クラスター内の各マシンで同じでない可能性があります。	これらの原因のどれも当てはまらないと思われる場合、インスタンス所有者としてログインを試みて、 DB2 を手作業で開始してください。 EE インスタンスの場合、インスタンスをホストしている論理ホストが現行のマシンによってホストされていれば、これはうまくいくはずです。 EEE インスタンスの場合、データベース区画をホストできるクラスター内のどのマシンからも、これはうまくいくはずです。
su - <instance> -c ″db2start″ が機能しない	• インスタンスの `.profile` が、 **su** に適していない可能性があります。 • Bourne シェル (`/bin/sh`) に関する既知の問題があります。このシェルでは、**su** コマンドは手作業では機能しますが、 HA エージェントを介しては機能しません。	• ルートとして、このコマンドを手作業で実行して機能することを確かめてから、 HA エージェントを介して再試行してください。 • 必要であれば、 Korn シェル (`/bin/ksh`) に切り替えてください。
EEE インスタンスが開始しないのに、ホーム・ディレクトリーはマウントされている	HA-NFS ディレクトリーが「ルート」許可と一緒にクラスター内のマシンにエクスポートされなかった可能性があります。 DB2 と HA エージェントの両方で、適切に実行されるにはこれが必要です。	これをテストするには、インスタンス所有者のホーム・ディレクトリーの下に、 (ルートとして) ファイルを作成してください。

表 16. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
EEE インスタンスのディレクトリーにアクセスしようとすると、「不良な NFS ファイル・ハンドル」エラーが戻される	インスタンス所有者のホーム・ディレクトリーの下にプロセスがまだ残っている可能性があります。	インスタンス所有者のホーム・ディレクトリーをアンマウントし、 HA エージェントがそれを再マウントできるようにしてください。 HA エージェントが再マウントするのは、 `hadb2` サービスがオフになり、再びオンになった場合です (290ページの『hadb2_setup コマンド』の **hadb2_setup** コマンドにある `-s` スイッチの説明を参照してください)。

表 16. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
制御メソッドが SC2.2 を介して正常に実行されない	hadb2 サービスが Sun Cluster に登録されていないか、それがオンになっていない可能性があります。	制御メソッドがコマンド行から通常どおり実行されているように見える場合は、 SYSLOG ファイルを検査して、問題を説明するのに役立つエラー・メッセージを探してください。 hadb2 サービスが Sun Cluster ソフトウェアに登録されており、それがオンになっていることを確認してください。 手動によるメソッドの実行は、問題のデバッグに役立ちます。[d] 制御メソッドは、ルートとして実行し、適切なコマンド行引き数を渡す必要があります。論理ホストのリストがヌルの場合、引き数は "" として渡されます。ブランク・スペースの区切り文字がない二重引用符は、ブランクの引き数を表します。以下はその例です。 `hadb2_startnet log0,log1 "" 600` 最初の引き数 log0,log1 は、論理ホスト log0 および log1 が現行のマシンによってホストされていることを、hadb2_startnet メソッドに知らせます。 2 つ目の引き数はヌルで、これはクラスター内のほかのマシンでホストされている論理ホストが存在しない (すべてが現行のマシンにある) ことを、hadb2_startnet メソッドに知らせます。 3 つめの引き数は、 SC2.2 が 600 秒後にタイムアウトになることをメソッドに知らせます。

表 16. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
ユーザー・スクリプトが実行されない	ユーザー・スクリプトは、それが適切なディレクトリーにあり、実行可能である場合にのみ実行できます。	所有権と属性を検査してください。それでもスクリプトが実行できない場合は、 IBM サービスに連絡してください。実行できないスクリプトのディレクトリー・リストと、スクリプトを実行できなければならなかったフェイルオーバーまたはクラスター構成の SYSLOG 出力を転送してください。
/etc/syslog.conf で指定したファイルに情報を記録できない		`touch(1)` を使用して、`/etc/syslog.conf` ファイルで指定したファイルを作成してから、SYSLOG デーモンを再始動してください。

表 16. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション

[a] 論理ホストのファイル・システムをマウントできないときに出されるエラー・メッセージは、以下のようなものです。

```
Aug 17 11:14:01 rash ID[SUNWcluster.loghost.1170]: importing data1
Aug 17 11:14:06 rash ID[SUNWcluster.scnfs.3040]: mount -F ufs -o ""
     /dev/vx/dsk/data1/data1-stat /log1 failed.
Aug 17 11:14:07 rash ID[SUNWcluster.ccd.ccdd.5304]: error freeze cmd =
     /opt/SUNWcluster/bin/loghost_sync
CCDSYNC_POST_ADDU LOGHOST_CM:log1:rash /etc/opt/SUNWcluster/conf/ccd.database
     2 "0 1" 1 error code = 1
```

[b] たとえば、以下の通りです。

```
scadmin@rash(218)# ps -fe | egrep db2
db2ee 1984 1 0 0:01 <defunct>

Solution:

   scadmin@rash(229)# cd /
   scadmin@rash(230)# mv /log1 /log1.bkp
   scadmin@rash(231)# mkdir /log1
```

[c] エラー・メッセージは以下のようなものです。

```
SQL6030N START or STOP DATABASE MANAGER failed. Reason code "13".
```

[d] たとえば、 `hadb2_startnet` メソッドが `libdb2.so.1` を見付けられないが、 Sun Cluster を介して通常どおり実行される場合、エラーは報告されません。このメソッドを手作業で実行すると、以下のような結果が出されます。

```
scadmin@crackle(213)# hadb2_startnet '''log0,log1' 600
ld.so.1: hadb2_startnet: fatal: libdb2.so.1: open failed:
   No such file or directory
Killed
```

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。

必須パラメーターは、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。

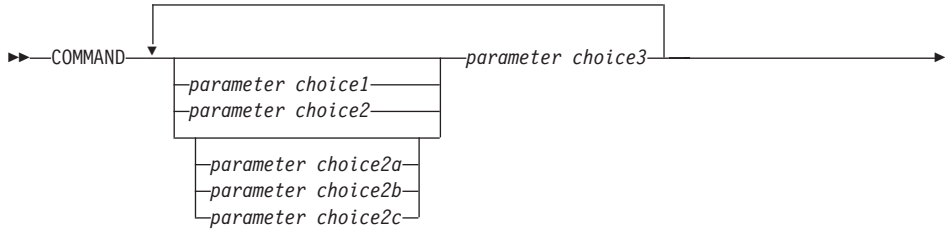
- 矢印が中断されていない場合、項目はブランク・スペースによって区切られたリストの形式で繰り返すことができます。

- 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。

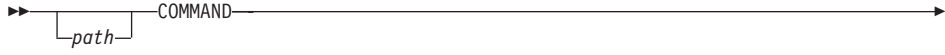
パラメーター・スタックの項目は、前述の必須およびオプション・パラメーターのスタック規則に従って繰り返すことができます。

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。

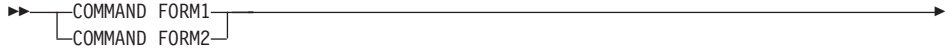


一部のコマンドの前には、オプションのパス・パラメーターが付けられています。



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、さまざまなバックアップおよび回復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。

すべての DB2 メッセージについては、*メッセージ解説書* を参照してください。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、適切な回復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

db2adutl - TSM アーカイブ・イメージによる作業

Tivoli Storage Manager (以前の ADSM) 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。

UNIX ベースのシステムでは、このユーティリティーは INSTHOME/sqllib/misc ディレクトリーにあります。 Windows オペレーティング・システムおよび OS/2 では、¥sqllib¥misc ディレクトリーにあります。

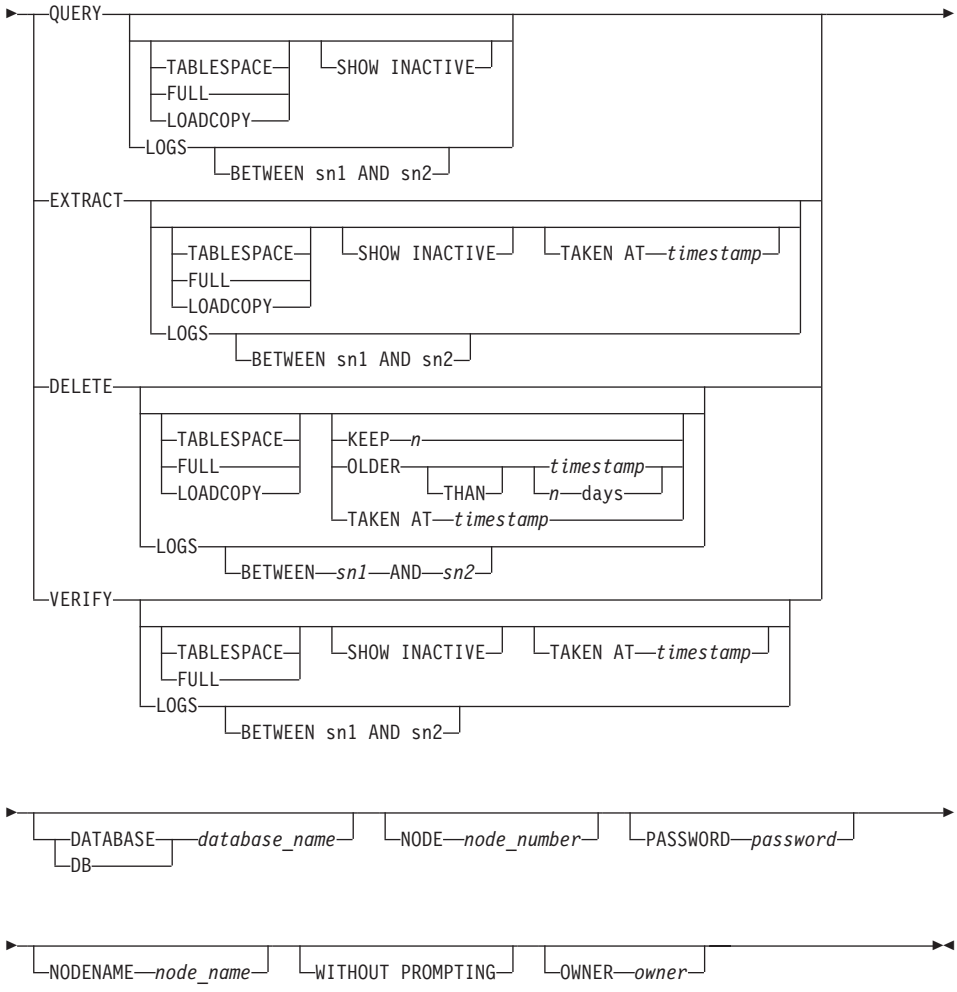
許可

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

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。

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 完全データベース・バックアップ・イメージだけを組み込みます。

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*
ログ順序番号 1 とログ順序番号 2 との間のログの使用を指定します。

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。

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* **or** *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。

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。

NODE *node_number*
指定したノード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 (要求される場合)。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。

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。

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。

例

以下に示すのは、db2 backup database rawsampl use tsm の出力例です。

```
Backup successful. The timestamp for this backup is : 19970929130942

db2adutl query

Query for database RAWSAMPL

Retrieving full database backup information.
   full database backup image: 1, Time: 19970929130942,
                                  Oldest log: S0000053.LOG, Sessions used: 1
   full database backup image: 2, Time: 19970929142241,
                                  Oldest log: S0000054.LOG, Sessions used: 1

Retrieving table space backup information.
   table space backup image: 1, Time: 19970929094003,
                                Oldest log: S0000051.LOG, Sessions used: 1
   table space backup image: 2, Time: 19970929093043,
                                Oldest log: S0000050.LOG, Sessions used: 1
   table space backup image: 3, Time: 19970929105905,
                                Oldest log: S0000052.LOG, Sessions used: 1

Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

以下に示すのは、db2adutl delete full taken at 19950929130942 db rawsampl の出力例です。

```
Query for database RAWSAMPL

Retrieving full database backup information.  Please wait.

  full database backup image: RAWSAMPL.0.db26000.0.19970929130942.001

  Do you want to deactivate this backup image (Y/N)? y

    Are you sure (Y/N)? y

db2adutl query

Query for database RAWSAMPL

Retrieving full database backup information.
   full database backup image: 2, Time: 19950929142241,
                              Oldest log: S0000054.LOG, Sessions used: 1
```

```
Retrieving table space backup information.
   table space backup image: 1, Time: 19950929094003,
                          Oldest log: S0000051.LOG, Sessions used: 1
   table space backup image: 2, Time: 19950929093043,
                          Oldest log: S0000050.LOG, Sessions used: 1
   table space backup image: 3, Time: 19950929105905,
                          Oldest log: S0000052.LOG, Sessions used: 1

Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

db2ckbkp - バックアップの検査

このユーティリティーは、バックアップ・イメージの保全性をテストするため、またそのイメージが復元可能かどうかを判別するのに使用できます。また、バックアップ・ヘッダーに保管されているメタ・データを表示することもできます。

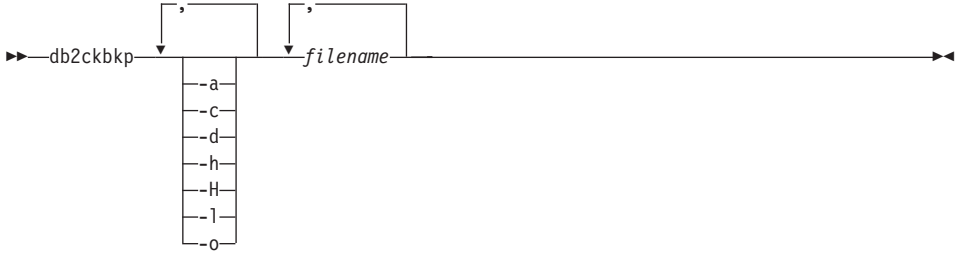
許可

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。

-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。

-h メディア・ヘッダー情報を表示します。これには、復元ユーティリティーで要求されるイメージの名前またはパスも含みます。

-H メディア・ヘッダー情報のみを表示します。

注:

1. このオプションは、イメージの妥当性検査は実行しません。このオプションを指定しない場合には、妥当性検査はイメージ全体に対して実行されます。
2. このオプションは、他のオプションと組み合わせた場合は無効です。

-l ログ・ファイル・ヘッダー・データを表示します。

-o オブジェクト・ヘッダーからの詳細情報を表示します。

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

```
db2ckbkp SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.*
[1] Buffers processed:  ##
[2] Buffers processed:  ##
[3] Buffers processed:  ##
Image Verification Complete - successful.

db2ckbkp -h SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001


=====================
MEDIA HEADER REACHED:
=====================
        Server Database Name           -- SAMPLE2
        Server Database Alias          -- SAMPLE2
        Client Database Alias          -- SAMPLE2
        Timestamp                      -- 19990818122909
        Node                           -- 0
        Instance                       -- krodger
        Sequence Number                -- 1
        Release ID                     -- 900
        Database Seed                  -- 65E0B395
        DB Comment's Codepage (Volume) -- 0
        DB Comment (Volume)            --
        DB Comment's Codepage (System) -- 0
        DB Comment (System)            --
        Authentication Value           -- 255
        Backup Mode                    -- 0
        Backup Type                    -- 0
        Backup Gran.                   -- 0
        Status Flags                   -- 11
        System Cats inc                -- 1
        Catalog Node Number            -- 0
        DB Codeset                     -- ISO8859-1
        DB Territory                   --
        Backup Buffer Size             -- 4194304
        Number of Sessions             -- 1
        Platform                       -- 0

The proper image file name would be:
```

```
SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001

[1] Buffers processed:  ####
Image Verification Complete - successful.
```

使用上の注意

複数のセッションを使用してバックアップ・イメージ作成した場合には、 **db2ckbkp** は同時にすべてのファイルを検査できます。順序番号 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。

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 (可変ブロック・サイズで作成されたイメージを除く) の検査も行うことができます。これは、復元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 NT では以下のようにします。

```
db2ckbkp -d ¥¥.¥tape1
```

バックアップ・イメージが TSM に常駐する場合には、314ページの『db2adutl - TSM アーカイブ・イメージによる作業』を参照してください。

db2ckrst - 増分復元イメージ・シーケンスの検査

データベース・ヒストリーを照会して、増分復元操作に必要な、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。手操作の増分復元に使用する、単純化された復元構文も生成されます。

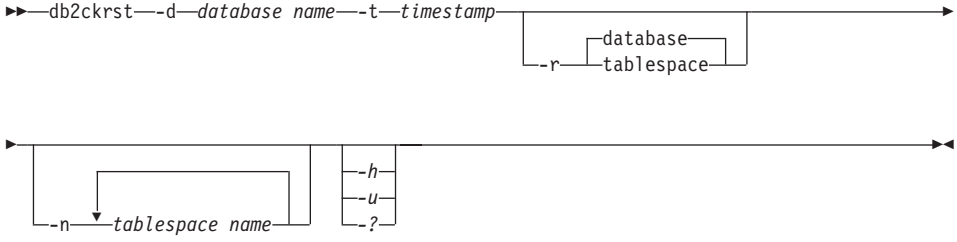
許可

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 namefile-name
復元されるデータベースの別名を指定します。

-t timestamp
増分復元されるバックアップ・イメージのタイム・スタンプを指定します。

-r 実行する復元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: 表スペースが選択されていて、表スペース名が指定されていない場合、このユーティリティーは指定されたイメージのヒストリー項目を調べて、復元を行うようリストされている表スペース名を使用します。

-n tablespace name
復元される 1 つ以上の表スペースの名前を指定します。

注: データベース復元タイプが選択されていて、表スペース名のリストが指定されている場合、このユーティリティーは指定された表スペース名を使用して表スペース復元操作を継続します。

-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・ヒストリーが存在していなければなりません。データベース・ヒストリーがない場合、このユーティリティーを使用する前に、 RESTORE DATABASE 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、最新の、完全データベース・バックアップ・イメージからの回復に必要な項目が削除される可能性があります。 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。

バックアップ操作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使用するのがよいでしょう。

db2flsn - ログ順序番号の検出

指定されたログ順序番号 (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。

許可

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 データベースが回復可能
- -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 バイトのストリング。

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 の日付が最新の *logfilsiz* 値の日付よりも前の場合、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、回復可能データベースでのみ使用することができます。データベースが回復可能なのは、*logretain* を RECOVERY に設定、または *userexit* を ON に設定して構成されている場合です。

db2inidb - ミラーリングされたデータベースの初期化

分割ミラーリング環境でこのコマンドは、異なる目的のためにミラーリングされたデータベースを初期化するのに使用します。

許可

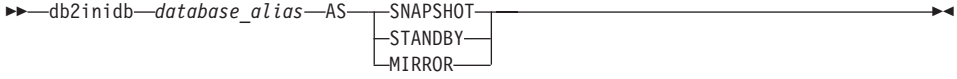
以下のどれかが必要です。

- *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
初期化するデータベースの別名を指定します。

SNAPSHOT
ミラーリングされたデータベースが、 1 次データベースの複製として初期化されるように指定します。このデータベースは読み取り専用です。

STANDBY
データベースがロールフォワード保留状態にされるように指定します。 1 次データベースからの新規ログを取り出し、待機データベースに適用することができます。こうすると、1 次データベースがダウンした場合、待機データベースをその代わりに使用することができます。

MIRROR
ミラーリングされたデータベースを、 1 次データベースを復元するのに使用できるバックアップ・イメージとして使用するように指定します。

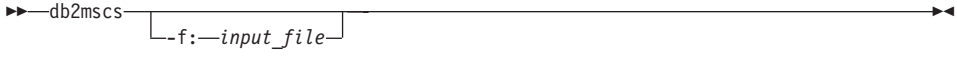
db2mscs - Windows NT フェイルオーバー・ユーティリティーのセットアップ

Microsoft Cluster Server (MSCS) を使用して、 Windows NT/2000 での DB2 フェイルオーバー・サポートの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、単一区画環境と区分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。

許可

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。

ARCHIVE LOG

回復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。

効力範囲

MPP 環境では、このコマンドはすべてのノード上のアクティブ・ログをクローズし、切り捨てます。ただし、ノードのサブセットを指定することもできます。

許可

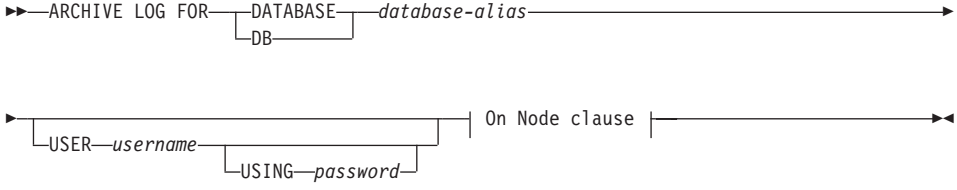
以下のどれかが必要です。

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

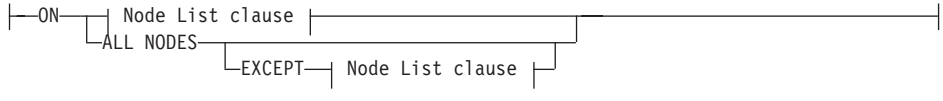
必要な接続

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。接続がすでに存在する場合、エラーが戻されます。

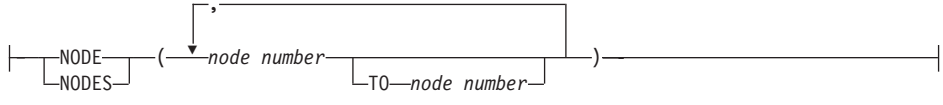
コマンド構文



On Node clause:



Node List clause:



コマンド・パラメーター

DATABASE database-alias
アクティブ・ログをアーカイブしたいデータベースの別名を指定します。

USER username
接続の試行に使用されるユーザー名を識別します。

USING password
ユーザー名の認証に使用されるパスワードを指定します。

ON ALL NODES
コマンドが db2nodes.cfg ファイル内のすべてのノード上で発行されるように指定します。ノード文節が指定されていない場合、これがデフォルトです。

EXCEPT
ノード・リストで指定されたノードを除き、コマンドが db2nodes.cfg ファイル内のすべてのノード上で発行されるように指定します。

ON NODE/ON NODES
あるノードのセット上の指定したデータベースのログがアーカイブされるように指定します。

node number
ノード・リスト内のノード番号を指定します。

TO node number
ログをアーカイブするノードの範囲を指定する際に使用します。指定された最初のノード番号から、指定された 2 番目のノード番号までのすべてのノードが、ノード・リストに含まれます。

使用上の注意

このコマンドを使用して、分かっている時点まで完全なセットのログ・ファイルを収集することができます。次に、そのログ・ファイルを使用して待機データベースを更新することができます。

このコマンドの起動時に、他のアプリケーションに進行中のトランザクションがある場合、ログ・バッファーがディスクにフラッシュされる際にパフォーマンスが若干減少します。ログ・レコードをバッファーに書き込もうとしている他のトランザクションが、フラッシュが完了するまで待機しなければならないためです。

このコマンドを実行すると、データベースの LSN スペースが一部失われます。そのため、有効な LSN が使用し尽くされるのが早くなります。

INITIALIZE TAPE

DB2 (Windows NT/2000 版) は、ストリーム・テープ装置へのバックアップおよび復元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。

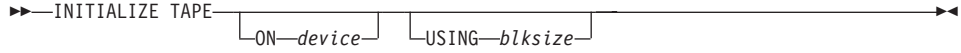
許可

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: 89ページの『BACKUP DATABASE コマンド』、および 116ページの『RESTORE DATABASE コマンド』 で指定するバッファー・サイズは、ここで指定するブロック・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。

参照

336ページの『REWIND TAPE』

337ページの『SET TAPE POSITION』

LIST HISTORY

ヒストリー・ファイルの中の項目のリストを表示します。ヒストリー・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・レベルの完全なバックアップ、復元、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、または名前変更、統計実行、表の再編成、表の除去、およびロードが含まれます。

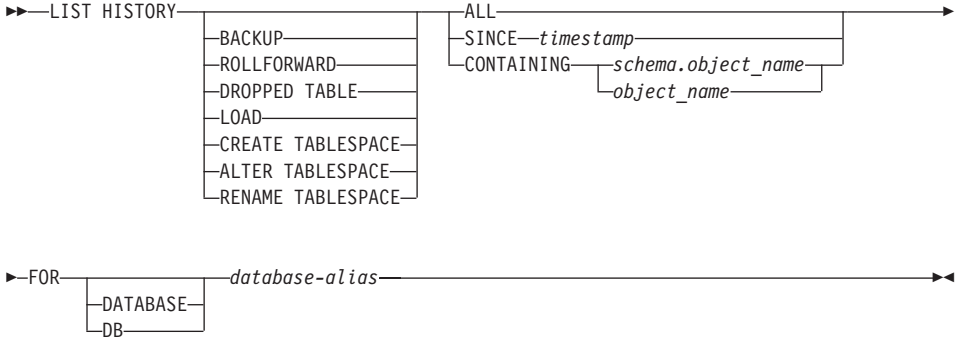
許可

なし

必要な接続

インスタンス。明示的な接続は必要ありません。データベースがリモートとして示されている場合、リモート・ノードへのインスタンス接続はコマンドの持続期間の間、ずっと確立されたままになります。

コマンド構文



コマンド・パラメーター

HISTORY
現在ヒストリー・ファイルの中に記録されているすべてのイベントのリストを表示します。

BACKUP
バックアップおよび復元操作のリストを表示します。

ROLLFORWARD
ロールフォワード操作のリストを表示します。

DROPPED TABLE
除去された表レコードのリストを表示します。

LOAD ロード操作のリストを表示します。

CREATE TABLESPACE
表スペースの作成および除去操作のリストを表示します。

RENAME TABLESPACE
表スペースの名前変更操作のリストを表示します。

ALTER TABLESPACE
表スペース変更操作のリストを表示します。

ALL ヒストリー・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 *yyyymmddhhnnss*)、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・スタンプのすべて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・ヒストリー・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 - Create table space
B - Backup
C - Load copy
D - Dropped table
F - Roll forward
G - Reorganize table
L - Load
N - Rename table space
O - Drop table space
Q - Quiesce
R - Restore
S - Run statistics
```

```
T - Alter table space
U - Unload
```

タイプ

バックアップのタイプ:

```
F - Offline
N - Online
I - Incremental offline
O - Incremental online
D - Delta offline
E - Delta online
```

ロールフォワードのタイプ:

```
E - End of logs
P - Point in time
```

ロードのタイプ:

```
I - Insert
R - Replace
```

表スペースの変更のタイプ:

```
C - Add containers
R - Rebalance
```

静止のタイプ:

```
S - Quiesce share
U - Quiesce update
X - Quiesce exclusive
Z - Quiesce reset
```

ロールフォワード操作がすべてのログの終わりまで実行された場合、バックアップ ID は最終時刻を表します。つまり、バックアップ ID 値は 99991231235959 になります。

PRUNE HISTORY/LOGFILE

リカバリー・ヒストリー・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。ファイルが大きくなりすぎたり、保存期間が長い場合は、リカバリー・ヒストリー・ファイルから項目を削除することが必要になる場合があります。ログを手操作で (ユーザー出口プログラムではなく) アーカイブする場合、アクティブ・ログ・ファイル・パスからログ・ファイルを削除することが必要な場合があります。

許可

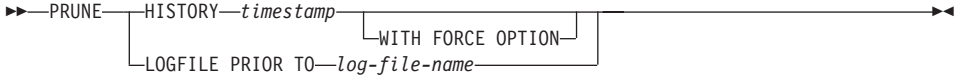
以下のどれかが必要です。

- *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・ヒストリー・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 (最低限、*yyyy*) を指定できます。指定されたタイム・スタンプ以前のタイム・スタンプを持つすべての項目が、リカバリー・ヒストリー・ファイルから削除されます。

WITH FORCE OPTION
最新の復元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枝取りすることを指定します。復元セットは、バックアップ・イメージのすべての復元を含む、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ヒストリーで保守されます。

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

1994 年 12 月 1 日以前 (その日を含む) に行われた、すべての復元、ロード、表スペース・バックアップ、および全データベース・バックアップのための項目をリカバリー・ヒストリー・ファイルから除去するには、次のように入力します。

```
db2 prune history 199412
```

注: 199412 は 19941201000000 と解釈されます。

使用上の注意

ヒストリー・ファイルからバックアップ項目を枝取りすると、 DB2 データ・リンク・マネージャー・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

DB2 (Windows NT/2000 版) は、ストリーム・テープ装置へのバックアップおよび復元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。

許可

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。

参照

330ページの『INITIALIZE TAPE』

337ページの『SET TAPE POSITION』

SET TAPE POSITION

DB2 (Windows NT/2000 版) は、ストリーム・テープ装置へのバックアップおよび復元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。

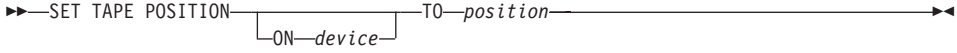
許可

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 NT/2000 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 が復元される位置に置かれます。

参照

330ページの『INITIALIZE TAPE』

336ページの『REWIND TAPE』

UPDATE HISTORY FILE

ヒストリー・ファイル項目にあるロケーション、装置タイプ、または注釈を更新します。

許可

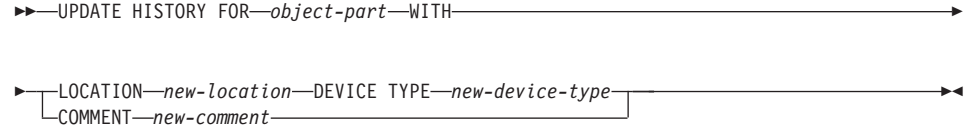
以下のどれかが必要です。

- *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

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 ～ 999 のオプションの順序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

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。

例

1997 年 4 月 13 日午前 10 時 00 分に取った全データベース・バックアップのヒストリー・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

ヒストリー・ファイルはデータベース管理者が記録保持のために使用します。また、DB2 が内部で増分バックアップの自動回復に使用します。

参照

334ページの『PRUNE HISTORY/LOGFILE』

付録D. 追加の API および関連データ構造

db2ArchiveLog - アクティブ・ログのアーカイブ API

回復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。

効力範囲

MPP 環境では、この API はすべてのノード上のアクティブ・ログをクローズし、切り捨てます。

許可

以下のどれかが必要です。

- *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。接続がすでに存在する場合、エラーが戻されます。

API 組み込みファイル

db2ApiDf.h

C API 構文

```
/* File: db2ApiDf.h */
/* API: Archive Active Log */
/* ... */
SQL_API_RC SQL_API_FN
   db2ArchiveLog (
    db2Uint32 version,
      void * pDB2ArchiveLogStruct,
      struct sqlca * pSqlca);

typedef struct
{
   char * piDatabaseAlias;
   char * piUserName;
   char * piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE * piNodeList;
   db2Uint32 iOptions;
} db2ArchiveLogStruct;
/* ... */
```

汎用 API 構文

```
/* File: db2ApiDf.h */
/* API: Archive Active Log */
/* ... */
SQL_API_RC SQL_API_FN
   db2gArchiveLog (
    db2Uint32 version,
      void * pDB2gArchiveLogStruct,
      struct sqlca * pSqlca);

typedef struct
{
   db2Uint32 iAliasLen;
   db2Uint32 iUserNameLen;
   db2Uint32 iPasswordLen;
   char * piDatabaseAlias;
   char * piUserName;
   char * piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE * piNodeList;
   db2Uint32 iOptions;
} db2gArchiveLogStruct;
/* ... */
```

API 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です。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

iAliasLen
入力。データベース別名の長さを示す 4 バイトの無符号整数 (バイト単位) です。

iUserNameLen
入力。ユーザー名の長さを示す 4 バイトの無符号整数 (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 バイトの無符号整数 (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
入力。 MPP 専用。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_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)。これがデフォルト値です。

DB2ARCHIVELOG_NODE_LIST
piNodeList で渡されたノード・リスト内で指定されたすべてのノードに適用されます。

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
入力。 MPP 専用。 *piNodeList* 配列内のノードの数を指定します。

piNodeList
入力。 MPP 専用。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。

使用上の注意

この API を使用して、分かっている時点まで完全なセットのログ・ファイルを収集することができます。次に、そのログ・ファイルを使用して待機データベースを更新することができます。

この API の呼び出し時に、他のアプリケーションに進行中のトランザクションがある場合、ログ・バッファーがディスクにフラッシュされる際にパフォーマンスが若干減少します。ログ・レコードをバッファーに書き込もうとしている他のトランザクションが、フラッシュが完了するまで待機しなければならないためです。

この API を使用すると、データベースの LSN スペースが一部失われます。そのため、有効な LSN が使用し尽くされるのが早くなります。

db2HistoryCloseScan - リカバリー・ヒストリー・ファイルの走査のクローズ API

リカバリー・ヒストリー・ファイルの走査を終了し、走査に必要だった DB2 リソースを解放します。この API は、352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』 を正常に呼び出した後でなければ使用できません。

許可

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: Close Recovery History File Scan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2HistoryCloseScan (
    db2Uint32 version,
    void * piHandle,
    struct sqlca * pSqlca);
/* ... */
```

汎用 API 構文

```
/* File: db2ApiDf.h */
/* API: Close Recovery History File Scan */
/* ... */
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。 352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』によって戻された、走査アクセス用のハンドルを指定します。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

REXX API 構文

```
CLOSE RECOVERY HISTORY FILE :scanid
```

REXX API パラメーター

scanid
OPEN RECOVERY HISTORY FILE SCAN から戻された走査 ID を含むホスト変数。

使用上の注意

リカバリー・ヒストリー・ファイル API の使用に関する詳細は、 352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』を参照してください。

参照

348ページの『db2HistoryGetEntry - リカバリー・ヒストリー・ファイルの次項目の入手 API』

352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』

362ページの『db2Prune API』

358ページの『db2HistoryUpdate - リカバリー・ヒストリー・ファイルの更新 API』

db2HistoryGetEntry - リカバリー・ヒストリー・ファイルの次項目の入手 API

リカバリー・ヒストリー・ファイルの次項目を入手します。この API は、352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』の正常呼び出しの後でなければ使用できません。

許可

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: Get Next Recovery History File Entry */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryGetEntry (
    db2Uint32 version,
    void * pDB2HistoryGetEntryStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2HistoryGetEntryStruct;
/* ... */
```

汎用 API 構文

```
/* File: db2ApiDf.h */
/* API: Get Next Recovery History File Entry */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryGetEntry (
    db2Uint32 version,
    void * pDB2GenHistoryGetEntryStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2GenHistoryGetEntryStruct;
/* ... */
```

API 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です。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

iHandle
入力。 352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』によって戻された、走査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です。この構造に関する詳細については、 369ページの『データ構造: db2HistData』を参照してください。

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 から戻された走査 ID を含むホスト変数。

value リカバリー・ヒストリー・ファイルの項目情報が戻される複合 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・スタンプおよび順序番号)
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復元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復元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復元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、 352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』への呼び出しで指定した値に基づいて選択されます。

リカバリー・ヒストリー・ファイル API の使用に関する詳細は、 352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』を参照してください。

参照

346ページの『db2HistoryCloseScan - リカバリー・ヒストリー・ファイルの走査のクローズ API』

352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』

362ページの『db2Prune API』

358ページの『db2HistoryUpdate - リカバリー・ヒストリー・ファイルの更新 API』

db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API

リカバリー・ヒストリー・ファイルの走査を開始します。

許可

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。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、リモート・ノードへのインスタン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: Open Recovery History File Scan */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryOpenScan (
    db2Uint32 version,
    void * pDB2HistoryOpenStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2HistoryOpenStruct;
/* ... */
```

汎用 API 構文

```
/* File: db2ApiDf.h */
/* API: Open Recovery History File Scan */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryOpenScan (
    db2Uint32 version,
    void * pDB2GenHistoryOpenStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2GenHistoryOpenStruct;
/* ... */
```

API 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です。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

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です。

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です。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ヌルに設定するか、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です。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

ラメーターをヌルに設定するか、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、一致したリカバリー・ヒストリー・ファイルの項目の数が入れられます。

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_LIST_HISTORY
現在ヒストリー・ファイルの中に記録されているすべてのイベントのリストを表示します。

DB2HISTORY_LIST_BACKUP
バックアップおよび復元操作のリストを表示します。

DB2HISTORY_LIST_ROLLFORWARD
ロールフォワード操作のリストを表示します。

DB2HISTORY_LIST_DROPPED_TABLE
除去された表レコードのリストを表示します。項目と関連する 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、 348ページの『db2HistoryGetEntry - リカバリー・ヒストリー・ファイルの次項目の入手 API』を呼び出す必要があります。

DB2HISTORY_LIST_LOAD
ロード操作のリストを表示します。

DB2HISTORY_LIST_CRT_TABLESPACE
表スペースの作成および除去操作のリストを表示します。

DB2HISTORY_LIST_REN_TABLESPACE
表スペースの名前変更操作のリストを表示します。

DB2HISTORY_LIST_ALT_TABLESPACE
表スペース変更操作のリストを表示します。項目と関連する 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、 348ページの『db2HistoryGetEntry - リカバリー・ヒストリー・ファイルの次項目の入手 API』を呼び出す必要があります。

DB2HISTORY_LIST_RUNSTATS
RUNSTATS 操作のリストを表示します。この値は、現在サポートされていません。

DB2HISTORY_LIST_REORG
REORGANIZE TABLE 操作のリストを表示します。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、走査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、348ページの『db2HistoryGetEntry - リカバリー・ヒストリー・ファイルの次項目の入手 API』、および 346ページの『db2HistoryCloseScan - リカバリー・ヒストリー・ファイルの走査のクローズ API』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
ヒストリー・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ヌルに設定すれば、*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ヌルに設定すれば、 *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 リカバリー・ヒストリー・ファイル情報が戻される複合 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 将来の走査アクセスに使用される ID (ハンドル)。

XXX.2 一致したリカバリー・ヒストリー・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。 (これがヒストリー・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、復元、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 つのヒストリー・ファイル走査が許可されています。

ヒストリー・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** を呼び出してヒストリー・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 の 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* 個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 の SQLCODE が戻された場合は、すべてのリカバリー・ヒストリー・ファイル項目の検索が済んだことを意味します。
   - 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、346ページの『db2HistoryCloseScan - リカバリー・ヒストリー・ファイルの走査のクローズ API』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する。

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。

参照

346ページの『db2HistoryCloseScan - リカバリー・ヒストリー・ファイルの走査のクローズ API』

348ページの『db2HistoryGetEntry - リカバリー・ヒストリー・ファイルの次項目の入手 API』

362ページの『db2Prune API』

358ページの『db2HistoryUpdate - リカバリー・ヒストリー・ファイルの更新 API』

db2HistoryUpdate - リカバリー・ヒストリー・ファイルの更新 API

ヒストリー・ファイル項目にあるロケーション、装置タイプ、または注釈を更新します。

許可

以下のどれかが必要です。

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

必要な接続

データベース。デフォルト・データベース以外のデータベースのヒストリー・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: Update Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryUpdate (
    db2Uint32 version,
    void * pDB2HistoryUpdateStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2HistoryUpdateStruct;
/* ... */
```

汎用 API 構文

```
/* File: db2ApiDf.h */
/* API: Update Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryUpdate (
    db2Uint32 version,
    void * pDB2GenHistoryUpdateStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2GenHistoryUpdateStruct;
/* ... */
```

API 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です。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

piNewLocation
入力。バックアップ、復元、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です。このパラメーターをヌルに設定するか、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、復元、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です。このパラメーターをヌルに設定するか、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です。このパラメーターをヌルに設定するか、ゼロを指すようにすれば、コメントは変更されません。

iEID 入力。ヒストリー・ファイルの特定の項目を更新するときに使用できる、固有の ID です。

REXX API 構文

```
UPDATE RECOVERY HISTORY USING :value
```

REXX API パラメーター

value リカバリー・ヒストリー・ファイル項目の新規ロケーションに関する情報を含む、複合 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・スタンプと順序番号 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。

ヒストリー・ファイルは、記録を保存する目的でのみ使用されます。復元またはロールフォワード関数によって、直接的に使用されることはありません。復元操作中は、バックアップ・イメージのロケーションを指定することができ、ヒストリー・ファイルはこのロケーションを追跡するのに役立ちます。このファイルの情報は、後でバックアップ・ユーティリティーに提供することができます (93ページの『Backup Database API』 を参照してください)。同様に、ロード・コピー・イメージのロケーションが移動される場合には、ロールフォワード・ユーティリティーに新規のロケーションとストレージ・メディアのタイプを提供する必要があります。詳細については、 156ページの『Rollforward Database API』 を参照してください。

参照

346ページの『db2HistoryCloseScan - リカバリー・ヒストリー・ファイルの走査のクローズ API』

348ページの『db2HistoryGetEntry - リカバリー・ヒストリー・ファイルの次項目の入手 API』

352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』

362ページの『db2Prune API』

db2Prune API

リカバリー・ヒストリー・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。

許可

以下のどれかが必要です。

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

必要な接続

データベース。デフォルト・データベース以外のデータベースのリカバリー・ヒストリー・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: Prune Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2Prune (
    db2Uint32 version,
    void * pDB2PruneStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2PruneStruct;
/* ... */
```

汎用 API 構文

```
/* File: db2ApiDf.h */
/* API: Prune Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenPrune (
    db2Uint32 version,
    void * pDB2GenPruneStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint32 iStringLen;
  char * piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2GenPruneStruct;
/* ... */
```

API 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です。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

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順序番号 (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。ヌル・パラメーターを指定してもデフォルト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・ログの枝取りが可能です。

iEID 入力。ヒストリー・ファイルから単一の項目の枝取りをするときに使用できる、固有の 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 で定義) は、以下のとおりです。

DB2PRUNE_ACTION_HISTORY
ヒストリー・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 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_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。提供されているそのタイム・スタンプ以下のタイム・スタンプを持つすべての項目が、リカバリー・ヒストリー・ファイルから削除されます。

WITH FORCE OPTION
指定した場合、最新の復元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・スタンプに従ってリカバリー・ヒストリー・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、最新の復元セットが保持されます。

使用上の注意

ヒストリー・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ストレージ・メディア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・バックアップをメディアから削除する (さらに、ヒストリー・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。

参照

346ページの『db2HistoryCloseScan - リカバリー・ヒストリー・ファイルの走査のクローズ API』

348ページの『db2HistoryGetEntry - リカバリー・ヒストリー・ファイルの次項目の入手 API』

352ページの『db2HistoryOpenScan - リカバリー・ヒストリー・ファイルの走査のオープン API』

358ページの『db2HistoryUpdate - リカバリー・ヒストリー・ファイルの更新 API』

sqlurlog - ログの非同期読み取り API

呼び出し側がデータベース・ログから特定のログ・レコードを抽出できるようにします。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。この API は、回復可能データベースでのみ使用することができます。データベースが回復可能なのは、*logretain* を RECOVERY に設定、または *userexit* を ON に設定して構成されている場合です。

許可

以下のどれかが必要です。

- *sysadm*
- *dbadm*

必要な接続

データベース

API 組み込みファイル

sqlutil.h

C API 構文

```
/* File: sqlutil.h */
/* API: Asynchronous Read Log */
/* ... */
SQL_API_RC SQL_API_FN
  sqlurlog (
   sqluint32 CallerAction,
    SQLU_LSN * pStartLsn,
    SQLU_LSN * pEndLsn,
    char * pLogBuffer,
    sqluint32 LogBufferSize,
    SQLU_RLOG_INFO * pReadLogInfo,
    struct sqlca * pSqlca);
/* ... */
```

API パラメーター

CallerAction
入力。実行するアクションを指定します。

SQLU_RLOG_READ
開始ログ順序番号から終了ログ順序番号までデータベース・ログを読み取り、この範囲内にある伝搬可能なログ・レコードをすべて戻します。

SQLU_RLOG_READ_SINGLE
開始ログ順序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。

SQLU_RLOG_QUERY
データベース・ログを照会します。照会した結果は、SQLU_RLOG_INFO 構造 (375ページの『データ構造: SQLU-RLOG-INFO』を参照) を介して戻されます。

pStartLsn
入力。開始ログ順序番号は、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。

pEndLsn
入力。終了ログ順序番号は、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。

pLogBuffer
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 バイトのログ順序番号 (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。

LogBufferSize
出力。バイト単位でログ・バッファーのサイズを指定します。

pReadLogInfo
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。この構造の詳細については、375ページの『データ構造: SQLU-RLOG-INFO』を参照してください。

pSqlca 出力。 *sqlca* 構造へのポインター。この構造の詳細については、*管理 API 解説書* または *SQL 解説書* を参照してください。

サンプル・プログラム

C ¥sqllib¥samples¥c¥asynrlog.sqc

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、呼び出し側はログ順序番号の範囲と、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 である表に関連したログ・レコードと、現在活動状態にあるログの情報が入った SQLU_RLOG_INFO 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、現在活動状態にあるログの情報が入った SQLU_RLOG_INFO 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 (SQLU-RLOG-INFO) 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。現在活動状態にあるログの終わりは、ログ情報の読み取り構造の curActiveLSN メンバーに入ります。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。

- curActiveLSN の値
- 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

ログ情報の読み取り構造の詳細については、 375ページの『データ構造: SQLU-RLOG-INFO』を参照してください。

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。 **sqlurlog** によって戻される各種の DB2 ログ・レコードについては、 *管理 API 解説書* に記載されています。

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、SQLU-RLOG-INFO で戻された、最後に読み取られたレコードの LSN に 1 を追加します。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実行依頼します。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、現在活動状態にあるログが最後まで読み取られたことを示します。

データ構造: db2HistData

この構造は、348ページの『db2HistoryGetEntry - リカバリー・ヒストリー・ファイルの次項目の入手 API』への呼び出しの後に情報を戻すのに使用されます。

表 17. 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)	ストレージ・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 文字は順序番号です。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。順序番号によって複数の位置を指定することができます。復元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順序番号 '001' に該当) だけが入ります。順序番号と結合されるタイム・スタンプは、固有の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 ～ S9999999)。 • オンライン・バックアップについてロールフォワード回復を適用するために必要 • オフライン・バックアップについてロールフォワード回復に適用するために必要 • ロードの開始時に現行だった全データベースまたは表スペース・レベル・バックアップの復元後に適用
oLastLog	db2Char	最新のログ・ファイル ID (範囲は S0000000 ～ S9999999)。 • オンライン・バックアップについてロールフォワード回復を適用するために必要 • オフライン・バックアップについて現時点へのロールフォワード回復を適用するために必要 • ロード操作の終了時に現行だった全データベースまたは表スペース・レベル・バックアップの復元後に適用 (ロールフォワード回復が適用されない場合は、*oFirstLog* と同じ)
oID	db2Char	固有のバックアップまたは表 ID。
oTableQualifier	db2Char	表修飾子。
oTableName	db2Char	表名。

表 17. 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
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順序番号は、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。復元およびロード操作の場合、ロケーションは、常に、復元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 O) 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	最新のログ順序番号。
oEID	構造体	固有の項目 ID。
poEventSQLCA	構造体	記録されたイベントの結果 *sqlca*。 *sqlca* 構造についての情報は、*管理 API 解説書* および *SQL 解説書* を参照してください。
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。各表スペース復元操作は 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復元を示している) 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復元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	371ページの表18を参照してください。
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	371ページの表19を参照してください。
oStatus	char	項目の状況。D (削除 (将来に使用))、 E (満了)、I (非アクティブ)、 N (コミットしていない)、および Y (コミット済みまたはアクティブ)。
oDeviceType	char	装置タイプ。このフィールドは、*oLocation* フィールドの解釈を判別します。 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 (他のベンダー装置のサポート))、 P (パイプ)、S (サーバー)、T (テープ)、および U (ユーザー出口)。

表 18. db2HistData 構造で有効な oOperation の値

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	除去された表	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	表スペースの除去	DB2HISTORY_OP_DROP_ TABLESPACE	DB2HIST_OP_DROP_ TABLESPACE
Q	静止	DB2HISTORY_OP_QUIESCE	DB2HIST_OP_QUIESCE
R	復元	DB2HISTORY_OP_RESTORE	DB2HIST_OP_RESTORE
S	統計の実行	DB2HISTORY_OP_RUNSTATS	DB2HIST_OP_RUNSTATS
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 19. db2HistData 構造で有効な oOptype の値

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	特定の時点	DB2HISTORY_OPTYPE_PIT
L	I	insert	DB2HISTORY_OPTYPE_INSERT
	R	置換	DB2HISTORY_OPTYPE_REPLACE
Q	S	静止共用	DB2HISTORY_OPTYPE_SHARE
	U	静止更新	DB2HISTORY_OPTYPE_UPDATE
	X	静止排他	DB2HISTORY_OPTYPE_EXCL
	Z	静止リセット	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

表 19. db2HistData 構造で有効な oOptype の値 (続き)

oOperation	oOptype	説明	C/COBOL/FORTRAN 定義
T	C	コンテナーの追加	DB2HISTORY_OPTYPE_ADD_CONT
	R	再調整	DB2HISTORY_OPTYPE_REB

表 20. 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。ヌルの場合、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 21. 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・ヒストリー・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;
/* ... */
```

データ構造: SQLU-LSN

366ページの『sqlurlog - ログの非同期読み取り API』 によって使用されるこの共用体には、ログ順序番号の定義が含まれます。ログ順序番号 (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 22. 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順序番号を指定します。
lsnWord	UNSIGNED SHORT の配列	3 メンバー文字配列のログ順序番号を指定します。

言語構文

C 構造

```
typedef union SQLU_LSN
{
unsigned char  lsnChar  [6] ;
unsigned short lsnWord  [3] ;
} SQLU_LSN;
```

データ構造: SQLU-RLOG-INFO

この構造には、366ページの『sqlurlog - ログの非同期読み取り API』 への呼び出しの状況、およびデータベース・ログについての情報が含まれています。

表 23. SQLU-RLOG-INFO 構造のフィールド

フィールド名	データ・タイプ	説明
initialLSN	SQLU_LSN	最初のデータベース CONNECT ステートメントが発行された後で書き込まれた最初のログ・レコードの LSN 値を指定します。詳細については、374ページの『データ構造: SQLU-LSN』 を参照してください。
firstReadLSN	SQLU_LSN	最初に読み取るログ・レコードの LSN 値を指定します。
lastReadLSN	SQLU_LSN	最後に読み取るログ・レコードの LSN 値を指定します。
curActiveLSN	SQLU_LSN	現行の (アクティブ・) ログの LSN 値を指定します。
logRecsWritten	sqluint32	バッファーに書き込まれるログ・レコードの数を指定します。
logBytesWritten	sqluint32	バッファーに書き込まれるバイト数を指定します。

言語構文

C 構造

```
typedef SQL_STRUCTURE SQLU_RLOG_INFO
{
SQLU_LSN       initialLSN ;
SQLU_LSN       firstReadLSN ;
SQLU_LSN       lastReadLSN ;
SQLU_LSN       curActiveLSN ;
sqluint32      logRecsWritten ;
sqluint32      logBytesWritten ;
} SQLU_RLOG_INFO;
```

付録E. 回復サンプル・プログラム

組み込み SQL のないサンプル・プログラム (backrest.c)

下記のサンプル・プログラムは、DB2 バックアップおよび回復 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。

- データベースのバックアップ
- データベースの復元
- データベースのロールフォワード回復

SAMPLE データベースについては、*SQL 解説書* を参照してください。

このサンプル・プログラムのソース・ファイル (backrest.c) は、 Windows オペレーティング・システムおよび OS/2 では ¥sqllib¥samples¥c ディレクトリーにあり、 UNIX ベースのシステムでは /sqllib/samples/c ディレクトリーにあります。それにはいくつかの DB2 API が含まれています。同じディレクトリーにある makefile には、このサンプル・プログラムや他のサンプル・プログラムを構築するためのコマンドが含まれています。 DB2 管理用 API を含むアプリケーションの作成についての一般情報、およびコンパイルとリンクのオプションについては、*アプリケーション構築の手引き* を参照してください。 Windows NT または Windows 2000 で、ソース・ファイル backrest.c からサンプル・プログラム backrest を構築するには、以下のようにします。

1. ファイル backrest.c、makefile、 utilapi.c、および utilapi.h を作業ディレクトリーにコピーします。
2. データベース・マネージャーが稼働していない場合、 DB2 コマンド・ウィンドウから **db2start** コマンドを発行します。 CLP 対応の DB2 ウィンドウをオープンし、Windows オペレーティング・システム上で DB2 コマンド行環境を初期化するには、コマンド・プロンプトから **db2cmd** を発行します。
3. nmake backrest と入力します。以下のファイルが生成されます。

```
backrest.exe
backrest.ilk
backrest.obj
backrest.pdb
utilapi.obj
```

サンプル・プログラム (実行可能ファイル) を実行するには、以下を入力します。

```
backrest database userID password
```

たとえば、以下のようになります。

```
backrest sample db2admin db2admin
```

組み込み SQL のないサンプル・プログラム (backrest.c)

以下は、このプログラムによって戻される出力の例です。

```
This is sample program : backrest.c

NOTE:   Ensure the database is not in use prior to running this program.

Updating the sample database configuration parameter LOGRETAIN
to 'ON' to enable rollforward recovery.

Backing up the 'sample' database.
The database has been successfully backed up.

Updating the sample database configuration parameter LOGRETAIN
to 'OFF'.

Restoring the database 'sample' as 'TESTBACK' (1st pass).
Should get returned value = SQLUD_INACCESSABLE_CONTAINER.
SQLUD_INACCESSABLE_CONTAINER is returned.

Need to SET TABLESPACES CONTAINERS
Tablespace container information for tablespace 1 obtained.
Tablespace containers have been set for tablespace 1.

Restoring the database 'sample' as 'TESTBACK' (2nd pass).
Database sample has been restored as 'TESTBACK'.

The database 'TESTBACK' has been successfully rolled forward.

The database 'TESTBACK' has been successfully dropped.
```

以下はサンプル・プログラムのソース・リストです。

```
/*****************************************************************************
**
** Source File Name = backrest.c
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1995, 2000
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**    PURPOSE :
**       an example showing how to use BACKUP & RESTORE APIs in order to:
**          - backup a database
**          - restore a database
**          - roll forward database to return the database to a state it was
**            in prior to the damage...
**
**    APIs USED :
**       BACKUP DATABASE                  sqlubkp()
**       RESTORE DATABASE                 sqlurestore()
**       UPDATE DATABASE CONFIGURATION    sqlfudb()
**       GET TABLESPACE CONTAINER QUERY   sqlbtcq()
```

```
**       SET TABLESPACE CONTAINERS         sqlbstsc()
**       ROLLFORWARD DATABASE              sqluroll()
**
**    STRUCTURES USED :
**       sqlu_tablespace_bkrst_list
**       SQLB_TBSCONTQRY_DATA
**       sqlu_media_list
**       rfwd_input
**       rfwd_output
**       sqlurf_info
**       sqlca
**
**    OTHER FUNCTIONS DECLARED :
**       'C' COMPILER LIBRARY :
**          stdio.h  -  printf
**
**       internal :
**
**       external :
**          check_error :     Checks for SQLCODE error, and prints out any
**          [in UTIL.C]          related information available.
**                               This procedure is located in the UTIL.C file.
**
**    EXTERNAL DEPENDENCIES :
**       - Ensure existence of database for precompile purposes.
**       - Compile and link with the IBM Cset++ compiler (AIX and OS/2)
**         or the Microsoft Visual C++ compiler (Windows)
**         or the compiler supported on your platform.
**
** For more information about these samples see the README file.
**
** For more information on programming in C, see the:
**   -  "Programming in C and C++" section of the Application Development Guide
** For more information on building C applications, see the:
**   -  "Building C Applications" section of the Application Building Guide.
**
** For more information on the SQL language see the SQL Reference.
**
******************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlutil.h>
#include <sqlenv.h>
#include "utilapi.h"

/* You can change the following path to another directory for which */
/* you have permission. create the "backup" directory in the same path. */

   #define TEMPDIR "."

int main (int argc, char *argv[]) {

   /* Variables for the BACKUP API */
```

```
sqluint32 buff_size;
sqluint32 num_buff;
struct sqlu_tablespace_bkrst_list *tablespace_list;
struct SQLB_TBSCONTQRY_DATA *cont;
struct sqlca sqlca;
struct sqlu_media_entry media_entry;
struct sqlu_media_list media_list;
char applid[SQLU_APPLID_LEN+1];
char timestamp[SQLU_TIME_STAMP_LEN+1];
char *target_path;

sqluint32 tsc_id;
sqluint32 num_cont;

/* variables for the Update Database Configuration API */
struct sqlfupd itemList;
short log_retain;

/* variables for the ROLLFORWARD API */
struct rfwd_input rfwdInput;
char dbAlias[] = "TESTBACK";
char overFlowLogPath[SQL_PATH_SZ] = TEMPDIR;

struct rfwd_output rfwdOutput;
sqlint32   numReplies;
struct sqlurf_info nodeInfo;

printf ("¥nThis is sample program : backrest.c¥n¥n");

printf ("NOTE:   Ensure the database is not in use prior to running
 this program.¥n¥n");

if (argc != 4) {
   printf ("¥nUSAGE: backrest database userID password¥n¥n");
   return 1;
} /* endif */

/* Altering the default Database Configuration to enable rollforward
   recovery of the TESTBACK database. */

log_retain = 1;
itemList.token = SQLF_DBTN_LOG_RETAIN;

itemList.ptrvalue = (char *) &log_retain;


printf ("Updating the %s database configuration parameter
 LOGRETAIN ¥n", argv[1]);
printf ("to 'ON' to enable rollforward recovery.¥n¥n");

/*****************************************\
* UPDATE DATABASE CONFIGURATION API called *
\*****************************************/
sqlfudb(argv[1], 1, &itemList, &sqlca);
API_SQL_CHECK("updating the database configuration");
```

```
/* Initialize the BACKUP API variables */
 buff_size = 16;
 num_buff = 1;
 tablespace_list = NULL;
 media_list.media_type = 'L';
 media_list.sessions = 1;

 strcpy (media_entry.media_entry, TEMPDIR);

 media_list.target.media = &media_entry;
 printf ("Backing up the '%s' database.¥n", argv[1]);
 /*****************\
 * BACKUP DATABASE *
 \*****************/
 sqlubkp (argv[1],
           buff_size,
           SQLUB_OFFLINE,
           SQLUB_FULL,
           SQLUB_BACKUP,
           applid,
           timestamp,
           num_buff,
           tablespace_list,
           &media_list,
           argv[2],
           argv[3],
           NULL,
           0,
           NULL,
           1,
           NULL,
           NULL,
           NULL,
           &sqlca);

 API_SQL_CHECK("backing up the database");
 printf ("The database has been successfully backed up.¥n¥n");



 /* Altering the default Database Configuration to disable rollforward
    recovery of the TESTBACK database. */

 log_retain = 0;
 itemList.token = SQLF_DBTN_LOG_RETAIN;
 itemList.ptrvalue = (char *) &log_retain;


 printf ("Updating the %s database configuration parameter
  LOGRETAIN ¥n", argv[1]);
 printf ("to 'OFF'.¥n¥n");
```

```
/*****************************************\
* UPDATE DATABASE CONFIGURATION API called *
\*****************************************/
sqlfudb(argv[1], 1, &itemList, &sqlca);
API_SQL_CHECK("updating the database configuration");



/* Initialize the variables for the RESTORE API */
buff_size = 1024;
target_path = NULL;
printf ("Restoring the database '%s' as 'TESTBACK' (1st pass).¥n", argv[1]);
printf ("Should get returned value = SQLUD_INACCESSABLE_CONTAINER.¥n");
/******************\
* RESTORE DATABASE *
\******************/
sqlurestore(argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_RESTORE_STORDEF,
          applid,
          timestamp,
          target_path,
          num_buff,
          NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
          NULL,
          0,
          NULL,
          1,
          NULL,
          NULL,
          NULL,
          &sqlca);
if (sqlca.sqlcode != SQLUD_INACCESSABLE_CONTAINER)
   API_SQL_CHECK("restoring database");

printf ("SQLUD_INACCESSABLE_CONTAINER is returned.¥n¥n");
printf ("Need to SET TABLESPACES CONTAINERS¥n");
/* Set the containers structure to a new list of containers */
tsc_id = 1;
cont = (struct SQLB_TBSCONTQRY_DATA *) malloc
   (sizeof(struct SQLB_TBSCONTQRY_DATA));

/*******************************\
* GET TABLESPACE CONTAINER QUERY *
\*******************************/
```

```
sqlbtcq (&sqlca, tsc_id, &num_cont, &cont);
API_SQL_CHECK("GET TABLESPACE CONTAINER QUERY");
printf ("Tablespace container information for tablespace 1 obtained.¥n");
/***************************\
* SET TABLESPACE CONTAINERS *
\***************************/
sqlbstsc (&sqlca,
          SQLB_SET_CONT_INIT_STATE,
          tsc_id,
          num_cont,
          cont);
API_SQL_CHECK("SET TABLESPACE CONTAINERS");
printf ("Tablespace containers have been set for tablespace 1.¥n¥n");

printf ("Restoring the database '%s' as 'TESTBACK' (2nd pass).¥n", argv[1]);
/******************\
* RESTORE DATABASE *
\******************/
sqlurestore (argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_CONTINUE,
          applid,
          timestamp,
          target_path,
          num_buff,
             NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
          NULL,
          0,
          NULL,
          1,
          NULL,
          NULL,
          NULL,
          &sqlca);
API_SQL_CHECK("restoring database 2");
printf ("Database %s has been restored as 'TESTBACK'.¥n¥n", argv[1]);


/**********************\
* ROLLFORWARD DATABASE *
\**********************/
rfwdInput.version=SQLUM_RFWD_VERSION;
rfwdInput.pDbAlias=dbAlias;
rfwdInput.CallerAction=SQLUM_ROLLFWD;
rfwdInput.pStopTime=SQLUM_INFINITY_TIMESTAMP;
rfwdInput.pUserName=argv[2];
```

```
    rfwdInput.pPassword=argv[3];
    rfwdInput.pOverflowLogPath=overFlowLogPath;
    rfwdInput.NumChngLgOvrflw=0;
    rfwdInput.pChngLogOvrflw=NULL;
    rfwdInput.ConnectMode=SQLUM_OFFLINE;
    rfwdInput.pTablespaceList=NULL;
    rfwdInput.AllNodeFlag= SQLURF_ALL_NODES;
    rfwdInput.NumNodes=0;
    rfwdInput.pNodeList=NULL;
    rfwdInput.NumNodeInfo=1;
    rfwdInput.DlMode=0;
    rfwdInput.pReportFile=NULL;
    rfwdInput.pDroppedTblID=NULL;
    rfwdInput.pExportDir=NULL;


    rfwdOutput.pApplicationId=applid;
    rfwdOutput.pNumReplies=&numReplies;
    rfwdOutput.pNodeInfo=&nodeInfo;


    sqluroll( &rfwdInput,
              &rfwdOutput,
              &sqlca);
    API_SQL_CHECK("rolling database forward");
    printf ("The database 'TESTBACK' has been successfully rolled
     forward.¥n¥n", argv[1]);


    /***************\
    * DROP DATABASE *
    \***************/
    sqledrpd ("TESTBACK",
              &sqlca);
    API_SQL_CHECK("DROP DATABASE");
    printf ("The database 'TESTBACK' has been successfully dropped.¥n");


  return 0;
}
```

組み込み SQLのあるサンプル・プログラム (dbrecov.sqc)

下記のサンプル・プログラムは、DB2 バックアップおよび回復 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。

- データベースのバックアップ
- データベースの復元
- データベースのロールフォワード回復

SAMPLE データベースについては、*SQL 解説書* を参照してください。

注: dbrecov サンプル・ファイルは、DB2 バージョン 7.2 ではインストールされません。 Web からのダウンロードでのみ入手できます。

dbrecov サンプル・プログラムに必要なファイルをダウンロードするには、以下のようにします。

1. http://www.ibm.com/software/data/db2/udb/ad/v7/abg.html にアクセスします。
2. 『Recent Updates』 の下の、 『dbrecov sample files by platform』 をクリックします。
3. 表示されるページで、リンクを選択し、ご使用のプラットフォームに合ったファイルをダウンロードしてください。 Windows オペレーティング・システムおよび OS/2 用では、ファイルは ZIP 実行可能ファイルに含まれています。 UNIX ベースのシステム用では、ファイルは tar.gz ファイルに含まれています。

注意:
新規ユーティリティー・ファイルは既存のバージョンの DB2 UDB バージョン 7 ユーティリティー・ファイルを上書きするため、この実行可能ファイルは、既存の DB2 UDB バージョン 7 サンプル・ファイルがあるディレクトリーでは実行しないでください。新規ユーティリティー・ファイルは他のサンプル・プログラムとの互換性がありません。

このサンプル・プログラムのソース・ファイル (dbrecov.sqc) には、 DB2 API および組み込み SQL 呼び出しの両方が含まれています。 DB2 管理用 API を含むアプリケーションの作成についての一般情報、およびコンパイルとリンクのオプションについては、*アプリケーション構築の手引き* を参照してください。

以下は、Windows オペレーティング・システムで dbrecov プログラムを構築および実行する手順の説明です。プログラムを、サポートされている他のオペレーティング・システム上に構築するには、サンプル・ファイル内に含まれている README ファイルを参照してください。

1. ダウンロードした実行可能ファイル dbrecov_win.exe を、作業ディレクトリーで実行します。これにより以下のファイルが取り出されます。

```
dbrecov.sqc - dbrecov プログラム用の組み込み SQL ソース・ファイル
utilapi.c   - DB2 API プログラム用のエラー・チェック・ユーティリティー
utilapi.h   - utilapi.c 用のヘッダー・ファイル
utilemb.sqc - 組み込み SQL プログラム用のエラー・チェック・ユーティリティー・ファイル
utilemb.h   - utilemb.sqc 用のヘッダー・ファイル
bldmapp.bat - Microsoft Visual C++ アプリケーション・プログラムのビルド
bldvapp.bat - VisualAge C++ アプリケーション・プログラムのビルド
embprep.bat - 組み込み SQL プログラムのプリコンパイルとバインド
README      - プログラム・ファイルについての情報が記載されています
```

2. データベース・マネージャーが稼働していない場合、 DB2 コマンド・ウィンドウから **db2start** コマンドを発行します。 CLP 対応の DB2 ウィンドウをオープンし、Windows オペレーティング・システム上で DB2 コマンド行環境を初期化するには、コマンド・プロンプトから **db2cmd** を発行します。

3. 使用しているコンパイラーに合わせて、bldmapp dbrecov、または bldvapp dbrecov と入力します。以下のファイルが生成されます。

```
dbrecov.exe
dbrecov.ilk
dbrecov.c
dbrecov.obj
dbrecov.bnd
dbrecov.pdb
utilemb.c
utilemb.obj
```

サンプル・プログラム (実行可能ファイル) を実行するには、以下を入力します。

```
dbrecov
```

出力は、ご使用のオペレーティング・システム環境によって異なります。以下は、Windows システム上でこのプログラムによって戻される出力の例です。

```
THIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.

USE THE DB2 API:
  sqlfxdb -- Get Database Configuration
TO GET THE DATABASE CONFIGURATION AND DETERMINE
THE SERVER WORKING PATH.

NOTE: Backup images will be created on the server
      in the directory D:¥DB2,
      and will not be deleted by the program.

**************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
**************************************

USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
  CONNECT
  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Prune the recovery history file for 'sample' database.

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.

**************************************
*** BACK UP AND RESTORE A DATABASE ***
**************************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
```

```
  sqlubkp -- Backup Database
  sqlurestore -- Restore Database
TO BACK UP AND RESTORE A DATABASE.

  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:¥DB2
    - backup image time stamp: 20010506162032

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162032
    - target database        : sample

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database restore -- start
line                = 506
file                = dbrecov.sqc
SQLCODE             = 2539

SQL2539W  Warning!  Restoring to an existing database that is the same as the
backup image database.  The database files will be deleted.

---- end warning report ----------------------

  Continuing the restore operation...

  Restore finished.

*************************
*** REDIRECTED RESTORE ***
*************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
  sqlbmtsq -- Tablespace Query
  sqlbtcq -- Tablespace Container Query
  sqlbstsc -- Set Tablespace Containers
  sqlefmem -- Free Memory
  sqledrpd -- Drop Database
TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.

  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO
```

```
  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:¥DB2
    - backup image time stamp: 20010506162125

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162125
    - target database        : RRDB

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database restore -- start
line                = 776
file                = dbrecov.sqc
SQLCODE             = 1277

SQL1277N  Restore has detected that one or more table space containers are
inaccessible, or has set their state to 'storage must be defined'.

---- end warning report ----------------------

  Continuing the restore operation...

  Find and redefine inaccessible containers.

    Redefine inaccessible container:
      - table space name: SYSCATSPACE
      - default container name: D:¥DB2¥NODE0000¥SQL00003¥SQLT0000.0
      - container type: path
      - new container name: D:¥DB2¥SQLT0000.0

    Redefine inaccessible container:
      - table space name: TEMPSPACE1
      - default container name: D:¥DB2¥NODE0000¥SQL00003¥SQLT0001.0
      - container type: path
      - new container name: D:¥DB2¥SQLT0001.0

    Redefine inaccessible container:
      - table space name: USERSPACE1
      - default container name: D:¥DB2¥NODE0000¥SQL00003¥SQLT0002.0
      - container type: path
      - new container name: D:¥DB2¥SQLT0002.0

  Restore finished.

  Drop the 'RRDB' database.

****************************
*** ROLLFORWARD RECOVERY ***
****************************
```

```
USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
  sqluroll -- Rollforward Database
  sqledrpd -- Drop Database
TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD.

  Update 'sample' database configuration:
    - Enable the configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:¥DB2
    - backup image time stamp: 20010506162223

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162223
    - target database        : RFDB

  Restore finished.

  Rolling 'RFDB' database forward ...
  Rollforward finished.

  Drop the 'RFDB' database.

*****************************
*** ASYNCHRONOUS READ LOG ***
*****************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlurlog -- Asynchronous Read Log
AND THE SQL STATEMENTS:
  CONNECT
  ALTER TABLE
  COMMIT
  INSERT
  DELETE
  ROLLBACK
  CONNECT RESET
TO READ LOG RECORDS FOR THE CURRENT CONNECTION.

  Update 'sample' database configuration:
    - Enable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES

  Backing up the 'sample' database...
  Backup finished.
```

```
    - backup image size      : 9 MB
    - backup image path      : D:¥DB2
    - backup image time stamp: 20010506162247

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Invoke the following SQL statements:
    ALTER TABLE emp_resume DATA CAPTURE CHANGES;
    COMMIT;
    INSERT INTO emp_resume
      VALUES('000777', 'ascii', 'This is a new resume.');
          ('777777', 'ascii', 'This is another new resume');
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000777';
    DELETE FROM emp_resume WHERE empno = '777777';
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000140';
    ROLLBACK;
    ALTER TABLE emp_resume DATA CAPTURE NONE;
    COMMIT;

  Start reading database log.

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database log info -- get
line               = 1457
file               = dbrecov.sqc
SQLCODE            = 2653

SQL2653W  A Restore, Forward or Crash Recovery may have reused log sequence
number ranges.  Reason code "1".

---- end warning report ----------------------

    Record type: Normal
      component ID: DMS log record
      function ID:  Alter Table Attribute
        Propagation attribute is changed to: ON

    Record type: Normal
      component ID: DMS log record
      function ID: Update Record
        oldRID: 2
        old subrecord length: 76
        old subrecord offset: 0
        subrecord type: Updatable, Internal control
        newRID: 2
        new subrecord length: 76
        new subrecord offset: 2916
        subrecord type: Updatable, Internal control

    Record type: Local pending list
```

```
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID: Insert Record
    RID: 12
    subrecord length: 88
    subrecord offset: 486
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 37 37 37 0F 00 05 00 *000777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 15 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 15 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Insert Record
    RID: 13
    subrecord length: 88
    subrecord offset: 398
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    37 37 37 37 37 37 0F 00 05 00 *777777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 1A 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 1A 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 01 00 00 00 *..........*

Record type: Normal commit
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 12
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 37 37 37 0F 00 05 00 *000777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 15 00 *I.........*
```

```
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 15 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 13
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    37 37 37 37 37 37 0F 00 05 00 *777777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 1A 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 1A 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 01 00 00 00 *..........*

Record type: Normal commit
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 6
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 05 00 *000140....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 24 05 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 01 3C 00 02 00 00 00 24 05 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 14 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 7
    subrecord length: 89
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
```

```
    30 30 30 31 34 30 0F 00 06 00 *000140....*
    15 00 3C 00 00 73 63 72 69 70 *.....scrip*
    74 49 06 01 00 00 00 00 00 56 *tI.......V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 01 3C 00 02 00 00 00 56 *.........V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 16 00 00 *..........*
    00                            *.         *

Record type: Compensation
  component ID: DMS log record
  function ID: Undo Delete Record
    RID: 7
    subrecord length: 89
    subrecord offset: 397
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 06 00 *000140....*
    15 00 3C 00 00 73 63 72 69 70 *.....scrip*
    74 49 06 01 00 00 00 00 00 56 *tI.......V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 01 3C 00 02 00 00 00 56 *.........V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 16 00 00 *..........*
    00                            *.         *

Record type: Compensation
  component ID: DMS log record
  function ID: Undo Delete Record
    RID: 6
    subrecord length: 88
    subrecord offset: 309
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 05 00 *000140....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 24 05 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 01 3C 00 02 00 00 00 24 05 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 14 00 00 00 *..........*

Record type: Normal abort
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID:  Alter Table Attribute
    Propagation attribute is changed to: OFF
```

```
    Record type: Local pending list
      UTC transaction committed (in seconds since 01/01/70): 989180591
      authorization ID of the application: MELNYK

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.

*******************************************
*** READ A DATABASE RECOVERY HISTORY FILE ***
*******************************************

USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryCloseScan -- Close Recovery History File Scan
TO READ A DATABASE RECOVERY HISTORY FILE.

  Open recovery history file for 'sample' database.

  Read entry number 0.

  Display entry number 0.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:¥DB2¥SAMPLE.0¥DB2¥NODE0000¥CATN0000¥20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 48
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Read entry number 1.

  Display entry number 1.
    object part: 20010506162058001
    end time: 200105061621
    first log: S0000000
    last log: S0000000
    ID: 20010506162032
    table qualifier:
```

```
  table name:
  location: D:¥DB2¥SAMPLE.0¥DB2¥NODE0000¥CATN0000¥20010506
  comment: DB2 RESTORE SAMPLE NO RF
  command text:
  history file entry ID: 49
  table spaces:
    SYSCATSPACE
    USERSPACE1
  type of operation: R
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:

Read entry number 2.

Display entry number 2.
  object part: 20010506162125001
  end time: 200105061622
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: D:¥DB2¥SAMPLE.0¥DB2¥NODE0000¥CATN0000¥20010506
  comment: DB2 BACKUP SAMPLE OFFLINE
  command text:
  history file entry ID: 50
  table spaces:
    SYSCATSPACE
    USERSPACE1
  type of operation: B
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:

Read entry number 3.

Display entry number 3.
  object part: 20010506162223001
  end time: 200105061622
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
```

```
    location: D:¥DB2¥SAMPLE.0¥DB2¥NODE0000¥CATN0000¥20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 51
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Read entry number 4.

  Display entry number 4.
    object part: 20010506162247001
    end time: 200105061623
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:¥DB2¥SAMPLE.0¥DB2¥NODE0000¥CATN0000¥20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 52
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Close recovery history file for 'sample' database.

******************************************************
*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***
******************************************************

USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryUpdate -- Update Recovery History File
  db2HistoryCloseScan -- Close Recovery History File Scan
```

```
TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.

  Open the recovery history file for 'sample' database.

  Read the first entry in the recovery history file.

  Display the first entry.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:¥DB2¥SAMPLE.0¥DB2¥NODE0000¥CATN0000¥20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 48
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Update the first entry in the history file:
    new location = 'this is the NEW LOCATION'
    new comment = 'this is the NEW COMMENT'

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.

  Close recovery history file for 'sample' database.

  Read the first recovery history file entry.

  Display the first entry.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: this is the NEW LOCATION
    comment: this is the NEW COMMENT
```

```
    command text:
    history file entry ID: 48
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Close the recovery history file for 'sample' database.

**************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
**************************************

USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
  CONNECT
  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Prune the recovery history file for 'sample' database.

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.
```

以下はサンプル・プログラムのソース・リストです。

```
/****************************************************************************
**
** Source File Name: dbrecov.sqc
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 2001
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**
**    PURPOSE:
**    This sample shows how to recover a database.
**
**    APIs USED:
```

```
**        db2HistoryCloseScan -- Close Recovery History File Scan
**        db2HistoryGetEntry -- Get Next Recovery History File Entry
**        db2HistoryOpenScan -- Open Recovery History File Scan
**        db2HistoryUpdate -- Update Recovery History File
**        db2Prune -- Prune Recovery History File
**        sqlbmtsq -- Tablespace Query
**        sqlbstsc -- Set Tablespace Containers
**        sqlbtcq -- Tablespace Container Query
**        sqledrpd -- Drop Database
**        sqlefmem -- Free Memory
**        sqlfudb -- Update Database Configuration
**        sqlfxdb -- Get Database Configuration
**        sqlubkp -- Backup Database
**        sqlurestore -- Restore Database
**        sqlurlog -- Asynchronous Read Log
**        sqluroll -- Rollforward Database
**
** For detailed information about database backup and recovery, see the
** "Data Recovery and High Availability Guide and Reference". This manual will
** help you to determine which database and table space recovery methods are
** best suited to your business environment.
**
** For more information about the sample programs, see the README file.
**
** For more information about programming in C, see the
** "Programming in C and C++" section of the "Application Development Guide".
**
** For more information about building C applications, see the
** section for your compiler in the "Building Applications" chapter
** for your platform in the "Application Building Guide".
**
** For more information about SQL, see the "SQL Reference".
**
** For more information about DB2 APIs, see the "Administrative API Reference".
**
** For the latest information on programming, compiling, and running DB2
** applications, visit the DB2 application development Web site at
**     http://www.software.ibm.com/data/db2/udb/ad
**
******************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlutil.h>
#include <db2ApiDf.h>
#include "utilemb.h"

int DbRecoveryHistoryFilePrune(char *, char *, char *);
int DbBackupAndRestore(char *, char *, char *, char *, char *);
int DbBackupAndRedirectedRestore(char *, char *, char *, char *, char *);
int DbBackupRestoreAndRollforward(char *, char *, char *, char *, char *);
int DbLogRecordsForCurrentConnectionRead(char *, char *, char *, char *);
int DbRecoveryHistoryFileRead(char *);
int DbFirstRecoveryHistoryFileEntryUpdate(char *, char *, char *);
```

```
/* support function called by the main() */
int ServerWorkingPathGet(char *, char *);

/* support function called by DbBackupAndRedirectedRestore() */
int InaccessableContainersRedefine(char *);

/* support function called by DbBackupAndRedirectedRestore() and
   DbBackupRestoreAndRollforward() */
int DbDrop(char *);

/* support function called by DbLogRecordsForCurrentConnectionRead() */
int LogBufferDisplay(char *, sqluint32);
int LogRecordDisplay(char *, sqluint32, sqluint16, sqluint16);
int SimpleLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32);
int ComplexLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32,
                            sqluint8, char *, sqluint32);
int LogSubRecordDisplay(char *, sqluint16);
int UserDataDisplay(char *, sqluint16);

/* support functions called by DbRecoveryHistoryFileRead() and
   DbFirstRecoveryHistoryFileEntryUpdate() */
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *);
int HistoryEntryDisplay(struct db2HistoryData);
int HistoryEntryDataFieldsFree(struct db2HistoryData *);

/* DbCreate will create a new database on the server with the server's
     code page.
   Use this function only if you want to restore a remote database.
   This support function is being called by DbBackupAndRedirectedRestore()
   and DbBackupRestoreAndRollforward().  */
int DbCreate(char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1];
  char serverWorkingPath[SQL_PATH_SZ + 1];
  char restoredDbAlias[SQL_ALIAS_SZ + 1];
  char redirectedRestoredDbAlias[SQL_ALIAS_SZ + 1];
  char rolledForwardDbAlias[SQL_ALIAS_SZ + 1];
  sqluint16 savedLogRetainValue;
  char dbAlias[SQL_ALIAS_SZ + 1];
  char user[USERID_SZ + 1];
  char pswd[PSWD_SZ + 1];

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  if (rc != 0)
  {
    return rc;
  }

  printf("¥nTHIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.¥n");
```

```
strcpy(restoredDbAlias, dbAlias);
strcpy(redirectedRestoredDbAlias, "RRDB");
strcpy(rolledForwardDbAlias, "RFDB");

/* attach to a local or remote instance */
rc = InstanceAttach(nodeName, user, pswd);
if (rc != 0)
{
  return rc;
}

printf("¥nUSE THE DB2 API:¥n");
printf("  sqlfxdb -- Get Database Configuration¥n");
printf("TO GET THE DATABASE CONFIGURATION AND DETERMINE¥n");
printf("THE SERVER WORKING PATH.¥n");

/* get the server working path */
rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
if (rc != 0)
{
  return rc;
}

printf("¥nNOTE: Backup images will be created on the server¥n");
printf("      in the directory %s,¥n", serverWorkingPath);
printf("      and will not be deleted by the program.¥n");

/* call the sample functions */
rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);

rc = DbBackupAndRestore(dbAlias,
                        restoredDbAlias,
                        user,
                        pswd,
                        serverWorkingPath);

rc = DbBackupAndRedirectedRestore(dbAlias,
                                  redirectedRestoredDbAlias,
                                  user,
                                  pswd,
                                  serverWorkingPath);

rc = DbBackupRestoreAndRollforward(dbAlias,
                                   rolledForwardDbAlias,
                                   user,
                                   pswd,
                                   serverWorkingPath);

rc = DbLogRecordsForCurrentConnectionRead(dbAlias,
                                          user,
                                          pswd,
                                          serverWorkingPath);

rc = DbRecoveryHistoryFileRead(dbAlias);
```

```
  rc = DbFirstRecoveryHistoryFileEntryUpdate(dbAlias, user, pswd);

  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);

  /* detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* end main */

int ServerWorkingPathGet(char dbAlias[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbConfigFields[1];
  char serverLogPath[SQL_PATH_SZ + 1];
  int len;

  /* get the server log path */
  /* SQLF_DBTN_LOGPATH is a token of the non-updatable database configuration
     parameter 'logpath'; it is used to get the server log path */
  dbConfigFields[0].token = SQLF_DBTN_LOGPATH;
  dbConfigFields[0].ptrvalue =
    (char *)malloc((SQL_PATH_SZ + 1) * sizeof(char));

  /* get database configuration */
  /* the API sqlfxdb returns the values of individual entries in a
     database configuration file */
  sqlfxdb(dbAlias, 1, &dbConfigFields[0], &sqlca);
  DB2_API_CHECK("server log path -- get");

  strcpy(serverLogPath, dbConfigFields[0].ptrvalue);
  free(dbConfigFields[0].ptrvalue);

  /* choose the server working path; if, for example, serverLogPath =
     "C:¥DB2¥NODE0001¥....", we'll keep "C:¥DB2" for the serverWorkingPath
     variable; backup images created in this sample will be placed under
     the 'serverWorkingPath' directory */
  len = (int)(strstr(serverLogPath, "NODE") - serverLogPath - 1);
  memcpy(serverWorkingPath, serverLogPath, len);
  serverWorkingPath[len] = '¥0';

  return 0;
} /* ServerWorkingPathGet */

int DbCreate(char existingDbAlias[], char newDbAlias[])
{
  struct sqlca sqlca;
  char dbName[SQL_DBNAME_SZ + 1];
  char dbLocalAlias[SQL_ALIAS_SZ + 1];
```

```
char dbPath[SQL_PATH_SZ + 1];
struct sqledbdesc dbDescriptor;
struct sqledbcountryinfo countryInfo;
struct sqlfupd dbConfigFields[2];

printf("¥n  Create '%s' empty database with the same code set as
 '%s' database.¥n",
       newDbAlias, existingDbAlias);

/* initialize dbConfigFields */
dbConfigFields[0].token = SQLF_DBTN_TERRITORY;
dbConfigFields[0].ptrvalue = (char *)malloc(10 * sizeof(char));
memset(dbConfigFields[0].ptrvalue, '¥0', 10);
dbConfigFields[1].token = SQLF_DBTN_CODESET;
dbConfigFields[1].ptrvalue = (char *)malloc(20 * sizeof(char));
memset(dbConfigFields[1].ptrvalue, '¥0', 20);

/* get two database configuration fields */
sqlfxdb(existingDbAlias, 2, &dbConfigFields[0], &sqlca);
DB2_API_CHECK("DB Config. -- Get");

/* create a new database */
strcpy(dbName, newDbAlias);
strcpy(dbLocalAlias, newDbAlias);
strcpy(dbPath, "");

strcpy(dbDescriptor.sqldbdid, SQLE_DBDESC_2);
dbDescriptor.sqldbccp = 0;
dbDescriptor.sqldbcss = SQL_CS_NONE;

strcpy(dbDescriptor.sqldbcmt, "");
dbDescriptor.sqldbsgp = 0;
dbDescriptor.sqldbnsg = 10;
dbDescriptor.sqltsext = -1;
dbDescriptor.sqlcatts = NULL;
dbDescriptor.sqlusrts = NULL;
dbDescriptor.sqltmpts = NULL;

strcpy(countryInfo.sqldbcodeset, (char *)dbConfigFields[1].ptrvalue);
strcpy(countryInfo.sqldblocale, (char *)dbConfigFields[0].ptrvalue);

/* create database */
sqlecrea(dbName,
         dbLocalAlias,
         dbPath,
         &dbDescriptor,
         &countryInfo,
         '¥0',
         NULL,
         &sqlca);
DB2_API_CHECK("Database -- Create");

/* free the allocated memory */
free(dbConfigFields[0].ptrvalue);
free(dbConfigFields[1].ptrvalue);
```

```
  return 0;
} /* DbCreate */

int DbDrop(char dbAlias[])
{
  struct sqlca sqlca;

  printf("¥n  Drop the '%s' database.¥n", dbAlias);

  /* drop and uncatalog the database */
  sqledrpd(dbAlias, &sqlca);
  DB2_API_CHECK("Database -- Drop");

  return 0;
} /* DbDrop */

int DbBackupAndRestore(char dbAlias[],
                       char restoredDbAlias[],
                       char user[],
                       char pswd[],
                       char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
  char *restoreTargetPath;
  sqluint32 restoreBuffersNb;
  struct sqlu_tablespace_bkrst_list restoreTablespaceList;
  struct sqlu_media_list restoreSourceMediaList;
  struct sqlu_media_entry restoreSourceMediaEntries[1];
  sqluint32 restoreParallelismDegree;

  printf("¥n************************************¥n");
```

```
printf("*** BACK UP AND RESTORE A DATABASE ***¥n");
printf("**************************************¥n");
printf("¥nUSE THE DB2 APIs:¥n");
printf("  sqlfudb -- Update Database Configuration¥n");
printf("  sqlubkp -- Backup Database¥n");
printf("  sqlurestore -- Restore Database¥n");
printf("TO BACK UP AND RESTORE A DATABASE.¥n");

printf("¥n  Update ¥'%s¥' database configuration:¥n", dbAlias);
printf("    - Disable the database configuration parameter LOGRETAIN¥n");
printf("        i.e., set LOGRETAIN = OFF/NO¥n");

/* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
   parameter 'logretain'; it is used to update the database configuration
   file */
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* disable the database configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_DISABLE;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");

/*******************************/
/*    BACK UP THE DATABASE    */
/*******************************/
printf("¥n  Backing up the '%s' database...¥n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
```

```
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */

  /* depending on the sqlca.sqlcode value, user action may be */
  /* required, such as mounting a new tape */

  printf("¥n  Continuing the backup operation...¥n");

  backupCallerAction = SQLUB_CONTINUE;

  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
          NULL,
          0,
          NULL,
          backupParallelismDegree,
          &backupImageSize,
          NULL,
          NULL,
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}

printf("  Backup finished.¥n");
printf("    - backup image size      : %d MB¥n", backupImageSize);
printf("    - backup image path      : %s¥n",
       backupTargetMediaEntries[0].media_entry);
```

```
printf("    - backup image time stamp: %s¥n",
       backupStartTimestamp);

/*****************************/
/*    RESTORE THE DATABASE    */
/*****************************/

restoreBufferSize = 1024; /*  1024 x 4KB */
rollforwardPendingState = SQLUD_NOROLLFWD;
datalinkMode = SQLUD_NODATALINK;
restoreType = SQLUD_FULL;
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* number of table spaces */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* number of elements in the target */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;

printf("¥n  Restoring a database ...¥n");
printf("    - source image alias     : %s¥n", dbAlias);
printf("    - source image time stamp: %s¥n", restoreTimestamp);
printf("    - target database        : %s¥n", restoredDbAlias);

restoreCallerAction = SQLUD_RESTORE;

/* The API sqlurestore is used to restore a database that has been backed
   up using the API sqlubkp. */
sqlurestore(dbAlias,
            restoredDbAlias,
            restoreBufferSize,
            rollforwardPendingState,
            datalinkMode,
            restoreType,
            restoreMode,
            restoreCallerAction,
            restoreAppId,
            restoreTimestamp,
            restoreTargetPath,
            restoreBuffersNb,
            NULL,
            &restoreTablespaceList,
            &restoreSourceMediaList,
            user,
            pswd,
            NULL,
            0,
            NULL,
            restoreParallelismDegree,
            NULL,
            NULL,
```

```
                NULL,
                &sqlca);
/*  DB2_API_CHECK("database restore -- start");  */
  EXPECTED_WARN_CHECK("database restore -- start");

  while (sqlca.sqlcode != 0)
  {
    /* continue the restore operation */
    printf("¥n  Continuing the restore operation...¥n");

    /* depending on the sqlca.sqlcode value, user action may be
       required, such as mounting a new tape */

    restoreCallerAction = SQLUD_CONTINUE;

    /* restore the database */
    sqlurestore(dbAlias,
                restoredDbAlias,
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
                NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
                NULL,
                0,
                NULL,
                restoreParallelismDegree,
                NULL,
                NULL,
                NULL,
                &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }

  printf("¥n  Restore finished.¥n");

  return 0;
} /* DbBackupAndRestore */

int DbBackupAndRedirectedRestore(char dbAlias[],
                                 char restoredDbAlias[],
                                 char user[],
                                 char pswd[],
                                 char serverWorkingPath[])
{
```

```
int rc = 0;
struct sqlca sqlca;
struct sqlfupd dbCfgParameters[1];
unsigned short logretain;
sqluint32 backupBufferSize;
sqluint32 backupMode;
sqluint32 backupType;
sqluint32 backupCallerAction;
char backupAppId[SQLU_APPLID_LEN + 1];
char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
sqluint32 backupBuffersNb;
struct sqlu_tablespace_bkrst_list backupTablespaceList;
struct sqlu_media_list backupTargetMediaList;
struct sqlu_media_entry backupTargetMediaEntries[1];
sqluint32 backupParallelismDegree;
sqluint32 backupImageSize;
sqluint32 restoreBufferSize;
sqluint32 rollforwardPendingState;
sqluint32 datalinkMode;
sqluint32 restoreType;
sqluint32 restoreMode;
sqluint32 restoreCallerAction;
char restoreAppId[SQLU_APPLID_LEN + 1];
char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
char *restoreTargetPath;
sqluint32 restoreBuffersNb;
struct sqlu_tablespace_bkrst_list restoreTablespaceList;
struct sqlu_media_list restoreSourceMediaList;
struct sqlu_media_entry restoreSourceMediaEntries[1];
sqluint32 restoreParallelismDegree;

printf("¥n**************************¥n");
printf("*** REDIRECTED RESTORE ***¥n");
printf("**************************¥n");
printf("¥nUSE THE DB2 APIs:¥n");
printf("  sqlfudb -- Update Database Configuration¥n");
printf("  sqlubkp -- Backup Database¥n");
printf("  sqlecrea -- Create Database¥n");
printf("  sqlurestore -- Restore Database¥n");
printf("  sqlbmtsq -- Tablespace Query¥n");
printf("  sqlbtcq -- Tablespace Container Query¥n");
printf("  sqlbstsc -- Set Tablespace Containers¥n");
printf("  sqlefmem -- Free Memory¥n");
printf("  sqledrpd -- Drop Database¥n");
printf("TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.¥n");

printf("¥n  Update ¥'%s¥' database configuration:¥n", dbAlias);
printf("    - Disable the database configuration parameter LOGRETAIN ¥n");
printf("        i.e., set LOGRETAIN = OFF/NO¥n");

/* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
   parameter 'logretain'; it is used to update the database configuration
   file */
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;
```

```
/* disable the database configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_DISABLE;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");

/*******************************/
/*    BACK UP THE DATABASE    */
/*******************************/
printf("¥n  Backing up the '%s' database...¥n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
```

```
  {
    /* continue the backup operation */

    /* depending on the sqlca.sqlcode value, user action may be
       required, such as mounting a new tape */

    printf("¥n  Continuing the backup operation...¥n");

    backupCallerAction = SQLUB_CONTINUE;

    /* back up the database */
    sqlubkp(dbAlias,
            backupBufferSize,
            backupMode,
            backupType,
            backupCallerAction,
            backupAppId,
            backupStartTimestamp,
            backupBuffersNb,
            &backupTablespaceList,
            &backupTargetMediaList,
            user,
            pswd,
            NULL,
            0,
            NULL,
            backupParallelismDegree,
            &backupImageSize,
            NULL,
            NULL,
            &sqlca);
    DB2_API_CHECK("Database -- Backup");
  }

  printf("  Backup finished.¥n");
  printf("    - backup image size      : %d MB¥n", backupImageSize);
  printf("    - backup image path      : %s¥n",
         backupTargetMediaEntries[0].media_entry);
  printf("    - backup image time stamp: %s¥n",
         backupStartTimestamp);

  /* To restore a remote database, you will first need to create an empty
     database if the client's code page is different from the server's
     code page.
     If this is the case, uncomment the call to DbCreate(). It will create
     an empty database on the server with the server's code page. */

/*
  rc = DbCreate(dbAlias, restoredDbAlias);
  if (rc != 0)
  {
  return rc;
  }
*/
```

```
/*****************************/
/*    RESTORE THE DATABASE    */
/*****************************/

restoreBufferSize = 1024; /*  1024 x 4KB */
rollforwardPendingState = SQLUD_NOROLLFWD;
datalinkMode = SQLUD_NODATALINK;
restoreType = SQLUD_FULL;
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* number of table spaces */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* number of elements in the target */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;

printf("¥n  Restoring a database ...¥n");
printf("    - source image alias     : %s¥n", dbAlias);
printf("    - source image time stamp: %s¥n", restoreTimestamp);
printf("    - target database        : %s¥n", restoredDbAlias);

restoreCallerAction = SQLUD_RESTORE_STORDEF;

/* The API sqlurestore is used to restore a database that has been backed
   up using the API sqlubkp. */
sqlurestore(dbAlias,
            restoredDbAlias,
            restoreBufferSize,
            rollforwardPendingState,
            datalinkMode,
            restoreType,
            restoreMode,
            restoreCallerAction,
            restoreAppId,
            restoreTimestamp,
            restoreTargetPath,
            restoreBuffersNb,
            NULL,
            &restoreTablespaceList,
            &restoreSourceMediaList,
            user,
            pswd,
            NULL,
            0,
            NULL,
            restoreParallelismDegree,
            NULL,
            NULL,
            NULL,
            &sqlca);
/*  DB2_API_CHECK("database restore -- start"); */
```

```
EXPECTED_WARN_CHECK("database restore -- start");

while (sqlca.sqlcode != 0)
{
  /* continue the restore operation */
  printf("¥n  Continuing the restore operation...¥n");

  /* depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape */

  if (sqlca.sqlcode == SQLUD_INACCESSABLE_CONTAINER)
  {
    /* redefine the table space container layout */
    printf("¥n  Find and redefine inaccessable containers.¥n");
    rc = InaccessableContainersRedefine(serverWorkingPath);
    if (rc != 0)
    {
      return rc;
    }
  }

  restoreCallerAction = SQLUD_CONTINUE;

  /* restore the database */
  sqlurestore(dbAlias,
              restoredDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
              restoreAppId,
              restoreTimestamp,
              restoreTargetPath,
              restoreBuffersNb,
              NULL,
              &restoreTablespaceList,
              &restoreSourceMediaList,
              user,
              pswd,
              NULL,
              0,
              NULL,
              restoreParallelismDegree,
              NULL,
              NULL,
              NULL,
              &sqlca);
  DB2_API_CHECK("database restore -- continue");
}

printf("¥n  Restore finished.¥n");

/* drop the restored database */
```

```
  rc = DbDrop(restoredDbAlias);

  return 0;
} /* DbBackupAndRedirectedRestore */

int InaccessableContainersRedefine(char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  sqluint32 numTablespaces;
  struct SQLB_TBSPQRY_DATA **ppTablespaces;
  sqluint32 numContainers;
  struct SQLB_TBSCONTQRY_DATA *pContainers;
  int tspNb;
  int contNb;
  char pathSep[2];

  /* The API sqlbmtsq provides a one-call interface to the table space query
     data. The query data for all table spaces in the database is returned
     in an array. */
  sqlbmtsq(&sqlca,
           &numTablespaces,
           &ppTablespaces,
           SQLB_RESERVED1,
           SQLB_RESERVED2);
  DB2_API_CHECK("tablespaces -- get");

  /* refedine the inaccessible containers */
  for (tspNb = 0; tspNb < numTablespaces; tspNb++)
  {
    /* The API sqlbtcq provides a one-call interface to the table space
       container query data. The query data for all the containers in a table
       space, or for all containers in all table spaces, is returned in an
       array. */
    sqlbtcq(&sqlca, ppTablespaces[tspNb]->id, &numContainers, &pContainers);
    DB2_API_CHECK("tablespace containers -- get");

    for (contNb = 0; contNb < numContainers; contNb++)
    {
      if (!pContainers[contNb].ok)
      {
        /* redefine inaccessible container */
        printf("¥n    Redefine inaccessible container:¥n");
        printf("      - table space name: %s¥n",
               ppTablespaces[tspNb]->name);
        printf("      - default container name: %s¥n",
               pContainers[contNb].name);
        if (strstr(pContainers[contNb].name, "/"))
        { /* UNIX */
          strcpy(pathSep, "/");
        }
        else
        { /* Intel */
          strcpy(pathSep, "¥¥");
        }
```

```
        switch (pContainers[contNb].contType)
        {
          case SQLB_CONT_PATH:
            printf("      - container type: path¥n");

            sprintf(pContainers[contNb].name, "%s%sSQLT%04d.%d",
                                              serverWorkingPath, pathSep,
                                              ppTablespaces[tspNb]->id,
                                              pContainers[contNb].id);
            printf("      - new container name: %s¥n",
             pContainers[contNb].name);
            break;
          case SQLB_CONT_DISK:
          case SQLB_CONT_FILE:
          default:
            printf("    Unknown container type.¥n");
            break;
        }
      }
    }

    /* The API sqlbstsc is used to set or redefine table space containers
       while performing a 'redirected' restore of the database. */
    sqlbstsc(&sqlca,
             SQLB_SET_CONT_FINAL_STATE,
             ppTablespaces[tspNb]->id,
             numContainers,
             pContainers);
    DB2_API_CHECK("tablespace containers -- redefine");

    /* The API sqlefmem is used here to free memory allocated by DB2 for use
       with the API sqlbtcq (Tablespace Container Query). */
    sqlefmem(&sqlca, pContainers);
    DB2_API_CHECK("tablespace containers memory -- free");
  }

  /* The API sqlefmem is used here to free memory allocated by DB2 for
     use with the API sqlbmtsq (Tablespace Query). */
  sqlefmem(&sqlca, ppTablespaces);
  DB2_API_CHECK("tablespaces memory -- free");

  return 0;
} /* InaccessibleContainersRedefine */

int DbBackupRestoreAndRollforward(char dbAlias[],
                                  char rolledForwardDbAlias[],
                                  char user[],
                                  char pswd[],
                                  char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
```

```
sqluint32 backupMode;
sqluint32 backupType;
sqluint32 backupCallerAction;
char backupAppId[SQLU_APPLID_LEN + 1];
char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
sqluint32 backupBuffersNb;
struct sqlu_tablespace_bkrst_list backupTablespaceList;
struct sqlu_media_list backupTargetMediaList;
struct sqlu_media_entry backupTargetMediaEntries[1];
sqluint32 backupParallelismDegree;
sqluint32 backupImageSize;
sqluint32 restoreBufferSize;
sqluint32 rollforwardPendingState;
sqluint32 datalinkMode;
sqluint32 restoreType;
sqluint32 restoreMode;
sqluint32 restoreCallerAction;
char restoreAppId[SQLU_APPLID_LEN + 1];
char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
char *restoreTargetPath;
sqluint32 restoreBuffersNb;
struct sqlu_tablespace_bkrst_list restoreTablespaceList;
struct sqlu_media_list restoreSourceMediaList;
struct sqlu_media_entry restoreSourceMediaEntries[1];
sqluint32 restoreParallelismDegree;
struct rfwd_input rfwdInput;
struct rfwd_output rfwdOutput;
char rollforwardAppId[SQLU_APPLID_LEN + 1];
sqlint32 numReplies;
struct sqlurf_info nodeInfo;

printf("¥n*****************************¥n");
printf("*** ROLLFORWARD RECOVERY ***¥n");
printf("*****************************¥n");
printf("¥nUSE THE DB2 APIs:¥n");
printf("  sqlfudb -- Update Database Configuration¥n");
printf("  sqlubkp -- Backup Database¥n");
printf("  sqlecrea -- Create Database¥n");
printf("  sqlurestore -- Restore Database¥n");
printf("  sqluroll -- Rollforward Database¥n");
printf("  sqledrpd -- Drop Database¥n");
printf("TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD. ¥n");

printf("¥n  Update ¥'%s¥' database configuration:¥n", dbAlias);
printf("    - Enable the configuration parameter LOGRETAIN ¥n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES¥n");


dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* enable the configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* The API sqlfudb is used to modify individual entries in a specific
```

```
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/* start the backup operation */
printf("¥n  Backing up the '%s' database...¥n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */
  printf("¥n  Continuing the backup operation...¥n");

  /* depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */
```

```
    backupCallerAction = SQLUB_CONTINUE;

    /* back up the database */
    sqlubkp(dbAlias,
            backupBufferSize,
            backupMode,
            backupType,
            backupCallerAction,
            backupAppId,
            backupStartTimestamp,
            backupBuffersNb,
            &backupTablespaceList,
            &backupTargetMediaList,
            user,
            pswd,
            NULL,
            0,
            NULL,
            backupParallelismDegree,
            &backupImageSize,
            NULL,
            NULL,
            &sqlca);
    DB2_API_CHECK("Database -- Backup");
  }

  printf("  Backup finished.¥n");
  printf("    - backup image size      : %d MB¥n", backupImageSize);
  printf("    - backup image path      : %s¥n",
         backupTargetMediaEntries[0].media_entry);
  printf("    - backup image time stamp: %s¥n",
         backupStartTimestamp);

  /* To restore a remote database, you will first need to create an empty
     database if the client's code page is different from the server's
     code page.
     If this is the case, uncomment the call to DbCreate(). It will create
     an empty database on the server with the server's code page. */

/*
  rc = DbCreate(dbAlias, rolledForwardDbAlias);
  if (rc != 0)
  {
    return rc;
  }
*/

  /*****************************/
  /*    RESTORE THE DATABASE    */
  /*****************************/

  restoreBufferSize = 1024; /*  1024 x 4KB */
  rollforwardPendingState = SQLUD_ROLLFWD;
  datalinkMode = SQLUD_NODATALINK;
  restoreType = SQLUD_FULL;
```

```
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* number of table spaces */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* number of elements in the target */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;

printf("¥n  Restoring a database ...¥n");
printf("    - source image alias     : %s¥n", dbAlias);
printf("    - source image time stamp: %s¥n", restoreTimestamp);
printf("    - target database        : %s¥n", rolledForwardDbAlias);

restoreCallerAction = SQLUD_RESTORE;

/* The API sqlurestore is used to restore a database that has been backed
   up using the API sqlubkp. */
sqlurestore(dbAlias,
            rolledForwardDbAlias,
            restoreBufferSize,
            rollforwardPendingState,
            datalinkMode,
            restoreType,
            restoreMode,
            restoreCallerAction,
            restoreAppId,
            restoreTimestamp,
            restoreTargetPath,
            restoreBuffersNb,
            NULL,
            &restoreTablespaceList,
            &restoreSourceMediaList,
            user,
            pswd,
            NULL,
            0,
            NULL,
            restoreParallelismDegree,
            NULL,
            NULL,
            NULL,
            &sqlca);
DB2_API_CHECK("database restore -- start");

while (sqlca.sqlcode != 0)
{
  /* continue the restore operation */
  printf("¥n  Continuing the restore operation...¥n");

  /* Depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */
```

```
  restoreCallerAction = SQLUD_CONTINUE;

  /* restore the database */
  sqlurestore(dbAlias,
              rolledForwardDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
              restoreAppId,
              restoreTimestamp,
              restoreTargetPath,
              restoreBuffersNb,
              NULL,
              &restoreTablespaceList,
              &restoreSourceMediaList,
              user,
              pswd,
              NULL,
              0,
              NULL,
              restoreParallelismDegree,
              NULL,
              NULL,
              NULL,
              &sqlca);
  DB2_API_CHECK("database restore -- continue");
}

printf("¥n  Restore finished.¥n");


/*****************************/
/*    ROLLFORWARD RECOVERY    */
/*****************************/

printf("¥n  Rolling '%s' database forward ...¥n", rolledForwardDbAlias);

rfwdInput.version = SQLUM_RFWD_VERSION;
rfwdInput.pDbAlias = rolledForwardDbAlias;
rfwdInput.CallerAction = SQLUM_ROLLFWD_STOP;
rfwdInput.pStopTime = SQLUM_INFINITY_TIMESTAMP;
rfwdInput.pUserName = user;
rfwdInput.pPassword = pswd;
rfwdInput.pOverflowLogPath = serverWorkingPath;
rfwdInput.NumChngLgOvrflw = 0;
rfwdInput.pChngLogOvrflw = NULL;
rfwdInput.ConnectMode = SQLUM_OFFLINE;
rfwdInput.pTablespaceList = NULL;
rfwdInput.AllNodeFlag = SQLURF_ALL_NODES;
rfwdInput.NumNodes = 0;
rfwdInput.pNodeList = NULL;
```

```
  rfwdInput.NumNodeInfo = 1;
  rfwdInput.DlMode = 0;
  rfwdInput.pReportFile = NULL;
  rfwdInput.pDroppedTblID = NULL;
  rfwdInput.pExportDir = NULL;

  rfwdOutput.pApplicationId = rollforwardAppId;
  rfwdOutput.pNumReplies = &numReplies;
  rfwdOutput.pNodeInfo = &nodeInfo;

  /* rollforward database */
  /* The API sqluroll rollforward recovers a database by
     applying transactions recorded in the database log files. */
  sqluroll(&rfwdInput, &rfwdOutput, &sqlca);
  DB2_API_CHECK("rollforward -- start");

  printf("  Rollforward finished.¥n");

  /* drop the restored database */
  rc = DbDrop(rolledForwardDbAlias);

  return 0;
} /* DbBackupRestoreAndRollforward */

int DbLogRecordsForCurrentConnectionRead(char dbAlias[],
                                         char user[],
                                         char pswd[],
                                         char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  SQLU_LSN startLSN;
  SQLU_LSN endLSN;
  char *logBuffer;
  sqluint32 logBufferSize;
  SQLU_RLOG_INFO readLogInfo;
  int i;

  printf("¥n*****************************¥n");
  printf("*** ASYNCHRONOUS READ LOG ***¥n");
  printf("*****************************¥n");
```

```
printf("¥nUSE THE DB2 APIs:¥n");
printf("  sqlfudb -- Update Database Configuration¥n");
printf("  sqlubkp -- Backup Database¥n");
printf("  sqlurlog -- Asynchronous Read Log¥n");
printf("AND THE SQL STATEMENTS:¥n");
printf("  CONNECT¥n");
printf("  ALTER TABLE¥n");
printf("  COMMIT¥n");
printf("  INSERT¥n");
printf("  DELETE¥n");
printf("  ROLLBACK¥n");
printf("  CONNECT RESET¥n");
printf("TO READ LOG RECORDS FOR THE CURRENT CONNECTION.¥n");

printf("¥n  Update ¥'%s¥' database configuration:¥n", dbAlias);
printf("    - Enable the database configuration parameter LOGRETAIN ¥n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES¥n");

dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* enable LOGRETAIN */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/* start the backup operation */
printf("¥n  Backing up the '%s' database...¥n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
```

```
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
        NULL,
        0,
        NULL,
        backupParallelismDegree,
        &backupImageSize,
        NULL,
        NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */
  printf("¥n  Continuing the backup operation...¥n");

  /* Depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */

  backupCallerAction = SQLUB_CONTINUE;

  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
          NULL,
          0,
          NULL,
          backupParallelismDegree,
          &backupImageSize,
          NULL,
          NULL,
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}

printf("  Backup finished.¥n");
printf("    - backup image size      : %d MB¥n", backupImageSize);
printf("    - backup image path      : %s¥n",
       backupTargetMediaEntries[0].media_entry);
```

```
printf("    - backup image time stamp: %s¥n",
       backupStartTimestamp);

/* connect to the database */
rc = DbConn(dbAlias, user, pswd);
if (rc != 0)
{
  return rc;
}

/* invoke SQL statements to fill database log */
printf("¥n  Invoke the following SQL statements:¥n"
       "    ALTER TABLE emp_resume DATA CAPTURE CHANGES;¥n"
       "    COMMIT;¥n"
       "    INSERT INTO emp_resume¥n"
       "      VALUES('000777', 'ascii', 'This is a new resume.');¥n"
       "             ('777777', 'ascii', 'This is another new resume');¥n"
       "    COMMIT;¥n"
       "    DELETE FROM emp_resume WHERE empno = '000777';¥n"
       "    DELETE FROM emp_resume WHERE empno = '777777';¥n"
       "    COMMIT;¥n"
       "    DELETE FROM emp_resume WHERE empno = '000140';¥n"
       "    ROLLBACK;¥n"
       "    ALTER TABLE emp_resume DATA CAPTURE NONE;¥n"
       "    COMMIT;¥n");

EXEC SQL ALTER TABLE emp_resume DATA CAPTURE CHANGES;
EMB_SQL_CHECK("SQL statement 1 -- invoke");

EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 2 -- invoke");

EXEC SQL INSERT INTO emp_resume
  VALUES('000777', 'ascii', 'This is a new resume.'),
        ('777777', 'ascii', 'This is another new resume');
EMB_SQL_CHECK("SQL statement 3 -- invoke");

EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 4 -- invoke");

EXEC SQL DELETE FROM emp_resume WHERE empno = '000777';
EMB_SQL_CHECK("SQL statement 5 -- invoke");

EXEC SQL DELETE FROM emp_resume WHERE empno = '777777';
EMB_SQL_CHECK("SQL statement 6 -- invoke");

EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 7 -- invoke");

EXEC SQL DELETE FROM emp_resume WHERE empno = '000140';
EMB_SQL_CHECK("SQL statement 8 -- invoke");

EXEC SQL ROLLBACK;
EMB_SQL_CHECK("SQL statement 9 -- invoke");
```

```
  EXEC SQL ALTER TABLE emp_resume DATA CAPTURE NONE;
  EMB_SQL_CHECK("SQL statement 10 -- invoke");

  EXEC SQL COMMIT;
  EMB_SQL_CHECK("SQL statement 11 -- invoke");

  printf("¥n  Start reading database log.¥n");

  logBuffer = NULL;
  logBufferSize = 0;

  /* The API sqlurlog (Asynchronours Read Log) is used to extract records
     from the database logs, and to query the log manager for current
     log state information.
     This API can only be used on recoverable databases. */

  /* Query the log manager for current log state information: */
  rc = sqlurlog(SQLU_RLOG_QUERY,
                &startLSN,
                &endLSN,
                logBuffer,
                logBufferSize,
                &readLogInfo,
                &sqlca);
/*  DB2_API_CHECK("database log info -- get"); */
  EXPECTED_WARN_CHECK("database log info -- get");

  logBufferSize = 64 * 1024;
  logBuffer = (char *)malloc(logBufferSize);

  memcpy(&startLSN, &(readLogInfo.initialLSN), sizeof(startLSN));
  memcpy(&endLSN, &(readLogInfo.curActiveLSN), sizeof(endLSN));

  /* Extract a log record from the database logs, and
     read the first log sequence asynchronously: */
  rc = sqlurlog(SQLU_RLOG_READ,
               &startLSN,
               &endLSN,
               logBuffer,
               logBufferSize,
               &readLogInfo,
               &sqlca);
  if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    DB2_API_CHECK("database logs -- read");
  }
  else
  {
    if (readLogInfo.logRecsWritten == 0)
    {
      printf("¥n  Database log empty.¥n");
    }
  }

  /* display log buffer */
```

```
  rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);

  while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    /* read the next log sequence */

    memcpy(&startLSN, &(readLogInfo.lastReadLSN), sizeof(startLSN));
    /* increase startLSN by 1 */
    startLSN.lsnChar[5] = startLSN.lsnChar[5] + 1;
    i = 5;
    while (startLSN.lsnChar[i] == 0 && i > 0)
    {
      startLSN.lsnChar[i - 1] = startLSN.lsnChar[i - 1] + 1;
      i = i - 1;
    }

    /* Extract a log record from the database logs, and
     read the next log sequence asynchronously: */
    rc = sqlurlog(SQLU_RLOG_READ,
                  &startLSN,
                  &endLSN,
                  logBuffer,
                  logBufferSize,
                  &readLogInfo,
                  &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
    {
      DB2_API_CHECK("database logs -- read");
    }

    /* display log buffer */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
  }

  /* free the log buffer */
  free(logBuffer);

  /* disconnect from the database */
  rc = DbDisconn(dbAlias);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* DbLogRecordsForCurrentConnectionRead */

int LogBufferDisplay(char *logBuffer, sqluint32 numLogRecords)
{
  int rc = 0;
  sqluint32 logRecordNb;
  sqluint32 recordSize;
  sqluint16 recordType;
  sqluint16 recordFlag;
  char *recordBuffer;
```

```
  /* initialize recordBuffer */
  recordBuffer = logBuffer + sizeof(SQLU_LSN);

  for (logRecordNb = 0; logRecordNb < numLogRecords; logRecordNb++)
  {
    recordSize = *(sqluint32 *)(recordBuffer);
    recordType = *(sqluint16 *)(recordBuffer + 4);
    recordFlag = *(sqluint16 *)(recordBuffer + 6);

    rc = LogRecordDisplay(recordBuffer, recordSize, recordType, recordFlag);
    /* update recordBuffer */
    recordBuffer = recordBuffer + recordSize + sizeof(SQLU_LSN);
  }

  return 0;
} /* LogBufferDisplay */

int LogRecordDisplay(char *recordBuffer,
                     sqluint32 recordSize,
                     sqluint16 recordType,
                     sqluint16 recordFlag)
{
  int rc = 0;
  sqluint32 logManagerLogRecordHeaderSize;
  char *recordDataBuffer;
  sqluint32 recordDataSize;
  char *recordHeaderBuffer;
  sqluint8 componentIdentifier;
  sqluint32 recordHeaderSize;

  /* determine logManagerLogRecordHeaderSize */
  if ((char)recordType == 'C')
  { /* compensation */
    if (recordFlag & 0x0002)
    { /* propagatable */
      logManagerLogRecordHeaderSize = 32;
    }
    else
    {
      logManagerLogRecordHeaderSize = 26;
    }
  }
  else
  { /* non compensation */
    logManagerLogRecordHeaderSize = 20;
  }

  switch ((char)recordType)
  {
    case 'E':
    case 'M':
    case 'A':
      recordDataBuffer = recordBuffer + logManagerLogRecordHeaderSize;
      recordDataSize = recordSize - logManagerLogRecordHeaderSize;
```

```
        rc = SimpleLogRecordDisplay(recordType,
                                    recordFlag,
                                    recordDataBuffer,
                                    recordDataSize);
        break;
      case 'N':
      case 'C':
        recordHeaderBuffer = recordBuffer + logManagerLogRecordHeaderSize;
        componentIdentifier = *(sqluint8 *)recordHeaderBuffer;
        switch (componentIdentifier)
        {
          case 1:
            recordHeaderSize = 6;
            break;
          default:
            printf("    Unknown complex log record: %lu %c %u¥n",
                   recordSize, recordType, componentIdentifier);
            return 1;
        }
        recordDataBuffer = recordBuffer +
                           logManagerLogRecordHeaderSize +
                           recordHeaderSize;
        recordDataSize = recordSize -
                         logManagerLogRecordHeaderSize -
                         recordHeaderSize;
        rc = ComplexLogRecordDisplay(recordType,
                                     recordFlag,
                                     recordHeaderBuffer,
                                     recordHeaderSize,
                                     componentIdentifier,
                                     recordDataBuffer,
                                     recordDataSize);
        break;
      default:
        printf("    Unknown log record: %lu %c¥n",
               recordSize, (char)recordType);
        break;
    }

    return 0;
  } /* LogRecordDisplay */

  int SimpleLogRecordDisplay(sqluint16 recordType,
                             sqluint16 recordFlag,
                             char *recordDataBuffer,
                             sqluint32 recordDataSize)
  {
    int rc = 0;
    sqluint32 timeTransactionCommited;
    sqluint16 authIdLen;
    char authId[129];

    switch ((char)recordType)
    {
      case 'E':
```

```
      printf("¥n    Record type: Local pending list¥n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = *(sqluint16 *)(recordDataBuffer + 4);
      memcpy(authId, (char *)(recordDataBuffer + 6), authIdLen);
      authId[authIdLen] = '¥0';
      printf("      %s: %lu¥n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s¥n", authId);
      break;
    case 'M':
      printf("¥n    Record type: Normal commit¥n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = (sqluint16) (recordDataSize - 4);
      memcpy(authId, (char *)(recordDataBuffer + 4), authIdLen);
      authId[authIdLen] = '¥0';
      printf("      %s: %lu¥n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s¥n", authId);
      break;
    case 'A':
      printf("¥n    Record type: Normal abort¥n");
      authIdLen = (sqluint16) (recordDataSize);
      memcpy(authId, (char *)(recordDataBuffer), authIdLen);
      authId[authIdLen] = '¥0';
      printf("      authorization ID of the application: %s¥n", authId);
      break;
    default:
      printf("    Unknown simple log record: %c %lu¥n",
             (char)recordType, recordDataSize);
      break;
  }

  return 0;
} /* SimpleLogRecordDisplay */

int ComplexLogRecordDisplay(sqluint16 recordType,
                            sqluint16 recordFlag,
                            char *recordHeaderBuffer,
                            sqluint32 recordHeaderSize,
                            sqluint8 componentIdentifier,
                            char *recordDataBuffer,
                            sqluint32 recordDataSize)
{
  int rc = 0;
  sqluint8 functionIdentifier;
  /* for insert, delete, undo delete */
  sqluint32 RID;
  sqluint16 subRecordLen;
  sqluint16 subRecordOffset;
  char *subRecordBuffer;
  /* for update */
  sqluint32 newRID;
  sqluint16 newSubRecordLen;
```

```
sqluint16 newSubRecordOffset;
char *newSubRecordBuffer;
sqluint32 oldRID;
sqluint16 oldSubRecordLen;
sqluint16 oldSubRecordOffset;
char *oldSubRecordBuffer;
/* for alter table attributes */
sqluint32 alterBitMask;
sqluint32 alterBitValues;

switch ((char)recordType)
{
  case 'N':
    printf("¥n    Record type: Normal¥n");
    break;
  case 'C':
    printf("¥n    Record type: Compensation¥n");
    break;
  default:
    printf("¥n    Unknown complex log record type: %c¥n", recordType);
    break;
}

switch (componentIdentifier)
{
  case 1:
    printf("      component ID: DMS log record¥n");
    break;
  default:
    printf("      unknown component ID: %d¥n", componentIdentifier);
    break;
}

functionIdentifier = *(sqluint8 *)(recordHeaderBuffer + 1);
switch (functionIdentifier)
{
  case 106:
    printf("      function ID: Delete Record¥n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu¥n", RID);
    printf("        subrecord length: %u¥n", subRecordLen);
    printf("        subrecord offset: %u¥n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 111:
    printf("      function ID: Undo Delete Record¥n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu¥n", RID);
    printf("        subrecord length: %u¥n", subRecordLen);
    printf("        subrecord offset: %u¥n", subRecordOffset);
```

```
  subRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
  break;
case 118:
  printf("      function ID: Insert Record¥n");
  RID = *(sqluint32 *)(recordDataBuffer + 2);
  subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
  subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
  printf("        RID: %lu¥n", RID);
  printf("        subrecord length: %u¥n", subRecordLen);
  printf("        subrecord offset: %u¥n", subRecordOffset);
  subRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
  break;
case 120:
  printf("      function ID: Update Record¥n");
  oldRID = *(sqluint32 *)(recordDataBuffer + 2);
  oldSubRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
  oldSubRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
  newRID = *(sqluint32 *)(recordDataBuffer +
                          12 + oldSubRecordLen + recordHeaderSize + 2);
  newSubRecordLen = *(sqluint16 *)(recordDataBuffer +
                                   12 + oldSubRecordLen +
                                   recordHeaderSize + 6);
  newSubRecordOffset =
    *(sqluint16 *)(recordDataBuffer + 12 + oldSubRecordLen +
                   recordHeaderSize + 10);
  printf("        oldRID: %lu¥n", oldRID);
  printf("        old subrecord length: %u¥n", oldSubRecordLen);
  printf("        old subrecord offset: %u¥n",
         oldSubRecordOffset);
  oldSubRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(oldSubRecordBuffer, oldSubRecordLen);
  printf("        newRID: %lu¥n", newRID);
  printf("        new subrecord length: %u¥n", newSubRecordLen);
  printf("        new subrecord offset: %u¥n",
         newSubRecordOffset);
  newSubRecordBuffer = recordDataBuffer +
    12 + oldSubRecordLen + recordHeaderSize + 12;
  rc = LogSubRecordDisplay(newSubRecordBuffer, newSubRecordLen);
  break;
case 124:
  printf("      function ID:  Alter Table Attribute¥n");
  alterBitMask = *(sqluint32 *)(recordDataBuffer + 2);
  alterBitValues = *(sqluint32 *)(recordDataBuffer + 6);
  if (alterBitMask & 0x00000001)
  {
    /* Alter the value of the 'propagation' attribute: */
    printf("        Propagation attribute is changed to: ");
    if (alterBitValues & 0x00000001)
    {
      printf("ON¥n");
    }
    else
    {
```

```
    printf("OFF¥n");
  }
}
if (alterBitMask & 0x00000002)
{
  /* Alter the value of the 'pending' attribute: */
  printf("        Pending attribute is changed to: ");
  if (alterBitValues & 0x00000002)
  {
    printf("ON¥n");
  }
  else
  {
    printf("OFF¥n");
  }
}
if (alterBitMask & 0x00010000)
{
  /* Alter the value of the 'append mode' attribute: */
  printf("        Append Mode attribute is changed to: ");
  if (alterBitValues & 0x00010000)
  {
    printf("ON¥n");
  }
  else
  {
    printf("OFF¥n");
  }
}
if (alterBitMask & 0x00200000)
{
  /* Alter the value of the 'LF Propagation' attribute: */
  printf("        LF Propagation attribute is changed to: ");
  if (alterBitValues & 0x00200000)
  {
    printf("ON¥n");
  }
  else
  {
    printf("OFF¥n");
  }
}
if (alterBitMask & 0x00400000)
{
  /* Alter the value of the 'LOB Propagation' attribute: */
  printf("        LOB Propagation attribute is changed to: ");
  if (alterBitValues & 0x00400000)
  {
    printf("ON¥n");
  }
  else
  {
    printf("OFF¥n");
  }
}
```

```
      break;
    default:
      printf("      unknown function identifier: %u¥n",
             functionIdentifier);
      break;
  }

  return 0;
} /* ComplexLogRecordDisplay */

int LogSubRecordDisplay(char *recordBuffer, sqluint16 recordSize)
{
  int rc = 0;
  sqluint8 recordType;
  sqluint8 updatableRecordType;
  sqluint16 userDataFixedLength;
  char *userDataBuffer;
  sqluint16 userDataSize;

  recordType = *(sqluint8 *)(recordBuffer);
  if (recordType != 0 && recordType != 4)
  {
    printf("        Unknown subrecord type.¥n");
  }
  else if (recordType == 4)
  {
    printf("        subrecord type: Special control¥n");
  }
  else
  {
    /* recordType == 0 */
    printf("        subrecord type: Updatable, ");
    updatableRecordType = *(sqluint8 *)(recordBuffer + 4);
    if (updatableRecordType != 1)
    {
      printf("Internal control¥n");
    }
    else
    {
      printf("Formatted user data¥n");
      userDataFixedLength = *(sqluint16 *)(recordBuffer + 6);
      printf("        user data fixed length: %u¥n",
             userDataFixedLength);
      userDataBuffer = recordBuffer + 8;
      userDataSize = recordSize - 8;
      rc = UserDataDisplay(userDataBuffer, userDataSize);
    }
  }

  return 0;
} /* LogSubRecordDisplay */

int UserDataDisplay(char *dataBuffer, sqluint16 dataSize)
{
  int rc = 0;
```

```
  sqluint16 line, col;

  printf("        user data:¥n");

  for (line = 0; line * 10 < dataSize; line = line + 1)
  {
        printf(" ");
    for (col = 0; col < 10; col = col + 1)
    {
      if (line * 10 + col < dataSize)
      {
        printf("%02X ", dataBuffer[line * 10 + col]);
      }
      else
      {
        printf(" ");
      }
    }
    printf("*");
    for (col = 0; col < 10; col = col + 1)
    {
      if (line * 10 + col < dataSize)
      {
        if (isalpha(dataBuffer[line * 10 + col]) ||
            isdigit(dataBuffer[line * 10 + col]))
        {
          printf("%c", dataBuffer[line * 10 + col]);
        }
        else
        {
          printf(".");
        }
      }
      else
      {
        printf(" ");
      }
    }
    printf("*");
    printf("¥n");
  }

  return 0;
} /* UserDataDisplay */

int DbRecoveryHistoryFileRead(char dbAlias[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint32 numEntries;
  sqluint16 recoveryHistoryFileHandle;
  sqluint32 entryNb;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
```

```
struct db2HistoryData histEntryData;

printf("¥n*********************************************¥n");
printf("*** READ A DATABASE RECOVERY HISTORY FILE ***¥n");
printf("*********************************************¥n");
printf("¥nUSE THE DB2 APIs:¥n");
printf("  db2HistoryOpenScan -- Open Recovery History File Scan¥n");
printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry¥n");
printf("  db2HistoryCloseScan -- Close Recovery History File Scan¥n");
printf("TO READ A DATABASE RECOVERY HISTORY FILE.¥n");

/* initialize the data structures */
dbHistoryOpenParam.piDatabaseAlias = dbAlias;
dbHistoryOpenParam.piTimestamp = NULL;
dbHistoryOpenParam.piObjectName = NULL;
dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;

dbHistoryEntryGetParam.pioHistData = &histEntryData;
dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
rc = HistoryEntryDataFieldsAlloc(&histEntryData);
if (rc != 0)
{
  return rc;
}

/*****************************************/
/* OPEN THE DATABASE RECOVERY HISTORY FILE */
/*****************************************/
printf("¥n  Open recovery history file for '%s' database.¥n", dbAlias);

/* open the recovery history file to scan */
db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("database recovery history file -- open");
numEntries = dbHistoryOpenParam.oNumRows;

/* dbHistoryOpenParam.oHandle returns the handle for scan access */
recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

/*********************************************/
/* READ AN ENTRY IN THE RECOVERY HISTORY FILE */
/*********************************************/
for (entryNb = 0; entryNb < numEntries; entryNb = entryNb + 1)
{
  printf("¥n  Read entry number %u.¥n", entryNb);

  /* get the next entry from the recovery history file */
  db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
  DB2_API_CHECK("database recovery history file entry -- read")

  /* display the entries in the recovery history file */
  printf("¥n  Display entry number %u.¥n", entryNb);
  rc = HistoryEntryDisplay(histEntryData);
}
```

```
  /*******************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************/
  printf("¥n  Close recovery history file for '%s' database.¥n", dbAlias);

  /* The API db2HistoryCloseScan ends the recovery history file scan and
     frees DB2 resources required for the scan. */
  db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);

  return 0;
} /* DbRecoveryHistoryFileRead */

int HistoryEntryDataFieldsAlloc(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb;

  strcpy(pHistEntryData->ioHistDataID, "SQLUHINF");

  pHistEntryData->oObjectPart.pioData = malloc(17 + 1);
  pHistEntryData->oObjectPart.iLength = 17 + 1;

  pHistEntryData->oEndTime.pioData = malloc(12 + 1);
  pHistEntryData->oEndTime.iLength = 12 + 1;

  pHistEntryData->oFirstLog.pioData = malloc(8 + 1);
  pHistEntryData->oFirstLog.iLength = 8 + 1;

  pHistEntryData->oLastLog.pioData = malloc(8 + 1);
  pHistEntryData->oLastLog.iLength = 8 + 1;

  pHistEntryData->oID.pioData = malloc(128 + 1);
  pHistEntryData->oID.iLength = 128 + 1;

  pHistEntryData->oTableQualifier.pioData = malloc(128 + 1);
  pHistEntryData->oTableQualifier.iLength = 128 + 1;

  pHistEntryData->oTableName.pioData = malloc(128 + 1);
  pHistEntryData->oTableName.iLength = 128 + 1;

  pHistEntryData->oLocation.pioData = malloc(128 + 1);
  pHistEntryData->oLocation.iLength = 128 + 1;

  pHistEntryData->oComment.pioData = malloc(128 + 1);
  pHistEntryData->oComment.iLength = 128 + 1;

  pHistEntryData->oCommandText.pioData = malloc(128 + 1);
  pHistEntryData->oCommandText.iLength = 128 + 1;

  pHistEntryData->poEventSQLCA =
    (struct sqlca *)malloc(sizeof(struct sqlca));
```

```
  pHistEntryData->poTablespace = (db2Char *)malloc(3 * sizeof(db2Char));
  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
  {
    pHistEntryData->poTablespace[tsNb].pioData = malloc(18 + 1);
    pHistEntryData->poTablespace[tsNb].iLength = 18 + 1;
  }

  pHistEntryData->iNumTablespaces = 3;

  return 0;
} /* HistoryEntryDataFieldsAlloc */

int HistoryEntryDisplay(struct db2HistoryData histEntryData)
{
  int rc = 0;
  char buf[129];
  sqluint32 tsNb;

  memcpy(buf, histEntryData.oObjectPart.pioData,
              histEntryData.oObjectPart.oLength);
  buf[histEntryData.oObjectPart.oLength] = '¥0';
  printf("    object part: %s¥n", buf);

  memcpy(buf, histEntryData.oEndTime.pioData,
              histEntryData.oEndTime.oLength);
  buf[histEntryData.oEndTime.oLength] = '¥0';
  printf("    end time: %s¥n", buf);

  memcpy(buf, histEntryData.oFirstLog.pioData,
              histEntryData.oFirstLog.oLength);
  buf[histEntryData.oFirstLog.oLength] = '¥0';
  printf("    first log: %s¥n", buf);

  memcpy(buf, histEntryData.oLastLog.pioData,
              histEntryData.oLastLog.oLength);
  buf[histEntryData.oLastLog.oLength] = '¥0';
  printf("    last log: %s¥n", buf);

  memcpy(buf, histEntryData.oID.pioData, histEntryData.oID.oLength);
  buf[histEntryData.oID.oLength] = '¥0';
  printf("    ID: %s¥n", buf);

  memcpy(buf, histEntryData.oTableQualifier.pioData,
              histEntryData.oTableQualifier.oLength);
  buf[histEntryData.oTableQualifier.oLength] = '¥0';
  printf("    table qualifier: %s¥n", buf);

  memcpy(buf, histEntryData.oTableName.pioData,
              histEntryData.oTableName.oLength);
  buf[histEntryData.oTableName.oLength] = '¥0';
  printf("    table name: %s¥n", buf);

  memcpy(buf, histEntryData.oLocation.pioData,
              histEntryData.oLocation.oLength);
```

```
  buf[histEntryData.oLocation.oLength] = '¥0';
  printf("    location: %s¥n", buf);

  memcpy(buf, histEntryData.oComment.pioData,
              histEntryData.oComment.oLength);
  buf[histEntryData.oComment.oLength] = '¥0';
  printf("    comment: %s¥n", buf);

  memcpy(buf, histEntryData.oCommandText.pioData,
              histEntryData.oCommandText.oLength);
  buf[histEntryData.oCommandText.oLength] = '¥0';
  printf("    command text: %s¥n", buf);
  printf("    history file entry ID: %u¥n", histEntryData.oEID.ioHID);
  printf("    table spaces:¥n");

  for (tsNb = 0; tsNb < histEntryData.oNumTablespaces; tsNb = tsNb + 1)
  {
    memcpy(buf, histEntryData.poTablespace[tsNb].pioData,
                histEntryData.poTablespace[tsNb].oLength);
    buf[histEntryData.poTablespace[tsNb].oLength] = '¥0';
    printf("      %s¥n", buf);
  }

  printf("    type of operation: %c¥n", histEntryData.oOperation);
  printf("    granularity of the operation: %c¥n", histEntryData.oObject);
  printf("    operation type: %c¥n", histEntryData.oOptype);
  printf("    entry status: %c¥n", histEntryData.oStatus);
  printf("    device type: %c¥n", histEntryData.oDeviceType);
  printf("    SQLCA:¥n");
  printf("      sqlcode: %ld¥n", histEntryData.poEventSQLCA->sqlcode);
  memcpy(buf, histEntryData.poEventSQLCA->sqlstate, 5);
  buf[5] = '¥0';
  printf("      sqlstate: %s¥n", buf);
  memcpy(buf, histEntryData.poEventSQLCA->sqlerrmc,
              histEntryData.poEventSQLCA->sqlerrml);
  buf[histEntryData.poEventSQLCA->sqlerrml] = '¥0';
  printf("      message: %s¥n", buf);

  return 0;
} /* HistoryEntryDisplay */

int HistoryEntryDataFieldsFree(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb;

  free(pHistEntryData->oObjectPart.pioData);
  free(pHistEntryData->oEndTime.pioData);
  free(pHistEntryData->oFirstLog.pioData);
  free(pHistEntryData->oLastLog.pioData);
  free(pHistEntryData->oID.pioData);
  free(pHistEntryData->oTableQualifier.pioData);
  free(pHistEntryData->oTableName.pioData);
  free(pHistEntryData->oLocation.pioData);
  free(pHistEntryData->oComment.pioData);
```

```
  free(pHistEntryData->oCommandText.pioData);
  free(pHistEntryData->poEventSQLCA);

  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
  {
    free(pHistEntryData->poTablespace[tsNb].pioData);
  }

  free(pHistEntryData->poTablespace);

  return 0;
} /* HistoryEntryDataFieldsFree */

int DbFirstRecoveryHistoryFileEntryUpdate(char dbAlias[],
                                          char user[],
                                          char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint16 recoveryHistoryFileHandle;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;
  char newLocation[DB2HISTORY_LOCATION_SZ + 1];
  char newComment[DB2HISTORY_COMMENT_SZ + 1];
  struct db2HistoryUpdateStruct dbHistoryUpdateParam;

  printf("¥n*****************************************************¥n");
  printf("*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***¥n");
  printf("*****************************************************¥n");
  printf("¥nUSE THE DB2 APIs:¥n");
  printf("  db2HistoryOpenScan -- Open Recovery History File Scan¥n");
  printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry¥n");
  printf("  db2HistoryUpdate -- Update Recovery History File¥n");
  printf("  db2HistoryCloseScan -- Close Recovery History File Scan¥n");
  printf("TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.¥n");

  /* initialize data structures */
  dbHistoryOpenParam.piDatabaseAlias = dbAlias;
  dbHistoryOpenParam.piTimestamp = NULL;
  dbHistoryOpenParam.piObjectName = NULL;
  dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;
  dbHistoryEntryGetParam.pioHistData = &histEntryData;
  dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
  rc = HistoryEntryDataFieldsAlloc(&histEntryData);
  if (rc != 0)
  {
    return rc;
  }

  /*****************************************/
  /* OPEN THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("¥n  Open the recovery history file for '%s' database.¥n", dbAlias);
```

```
/* The API db2HistoryOpenScan starts a recovery history file scan */
db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("database recovery history file -- open");

/* dbHistoryOpenParam.oHandle returns the handle for scan access */
recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

/*************************************************/
/* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE */
/*************************************************/
printf("¥n  Read the first entry in the recovery history file.¥n");

/* The API db2HistoryGetEntry gets the next entry from the recovery
   history file. */
db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
DB2_API_CHECK("first recovery history file entry -- read");
printf("¥n  Display the first entry.¥n");

/* HistoryEntryDisplay is a support function used to display the entries
   in the recovery history file. */
rc = HistoryEntryDisplay(histEntryData);

/* update the first history file entry */
rc = DbConn(dbAlias, user, pswd);
if (rc != 0)
{
  return rc;
}

strcpy(newLocation, "this is the NEW LOCATION");
strcpy(newComment, "this is the NEW COMMENT");
printf("¥n  Update the first entry in the history file:¥n");
printf("    new location = '%s'¥n", newLocation);
printf("    new comment = '%s'¥n", newComment);
dbHistoryUpdateParam.piNewLocation = newLocation;
dbHistoryUpdateParam.piNewDeviceType = NULL;
dbHistoryUpdateParam.piNewComment = newComment;
dbHistoryUpdateParam.iEID.ioNode = histEntryData.oEID.ioNode;
dbHistoryUpdateParam.iEID.ioHID = histEntryData.oEID.ioHID;

/* The API db2HistoryUpdate can be used to update the location,
   device type, or comment in a history file entry. */

/* Call this API to update the location and comment of the first
   entry in the history file: */
db2HistoryUpdate(db2Version710, &dbHistoryUpdateParam, &sqlca);
DB2_API_CHECK("first history file entry -- update");

rc = DbDisconn(dbAlias);
if (rc != 0)
{
  return rc;
}
```

```
  /*******************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************/
  printf("¥n  Close recovery history file for '%s' database.¥n", dbAlias);

  /* The API db2HistoryCloseScan ends the recovery history file scan and
     frees DB2 resources required for the scan. */
  db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");


  /*******************************************/
  /* RE-OPEN THE DATABASE RECOVERY HISTORY FILE */
  /**********************************************
  printf("¥n  Open the recovery history file for '%s' database.¥n", dbAlias);

  /* starts a recovery history file scan */
  db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
  DB2_API_CHECK("database recovery history file -- open");

  recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;

  dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
  printf("¥n  Read the first recovery history file entry.¥n");

  /*************************************************************************/
  /* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE AFTER MODIFICATION */
  /*************************************************************************/
  db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
  DB2_API_CHECK("first recovery history file entry -- read");

  printf("¥n  Display the first entry.¥n");
  rc = HistoryEntryDisplay(histEntryData);

  /*******************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************/
  printf("¥n  Close the recovery history file for '%s' database.¥n",
         dbAlias);

  /* ends the recovery history file scan */
  db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);

  return 0;
} /* DbFirstRecoveryHistoryFileEntryUpdate */

int DbRecoveryHistoryFilePrune(char dbAlias[], char user[], char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2PruneStruct histPruneParam;
```

```
  char timeStampPart[14 + 1];

  printf("¥n*************************************¥n");
  printf("*** PRUNE THE RECOVERY HISTORY FILE ***¥n");
  printf("*************************************¥n");
  printf("¥nUSE THE DB2 API:¥n");
  printf("  db2Prune -- Prune Recovery History File¥n");
  printf("AND THE SQL STATEMENTS:¥n");
  printf("  CONNECT¥n");
  printf("  CONNECT RESET¥n");
  printf("TO PRUNE THE RECOVERY HISTORY FILE.¥n");

  /* Connect to the database: */
  rc = DbConn(dbAlias, user, pswd);
  if (rc != 0)
  {
    return rc;
  }

  /* Prune the recovery history file: */
  printf("¥n  Prune the recovery history file for '%s' database.¥n",
         dbAlias);

  /* timeStampPart is a pointer to a string specifying a time stamp or
     log sequence number. Time stamp is used here to select records for
     deletion. All entries equal to or less than the time stamp will be
     deleted. */
  histPruneParam.piString = timeStampPart;
  strcpy(timeStampPart, "2010"); /* year 2010 */

  /* The action DB2PRUNE_ACTION_HISTORY removes history file entries: */
  histPruneParam.iAction = DB2PRUNE_ACTION_HISTORY;

  /* The option DB2PRUNE_OPTION_FORCE forces the removal of the last backup: */
  histPruneParam.iOptions = DB2PRUNE_OPTION_FORCE;

  /* db2Prune can be called to delete entries from the recovery history file
     or log files from the active log path. Here we call it to delete
     entries from the recovery history file.
     You must have SYSADM, SYSCTRL, SYSMAINT, or DBADM authority to prune
     the recovery history file. */
  db2Prune(db2Version710, &histPruneParam, &sqlca);
  DB2_API_CHECK("recovery history file -- prune");

  /* Disconnect from the database: */
  rc = DbDisconn(dbAlias);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* DbRecoveryHistoryFilePrune */
```

付録F. 回復 CLP スクリプト

以下の DB2 コマンド・スクリプトは、CLP コマンドを使用して以下のことを行う方法を示します。

- データベースのバックアップ
- データベースの復元
- データベースのロールフォワード回復

SAMPLE データベースが存在していて、使用中でないことを確認してください。SAMPLE データベースについては、*SQL 解説書* を参照してください。 DB2 コマンド行プロセッサーについての一般情報は、*コマンド解説書* を参照してください。

Windows 互換および UNIX 互換の両方のバージョンのスクリプトが説明されています。

Windows オペレーティング・システム用のサンプル・コマンド・スクリプト

Windows NT または Windows 2000 上でスクリプトを実行するには、以下のようにします。

1. スクリプトを、たとえば `backrest.db2` のような名前の付いたファイルに保管します。
2. データベース・マネージャーが実行されていない場合、 **db2start** コマンドを DB2 コマンド・ウィンドウから発行します。 CLP 対応の DB2 ウィンドウをオープンし、 Windows オペレーティング・システム上で DB2 コマンド行環境を初期化するには、コマンド・プロンプトから **db2cmd** を発行します。
3. `db2 -f backrest.db2 -t` と入力します。

以下はこのスクリプトによって戻される出力の例です。

```
D:¥>db2 -f backrest.db2 -t
This is CLP script: backrest.db2


Deleting old SAMPLE database backup images...

process SAMPLE.0¥DB2¥NODE0000¥CATN0000
process 20010403

Updating the database configuration parameter LOGRETAIN to 'ON'...

DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  For most configuration parameters, all applications must disconnect
```

```
from this database before the changes become effective.


Backing up the SAMPLE database...


Backup successful. The timestamp for this backup image is : 20010403131027


Restoring the SAMPLE database as TESTBACK (1st pass)...

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.

Listing the table spaces for the TESTBACK database...


           Tablespaces for Current Database

 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined


Defining new table space containers for Tablespace 2...

DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.
```

```
Listing table space containers for Tablespace 2 (TESTBACK database)...


           Tablespace Containers for Tablespace 2

 Container ID                         = 0
 Name                                 = c:¥ts2con1
 Type                                 = Path



Restoring the SAMPLE database as TESTBACK (2nd pass)...

DB20000I  The RESTORE DATABASE command completed successfully.

Rolling the TESTBACK database forward...


                                Rollforward Status

 Input database alias                 = testback
 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           = 2001-04-03-03.16.07.000000

DB20000I  The ROLLFORWARD command completed successfully.

Dropping the TESTBACK database...

DB20000I  The DROP DATABASE command completed successfully.

Terminating the command line processor's back-end process...

DB20000I  The TERMINATE command completed successfully.


D:¥>
```

以下はこのスクリプトのソース・リストです。

```
-- Before proceeding, ensure that:
--   The database manager is running
--   The SAMPLE database exists and is not in use.

-- Run the script by issuing:
--   db2 -f backrest.db2 -t
--     where -f tells the command line processor to read command input
--               from a file instead of from standard input, and
--           -t tells the command line processor to use a semicolon (;)
--              as the statement termination character.
```

```
!ECHO This is CLP script: backrest.db2;

-- Ensure that the DB2 profile registry variable DB2_ENABLE_LDAP
--   is set to 'NO':
!db2set DB2_ENABLE_LDAP=NO;

!ECHO Deleting old SAMPLE database backup images...;
!rd! SAMPLE.0¥DB2¥NODE0000¥CATN0000;

!ECHO Updating the database configuration parameter LOGRETAIN to 'ON'...;
update db cfg for sample using logretain on;

!ECHO Backing up the SAMPLE database...;
backup db sample;

!ECHO Restoring the SAMPLE database as TESTBACK (1st pass)...;
restore db sample into testback redirect;

!ECHO Listing the table spaces for the TESTBACK database...;
list tablespaces;

!ECHO Defining new table space containers for Tablespace 2...;
set tablespace containers for 2 using (path "c:¥ts2con1");

!ECHO Listing table space containers for Tablespace 2 (TESTBACK database)...;
list tablespace containers for 2;

!ECHO Restoring the SAMPLE database as TESTBACK (2nd pass)...;
restore db sample continue;

!ECHO Rolling the TESTBACK database forward...;
rollforward db testback stop;

!ECHO Dropping the TESTBACK database...;
drop db testback;

!ECHO Terminating the command line processor's back-end process...;
terminate;

-- End file
```

UNIX ベースのシステム用のサンプル・コマンド・スクリプト

UNIX ベースのオペレーティング・システム上でスクリプトを実行するには、以下のようにします。

1. スクリプトを、たとえば backrest.db2 のような名前の付いたファイルに保管します。
2. データベース・マネージャーが実行されていない場合、コマンド・プロンプトから **db2start** コマンドを発行します。
3. db2 -f backrest.db2 -t と入力します。

以下はこのスクリプトによって戻される出力の例です。

```
sunfish /export/home2/falexand/samples/clp>db2 -f backrest.db2 -t
This is CLP script: backrest.db2

Deleting old SAMPLE database backup images...

Updating the database configuration parameter LOGRETAIN to 'ON'...
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  For most configuration parameters, all applications must disconnect
from this database before the changes become effective.


Backing up the SAMPLE database...

Backup successful. The timestamp for this backup image is : 20010531172525


Restoring the SAMPLE database as TESTBACK (1st pass)...
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.

Listing the table spaces for the TESTBACK database...

           Tablespaces for Current Database

 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined
```

```
Defining new table space containers for Tablespace 2...
DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.

Listing table space containers for Tablespace 2 (TESTBACK database)...

            Tablespace Containers for Tablespace 2

 Container ID                         = 0
 Name                                 = /export/home2/falexand/falexand...
                                          .../NODE0000/SQL00020/ts2con1
 Type                                 = Path



Restoring the SAMPLE database as TESTBACK (2nd pass)...
DB20000I  The RESTORE DATABASE command completed successfully.

Rolling the TESTBACK database forward...

                                Rollforward Status

 Input database alias                 = testback
 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           = 2001-05-29-21.20.16.000000

DB20000I  The ROLLFORWARD command completed successfully.

Dropping the TESTBACK database...
DB20000I  The DROP DATABASE command completed successfully.

Terminating the command line processor's back-end process...
DB20000I  The TERMINATE command completed successfully.
```

以下はこのスクリプトのソース・リストです。

```
-- Before proceeding, ensure that:
--   The database manager is running
--   The SAMPLE database exists and is not in use.

-- Run the script by issuing:
--   db2 -f backrest.db2 -t
--     where -f tells the command line processor to read command input
--              from a file instead of from standard input, and
--           -t tells the command line processor to use a semicolon (;)
--              as the statement termination character.

echo This is CLP script: backrest.db2;
```

```
-- Ensure that the DB2 profile registry variable DB2_ENABLE_LDAP
--   is set to 'NO':
!db2set DB2_ENABLE_LDAP=NO;

echo Deleting old SAMPLE database backup images...;
!rm -f ./SAMPLE.0.*;

echo Updating the database configuration parameter LOGRETAIN to 'ON'...;
update db cfg for sample using logretain on;

echo Backing up the SAMPLE database...;
backup db sample;

echo Restoring the SAMPLE database as TESTBACK (1st pass)...;
restore db sample into testback redirect;

echo Listing the table spaces for the TESTBACK database...;
list tablespaces;

echo Defining new table space containers for Tablespace 2...;
set tablespace containers for 2 using (path "ts2con1");

echo Listing table space containers for Tablespace 2 (TESTBACK database)...;
list tablespace containers for 2;

echo Restoring the SAMPLE database as TESTBACK (2nd pass)...;
restore db sample continue;

echo Rolling the TESTBACK database forward...;
rollforward db testback stop;

echo Dropping the TESTBACK database...;
drop db testback;

echo Terminating the command line processor's back-end process...;
terminate;

-- End file
```

付録G. Tivoli Storage Manager

DB2 バックアップまたは復元ユーティリティーの起動時に、 Tivoli Storage Manager (TSM、以前の ADSM) 製品を使用して、バックアップまたは復元ユーティリティーの管理を行うことを指定できます。 DB2 では、TSM クライアント バージョン 3.1.x.3 以降を使用できます。

UNIX ベースのプラットフォーム上で Tivoli Storage Manager クライアントをセットアップする

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、事前に以下のセットアップ活動を必ず実行してください。

1. SunOS および Solaris 環境では、以下の手順に従います。 (他の UNIX ベースのプラットフォームでは、2 から始めてください。)
   a. 次の必要とされるオペレーティング・システムのレベルがインストールされていることを確認してください。 SunOS 5.5.1 または Solaris 2.5.1。
   b. TSM クライアント バージョン 3.1.x.3 またはそれ以降をインストールします。このバージョンのクライアントをインストールする前に、以前の TSM パッケージをすべて 除去するようにしてください。
   c. TSM が /opt/IBMDSMap5、/opt/IBMDSMba5、および /opt/IBMDSMsa5 ディレクトリーにインストールされていることを確認します。
   d. 次の記号リンクがまだ存在していないならば、/usr/lib ディレクトリーに作成します。

      ```
      libApiDS.so -> libApiDS.so.1
      libApiDS.so.1 -> /opt/IBMDSMap5/api/libApiDS.so.2
      ```

2. TSM ユーザー構成オプション・ファイル /usr/sbin/dsm.opt、および TSM システム構成オプション・ファイルである /usr/sbin/dsm.sys を、使用している環境に合わせて作成および修正します。
3. SunOS および Solaris 環境では、以下の手順に従います。 (他の UNIX ベースのプラットフォームでは、4 から続けてください。)
   a. /usr/sbin/dsm.opt および /usr/sbin/dsm.sys を、 /opt/IBMDSMap5 ディレクトリーにコピーします。
   b. /opt/IBMDSMap5/solaris/dsmaptica を、 /opt/IBMDSMap5 ディレクトリーにコピーします。
4. TSM 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 (dsmapicta または 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。 SunOS および Solaris 環境では、これを /opt/IBMDSMap5 に設定する必要があります。

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。 SunOS および Solaris 環境では、これを /opt/IBMDSMap5/dsm.opt に設定する必要があります。

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。

5. TSM パスワードを確立します。

   Tivoli クライアントが TSM サーバーとインターフェースで接続できるようにするには、サーバーのパスワードが必要です。実行可能ファイル dsmapipw は、インスタンス所有者の INSTHOME/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、 『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: バックアップまたは復元ユーティリティーを起動するユーザーは、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。

6. 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。

他のプラットフォームで Tivoli Storage Manager クライアントをセットアップする

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、事前に以下のセットアップ活動を必ず実行してください。

1. TSM 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 (dsmapicta または 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。

2. 使用しているオペレーティング・システムに該当する場合は、 TSM システム構成オプション・ファイル (dsm.sys) を作成します (または修正します)。
3. dsm.opt TSM ユーザー構成オプション・ファイルを作成します (または修正します)。環境変数 DSMI_CONFIG は、このファイルを指しています。
4. TSM パスワードを確立します。

   Tivoli クライアントが TSM サーバーとインターフェースで接続できるようにするには、サーバーのパスワードが必要です。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、ローカル管理者として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起動時には、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: バックアップまたは復元ユーティリティーを起動するユーザーは、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。
5. 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。

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 NT オペレーティング・システムおよび OS/2 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

- 表スペース・バックアップ・オブジェクト
  /<database>/NODEnnnn/TSP_BACKUP.timestamp.seq_no
- ロード・コピー・オブジェクト
  /<database>/NODEnnnn/LOAD_COPY.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・スタンプと順序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。
- 個別のバックアップ・イメージは、 TSM グラフィカル・ユーザー・インターフェースから認識することはできません。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 (314ページの『db2adutl - TSM アーカイブ・イメージによる作業』 を参照してください)。
- 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、復元操作中に大きな 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復元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。
- データベース・マネージャーでは、TSM 全バックアップ・オプションを使用します。 TSM 増分バックアップ操作はサポートされていません。
- スループットを大きくするには、複数のセッションを使用します。
- 非 UNIX ベースのシステムでは、 DB2 バックアップおよび復元ユーティリティーで複数の TSM セッションを使用することはできません。

Windows オペレーティング・システムおよび OS/2 での現行の Tivoli クライアントは、再入可能です。したがって、1 つのマシンからバックアップ、復元、またはロード・ユーティリティーを使用して、複数の入出力セッションを安全に作成することができます。インストールした TSM クライアントのバージョンがこの機能をサポートしているかどうかの確認は、ユーザーの責任で行ってください。

単一ノード構成で、ユーザーが以下のような BACKUP DATABASE コマンドの発行を試行するとします。

```
db2 backup db sample use tsm open 3 sessions
```

この場合 DB2 は、複数のセッションが TSM によってサポートされていないことを検出し、エラー・メッセージを戻します。 (このことは、TSM を使用し、COPY YES オプションを指定したロード操作にも当てはまります。)

ただし、Windows NT 上の複数の論理ノード (MLN) 構成では、それぞれの論理ノードが 1 つのセッションしか作成しようとしない場合、 DB2 は単一のマシン上の複数のセッションの使用を検出できないので注意してください。このため MLN 構成では、それらの TSM クライアントが再入可能かどうかを検査することが重要です。 TSM を使用して複数の論理ノードが並列してバックアップ、復元、またはロードされている場合には、論理ノードが実際には同一のハードウェアに常駐している場合であっても、それぞれのノードが単一セッションを使用しようとしているなら、DB2 はオペレーションの継続を許可します。そのため、最新の TSM クライアントを使用していない場合、バックアップの試行に失敗し、ロード・プロセスをハングさせる可能性があるので、そのような試行は行わないでください。

TSM でのバックアップおよびログ・アーカイブの管理

db2adutl ユーティリティーを使用すると、 TSM を使用して保管されたバックアップ・イメージ、ログ、およびロード・コピー・イメージの照会、取り出し、および削除を行うことができます。このユーティリティーは、UNIX ベースのシステムでは `INSTHOME/sqllib/misc` ディレクトリーにインストールされます。 Windows オペレーティング・システムおよび OS/2 では `¥sqllib¥misc` ディレクトリーにインストールされます。このユーティリティーの詳細については、314ページの『db2adutl - TSM アーカイブ・イメージによる作業』を参照してください。

QUERY オプションを使用すると、バックアップ・イメージ、ロード・コピー・イメージ、またはログをリストすることができます。リストされるログの範囲を選択することができます。また、非活動バックアップ・イメージの表示を要求することもできます。

EXTRACT オプションを使用すると、バックアップ・イメージまたはログを TSM から現行ディレクトリーにコピーすることができます。どのバックアップまたはログを取り出すかを選択することができます。

DELETE オプションを使用すると、バックアップ・イメージを非活動化する、また TSM からログを削除することができます。どのバックアップまたはログを処理するかを選択することができます。 KEEP *n* オプションを使用して、最新の *n* 個のバックアップ・イメージを保持することができます。また、OLDER THAN *timestamp* または *n* DAYS オプションを使用して、 DELETE 要求を調整することもできます。

データ・リンクとの Tivoli Space Manager の統合

DB2 データ・リンク・マネージャーは、Tivoli Space Manager (TSM) とその仮想ファイル・システム (FSM) を使用することができます。それは、固有のジャーナル・ファイル・システム (JFS) の階層の最上位にあります。 FSM は JFS と同様の方法でアクセスおよび構成することができます。

この機能は、定期的に 3 次ストレージに移動させる必要のある、大規模ファイルのあるファイル・システムを持つユーザーにとって役立ちます。これらのファイル・システムのスペースは定期的に管理する必要があります。 DB2 データ・リンク・マネージャーが TSM をサポートすることで、 DATALINK ファイルのスペースをより柔軟に管理できるようになります。 DB2 データ・リンク・マネージャー・ファイル・システムに、そこに保管される可能性のあるすべてのファイルに十分なストレージを前もって割り当てる代わりに、 TSM を使用すると、データ・リンクが管理するファイル・システムの割り振りをある期間にわたって調整することができます。こうすると、通常の使用中にファイル・システムを使用し尽くしてしまう危険性がありません。

制約事項と制限

- この機能は現在 AIX でのみサポートされています。
- 選択的移行 (**dsmmigrate**) および FC (読み取り許可データベース) にリンクしているファイルの再呼び出しは、ルート・ユーザーのみが行わなければなりません。

  選択的移行は、ファイル所有者によってのみ行うことができます。読み取り許可データベース・ファイルの場合は DataLink Manager 管理者 (dlfm) です。そのようなファイルにアクセスするには、ホスト・データベースからのトークンが必要です。トークンを必要としないユーザーは 『root』 ユーザーのみです。 『root』 ユーザーがその種のファイルに対して選択的移行および再呼び出しを行うのは比較的容易です。 dlfm ユーザーは、最初は有効なトークンを使用して FC ファイルを移行することができます。 2 度目に移行が試行されると (再呼び出しの後)、操作は失敗し、 『ANS1028S Internal program error. Please see your service representative.』 というエラー・メッセージが戻されます。非ルート・ユーザーが FC ファイルに対して **dsmmigrate** を実行しようとすると、操作は失敗します。ファイル・サーバー上のファイルにアクセスするのは通常は管理者なので、この制限は小さいものです。
- **stat** および **statfs** システム呼び出しは、 dlfs が fsm 上にマウントされている場合でも、 *Vfs-type* を `dlfs` ではなく `fsm` として表示します。

  この動作により、**dsmrecalld** デーモンの正常な機能性がサポートされます。これはファイル・システム上で **statfs** を実行して、その *Vfs-type* が `fsm` かどうか判別します。
- **dsmls** コマンドは、最小の *inode* 番号を持つファイルが FC (読み取り許可データベース) にリンクしている場合、何の出力も表示しません。

  dsmls コマンドは **ls** コマンドに類似していて、 TSM によって管理されているファイルをリストします。ユーザー処置は必要ありません。

付録H. データベース回復用のユーザー出口

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。 (OS/2 では、バックアップおよび復元操作用にもユーザー出口プログラムを使用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、 *userexit* データベース構成パラメーターが YES 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回復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。 (OS/2 では、バックアップおよび復元操作によってまず db2uexit.cmd が呼び出され、次にそれにより db2uext2 が呼び出されます。) 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 (463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』
- 460ページの『呼び出し形式』
- 462ページの『バックアップおよび復元に関する考慮事項 - DB2 (OS/2 版) のみ』
- 463ページの『エラー処理』

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回復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、回復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.cadsm

    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、Legato** Systems 社から入手できる、 Legato NetWorker** Version 4.2.5 プログラムを使用します。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.cadsm

    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。

- **OS/2**

  DB2 (OS/2 版) のユーザー出口サンプル・プログラムは、 ¥sqllib¥samples¥rexx ディレクトリーのインスタンス・サブディレクトリーにあります。 (dbuexit.CAD プログラムは例外で、 ¥sqllib¥samples¥c ディレクトリーのインスタンス・サブディレクトリーにあります。) 用意されているサンプルはほとんどが REXX コマンド・ファイルですが、ユーザー出口プログラムはそれとは異なるプログラミング言語で作成することができます。

  インプリメントしたいサンプルの名前は、db2uexit という名前に変更し、 .cmd または .exe のいずれかの拡張子を付ける必要があります。名前変更したファイルを ¥sqllib¥bin ディレクトリーに移動してください。

  5 つの OS/2 サンプル・ユーザー出口プログラムがあります。

  – db2uexit.ex1

    このサンプルは、 Seagate Software Inc. から入手できる Sytos Premium バージョン 2.2 プログラムを使用します。 IBM 外部テープ装置へのデータの保管とその検索を行います。現時点では Sytos Premium 製品のバージョン 2.2 だけがサポートされています。 (この製品を使用するには OS/2 FixPak 26 が必要です。) サンプル・プログラムのリストを検討して、他の要件を調べてください。

  – db2uexit.ex2

    このサンプルは、Mountain Corporation から入手できる Filesafe プログラムを使用します。 Mountain テープ装置へのデータの保管とその検索を行うのに使用できます。固有のボリューム・ラベルが、データベースの各バックアップ・コピーに割り当てられて、 1 つ以上のデータベースからの複数のバックアップ・イメージを同じテープ上に保管することができます。

  – db2uexit.ex3

    このサンプルは、Maynard Corporation から入手できる MaynStream プログラムを使用します。 Maynard テープ装置へのデータの保管とその検索を行うのに使用できます。 MaynStream は、データベースがバックアップされた場所以外のドライブにデータベース復元操作をリダイレクトすることをサポートしません。

  – db2uexit.ex4

    このサンプルは OS/2 XCOPY コマンドを使用します。ストレージ・デバイスとしては、ハード・ディスク、ディスケット、または光ディスク・カートリッジなどの、OS/2 によりサポートされる装置を使用できます。これらの装置は、ワークステーションが適切に構成されている場合、 LAN リダイレクト・ドライブでも構いません。

    データベースのバックアップと復元に XCOPY を使うことはできません。

  – db2uexit.CAD

このサンプルは C 言語で書かれていて、 Tivoli Storage Manager (TSM) サンプル・プログラムと等価です。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。

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。

- UNIX ベースのオペレーティング・システムまたは Windows NT/2000 での呼び出し形式:

```
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、Dynix/ptx、SGI、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、/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 キロバイトのオフセット・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 キロバイト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グを取る場合に限り有効です。

- OS/2 の場合の呼び出し形式

```
action drive db_alias log_path log_file indicator
```

action 有効な値は次のとおりです。 BACKUP、RESTORE、ARCHIVE、および RETRIEVE。

drive バックアップ操作では、バックアップされるデータベースが位置しているドライブを指定します。復元操作では、データベースを復元する先のドライブを指定します。アーカイブまたは検索操作では、データベースが位置しているドライブを指定します。どの場合にも、ドライブ文字の後にコロンを続けてください (たとえば、C:)。

db_alias データベースの別名を指定します (別名がない場合は、データベース名を指定します)。

log_path バックアップまたは復元操作の場合、バックアップまたは復元されるファイルのリストを含む応答ファイルの完全修飾名を指定します。リスト内の各名前は完全修飾名で、ワイルドカード文字を含めることができます。復元操作の場合、ドライブ文字およびパスは、バックアップ時の元ドライブおよびデータベース・ファイルのパスを示します。たとえば、C:¥SQLUTIL¥dbname.MH1 が応答ファイルに含まれていれば、 dbname.MH1 ファイルが C:¥SQLUTIL からバックアップされたことを意味します。

アーカイブまたは検索操作では、ログ・パスのディレクトリーを指定します。たとえば、C:¥SQL00001¥SQLOGDIR¥ のように指定します。

log_file バックアップ操作の場合は、バックアップ・ユーティリティーによって生成されたメディア・ラベルを指定します。このラベルは、データベース別名とタイム・スタンプから成っています。復元操作の場合は、ファイルが復元される先のデータベース・サブディレクトリーのパス名を指定します。ドライブ文字は、*drive* パラメーターで指定されているため、含まれません。形式は、¥SQL*nnnnn*¥ です。

アーカイブまたは検索操作では、 ログ・ファイル名を指定します。たとえば、S0000001.LOG。

indicator バックアップまたは復元操作中に、複数の呼び出しをサポートするために使用できるインディケーターを指定します。最初の呼び出しには、1 の文字値があり、それ以降の呼び出しには 2 の文字値があります。

ユーザー出口プログラムは、バックアップか復元の操作中に複数回呼び出されます。最初の呼び出しは、メディア・ヘッダー (.MH*n*) ファイルのバックアップまたは復元を行い、 2 番目の呼び出しは、データベース・ファイルの集まり全体のバックアップまたは復元を行います。

このパラメーターは、アーカイブまたは検索操作では使用されません。

バックアップおよび復元に関する考慮事項 - DB2 (OS/2 版) のみ

以下の考慮事項は、バックアップまたは復元ユーティリティーから呼び出されるユーザー出口プログラムを作成しようとしている場合に適用されます。

- ゼロ以外の戻りコードがユーザー出口プログラムから戻された場合は、ユーティリティーは失敗し、再試行は行われません。
- 完全修飾パス名には、サポートされているワイルドカード文字のみを使用してください。たとえば、`C:¥SQL00001¥*.*` および `C:¥*.MH*` はどちらも受け入れ可能な検索基準です。
- ユーザー出口プログラムは、1 行あたり 1 つの完全修飾ファイル名があり、各行が復帰および改行文字により終了する応答ファイル形式を扱う必要があります。ファイルにはファイル終わり文字はありません。
- 同一のデータベースの複数のバックアップ・イメージを 1 つのメディアにおく場合は、ユーザー出口プログラムは、復元操作中に正しいイメージを選択できる必要があります。 (457ページの『ユーザー出口プログラムの例』 にある、`db2uexit.ex2` の説明を参照してください。)
- 1 つのバックアップ装置を共用している 2 つの並行稼働バックアップ操作は、直列化される必要があります。
- バックアップ・イメージが複数のメディアにまたがる場合は、メディアを求めるプロンプトが、ユーザー出口プログラムまたはそのプログラムが呼び出すアプリケーションによって扱われる必要があります。この機能をサポートするために、バックアップおよび復元ユーティリティーはオペレーティング・システムのフォアグラウンド・セッションをオープンして、ユーザー出口プログラムを呼び出します。
- ユーザー出口プログラムは、データベース・ディレクトリー内のサブディレクトリーのバックアップを行う必要はありません。
- ユーザー出口プログラムを使用するデータベースを復元するときには、復元ユーティリティーが完全にそのデータベースを制御することが必要です。しかし、ワークステーションは、復元中でないデータベースに対して、アクティブ接続を行うことができます。
- データベースがユーザー出口プログラムによってバックアップまたは復元されており、別の操作が同じテープ装置を使用している場合、バックアップまたは復元操作は失敗する可能性があり、その場合には再始動する必要があります。この状況を避けるには、ロギングするためにユーザー出口プログラムを呼び出す他のデータベースが、バックアップか復元の操作の進行中に使用されていないことを確認するか、または装置が作動可能状態でない場合は、あとで、バックアップか復元の操作をユーザー出口プログラムが再試行するようにできます。

- 復元操作時に、ドライブ文字とパスがバックアップ操作時に指定されたものと違っていてもかまいません。たとえば、ファイル dbname.MH1 が C:¥SQLUTIL からバックアップされる場合、それを D:¥SQLUTIL2 に復元することができます。

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それらについての情報を入手してください。

OS/2 では、ゼロ以外の戻りコードがユーザー出口プログラムから戻された場合は、ユーティリティーは失敗し、再試行は行われません。このユーティリティーは汎用 SQLCODE -2029 を報告します。このメッセージ・テキストはユーザー出口プログラムまたはオペレーティング・システムによって戻されたコードを表示します。

表24 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 24. 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。 OS/2 では、このエラー・メッセージは復元操作を完了するために必要なファイルが現行のバックアップ・メディア上で検出されなかったことも意味します。[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]

表 24. 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
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]

[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 はハングします。 (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 **sqlurlog** 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、ロールフォワード回復は中断されますが、 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、問題を訂正して回復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、4、および 24 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I. ベンダー製品用のバックアップおよび復元 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復元操作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 のバックアップおよび復元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と復元を行います。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 (OS/2 または 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、バックアップまたは復元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、必要なタスクを実行するためにベンダー提供の関数が呼び出されます。

この付録は、次の 4 つの部分に分かれています。

- ベンダー製品と DB2 との対話の操作概要。
- DB2 のベンダー API の詳細記述。
- API 呼び出しで使用されるデータ構造についての情報。
- ベンダー製品を使用したバックアップおよび復元の呼び出しについての詳細。

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、 5 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

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 (OS/2 または 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復元操作で 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。
- バックアップと復元操作のどちらが行われるか。
- バックアップまたは復元操作で指定された 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復元をサポートしています。バックアップまたは復元に 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論理装置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論理装置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復元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 (474ページの『sqluvint - 初期設定と装置へのリンク』を参照してください)。

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、ベンダー製品には活動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復元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 が復元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・イメージの名前に順序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表す 1 で始まり、バックアップまたは復元操作用の **sqluvint** 呼び出しで他のセッションが開始されるたびに 1 ずつ増加します。詳細については、493ページの『INIT-INPUT』を参照してください。

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のに使用したセッションの数を DB2 へ知らせる情報が含まれます。この情報は、復元操作時に、すべてのセッションまたはデータ・ストリームが復元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 (489ページの『DB2-INFO』を参照してください)。順序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、この情報を復元時に使用し、復元されるバックアップを識別します。

復元の場合、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。順序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、順序番号 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復元されます。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、メディアの末尾を検査します。

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。復元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復元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。

PROMPTING モード

バックアップまたは復元操作の開始時には、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復元について次の 3 つの応答が可能です。

- 継続

  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復元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復元操作時に、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復元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復元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** (書きこみ) または **sqluvget** (読み取り) 呼び出しから SQLUV_ENDOFMEDIA または SQLUV_ENDOFMEDIA_NO_DATA 戻りコードを受け取ると、 DB2 は以下のことを行います。

- 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。
  - メディア終端状態を示している戻りコード構造からベンダー・メディア処理メッセージをユーザーに送る。
  - 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 は順序番号を増分させます。順序番号は、**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復元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することはありません。
- *装置終了* の場合、DB2 は別のセッションを初期設定せず、活動セッションの数が 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復元操作が完了するまで、少なくとも 1 つのセッションが活動状態のままでなければなりません。
- 応答が終了 の場合、 DB2 はバックアップまたは復元操作を終了します。セッションを終了させるときの DB2 の処理については、 470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復元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、処理の並列性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、残りの活動セッションで、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。復元操作の場合、すべてのメディアが処理されるまで、継続の応答を行うこともお勧めします。

バックアップまたは復元モードが 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復元が完了する前に、すべてのセッションが DB2 へメディア終端を戻すと、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、メディアの装てん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復元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 は消失したデータを検出する方法を持たないため、復元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、復元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、復元操作は失敗します。

エラー条件が DB2 に戻される場合

バックアップまたは復元操作時に、DB2 は、すべてのセッションが正常に完了することを想定しています。正常でない場合はバックアップまたは復元操作全体が失敗します。セッションは、**sqluvend** 呼び出し (アクション = SQLUV_COMMIT) 時に SQLUV_OK 戻りコードを使用して、正常に完了したことを DB2 に知らせます。

回復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復元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回復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 情報と共にユーザーに提供されます。

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回復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と見なされ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 構造内には順序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** の呼び出し時に、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、ユーザーに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ユーザーに表示されます。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、 DB2 は、バックアップ操作中 **sqluvput** を使用してバッファーに再書き込みを試行します。復元操作中は、 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 は回復不能エラーが生じたときと同じ方法で、バックアップまたは復元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、ベンダー製品の作成時のヒントをいくつか提供します。

リカバリー・ヒストリー・ファイル

リカバリー・ヒストリー・ファイルは、データベース回復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復元操作が行われるたびに自動的に更新されます。リカバリー・ヒストリー・ファイルについては、49ページの『リカバリー・ヒストリー・ファイルについて』 を参照してください。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、 369ページの『データ構造: 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 つ以上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た場合、ヒストリー・レコード内の DEVICE フィールドには 0 が入り、 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ければ、共用ライブラリーの名前。

このオプションの設定についての詳細は、498ページの『ベンダー製品を使用してバックアップまたは復元操作を起動する』 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 を使用して更新することができます。バックアップ・イメージを保持するの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保管場所 (オフサイトなど) へ物理的に移動された場合に、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。このような場合には、リカバリー・ヒストリー・ファイルを使用して、回復操作が必要になったときにバックアップ・イメージを見付けるのに役立てることができます。

関数およびデータ構造

以下のセクションでは、ベンダー製品で使用できる汎用関数とデータ構造について説明します。

ベンダー製品用の API は、以下のとおりです。

- 474ページの『sqluvint - 初期設定と装置へのリンク』
- 478ページの『sqluvget - 装置からのデータの読み取り』
- 481ページの『sqluvput - 装置へのデータの書き込み』
- 484ページの『sqluvend - 装置のリンク解除およびリソースの解放』
- 487ページの『sqluvdel - コミット済みセッションの削除』

ベンダー API によって使用されるデータ構造は、以下のとおりです。

489ページの『DB2-INFO』
ベンダー装置に DB2 を識別させる情報が含まれます。

492ページの『VENDOR-INFO』
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ます。

493ページの『INIT-INPUT』
DB2 とベンダー装置との間に論理リンクを設定します。

495ページの『INIT-OUTPUT』
装置からの出力が含まれます。

496ページの『DATA』
DB2 とベンダー装置の間で転送されたデータが含まれます。

497ページの『RETURN-CODE』
エラーの戻りコードと説明が含まれます。

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。

許可

以下のどれかが必要です。

- *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 (476ページの表25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復元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・レベルと関連づけて使用して、バックアップまたは復元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 (詳細については、 468ページの『PROMPTING モード』および 468ページの『装置の特性』を参照してください)。

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復元を指定します。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 25. 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	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 (データの復元) 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。
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。

表 25. *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	次の呼び出し	その他のコメント
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 (データの復元) 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。

許可

以下のどれかが必要です。

- *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) および Return_code 用に割り振られる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

この関数は復元ユーティリティーで使用されます。

戻りコード

表 26. 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	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 にメディア終端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	警告に対する DB2 の扱いについての説明 (471ページの『警告条件』) を参照してください。
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	メディア終端条件に対する DB2 の扱いについての説明は、 468ページの『PROMPTING モード』および 468ページの『装置の特性』を参照してください。

表 26. *sqluvget* 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	次の呼び出し	その他のコメント
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、468ページの『PROMPTING モード』および468ページの『装置の特性』で説明されているように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、アクション = SQLU_ABORT である場合には、このセッションおよび他のアクティブ・セッションは終了されます。

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。

許可

以下のどれかが必要です。

- *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) および Return_code 用に割り振られる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 27. 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	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。
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	メディア終端条件に対する DB2 の扱いについての説明は、 468ページの『PROMPTING モード』および 468ページの『装置の特性』を参照してください。
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 にメディア終端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	警告に対する DB2 の扱いについての説明 (471ページの『警告条件』) を参照してください。
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。

表 27. *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	次の呼び出し	その他のコメント
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、アクション = SQLU_ABORT である場合には、このセッションおよび他のアクティブ・セッションは終了されます。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 (470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)。			

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) を解放する必要があります。

許可

以下のどれかが必要です。

- *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、オープンされたそれぞれの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 (backup) 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 (restore) 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 はバックアップまたは復元操作全体に問題があると判断します。すべての活動セッションは、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 (470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)。

- 打ち切り

  DB2 によって問題が生じているときには、セッションでは読み取りまたはデータの書き込みは行われません。

  書き込み (backup) 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。すべての活動セッションは、アクション = 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 (470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)。

  読み取り (restore) 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、アクション = SQLUV_ABORT の **sqluvend** 呼び出しですべての復元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ず、後続の障害は無視されます。この場合、アクション = SQLUV_ABORT の **sqluvend** 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 28. 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	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。

許可

以下のどれかが必要です。

- *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。順序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 29. 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	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	

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 30. DB2-INFO 構造のフィールド. フィールドはすべて、ヌル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 製品を表す ID。指示する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。指示する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。重要性がなければヌルに設定します。指示するストリングの最大長は 8 文字です。
level	char	DB2 製品の現行レベルです。重要性がなければヌルに設定します。指示する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。指示する文字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。ヌルであれば、 *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。指示する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固有名。指示する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。指示する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、または実行する復元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 - 全復元 3 - オンラインの表スペース復元 4 - 表スペース復元 5 - ヒストリー・ファイル復元
dbname	char	バックアップまたは復元するデータベースの名前。指示する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復元するデータベースの別名。指示する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。指示するストリングの最大長は 26 文字です。
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。指示するストリングの最大長は 3 文字です。

表 30. DB2-INFO 構造のフィールド (続き). フィールドはすべて、ヌル終了ストリングです。

フィールド名	データ・タイプ	説明
obj_list	struct sqlu_gen_list	バックアップ・イメージ内のオブジェクトをリストします。これは、情報としてのみベンダーに提供されます。
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*、または *server_id*、 *db2instance*、*type*、 *dbname* と *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・イメージを復元するときには、同じ値を指定してバックアップ・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ヌルに設定されたり、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ます。

表 31. VENDOR-INFO 構造のフィールド. フィールドはすべて、ヌル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。指示する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。指示する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。重要性がなければヌルに設定します。指示する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。重要性がなければヌルに設定します。指示する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固有名。指示する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、復元操作時に、すべてのバックアップ・イメージが処理されたかどう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 32. INIT-INPUT 構造のフィールド. フィールドはすべて、ヌル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	options フィールドの長さ。 DB2 バックアップまたは復元機能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復元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、バイト逆転が行われず、また、コード・ページがチェックされません。 DB2 バックアップまたは復元機能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	バックアップまたは復元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。指示する文字ストリングの最大長は 1 文字です。
num_sessions	unsigned short	バックアップまたは復元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 33. 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 とベンダー装置の間で転送されるデータが含まれます。

表 34. 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順序番号。
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 へ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 35. 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、各種の DB2 API によって戻される値とは異なるベンダー固有の戻りコードです。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[60],
  void      *reserve,
} Return_code;
```

ベンダー製品を使用してバックアップまたは復元操作を起動する

DB2 バックアップまたは DB2 復元ユーティリティーを以下のものから起動するときに、ベンダー製品を指定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)
- 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 に付属してい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復元操作用のコントロール・センターの入力変数
ベンダー装置の使用とライブラリー名	*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・システムまたは OS/2 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	バックアップ用には *Size of each Buffer*、復元用には適用できません。

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のパラメーター	
	バックアップ用	復元用
ベンダー装置の使用とライブラリー名	*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*

アプリケーション・プログラミング・インターフェース (API)

2 つの API 関数呼び出しがバックアップおよび復元操作をサポートしています。バックアップ用には **sqlubkp** (93ページの『Backup Database API』 を参照)、復元用には **sqlurestore** (122ページの『Restore Database API』 を参照) です。

指定内容	API のパラメーター (sqlubkp と sqlurestore の両方)
ベンダー装置の使用とライブラリー名	構造 *sqlu_media_list* に SQLU_OTHER_MEDIA のメディア・タイプを指定し、構造 *sqlu_vendor* の *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 SQLU_OTHER_MEDIA のメディア・タイプを指定し、構造 *sqlu_vendor* の *filename* にファイル名を指定します。
転送バッファー・サイズ	*BufferSize*

付録J. DB2 ライブラリーの使用法

DB2 ユニバーサル・データベース ライブラリーは、オンライン・ヘルプ、ブック (PDF および HTML)、および HTML 形式のサンプル・プログラムから成っています。このセクションでは、ユーザーに提供される情報について紹介し、その入手方法を示します。

オンライン製品情報をご利用になるには、インフォメーション・センターを使用することができます。詳細については、517ページの『インフォメーション・センターを使用した情報へのアクセス』を参照してください。ここではタスク情報、DB2 ブック、トラブルシューティング情報、サンプル・プログラム、および Web の DB2 情報を見ることができます。

DB2 PDF ファイルおよびハードコピー版資料

DB2 情報

以下に示す表では、DB2 ブックを 4 つのカテゴリーに分類しています。

DB2 の手引きおよび解説書
これらの資料は、すべてのプラットフォームに共通の DB2 情報を含んでいます。

DB2 のインストールおよび構成の情報
これらの資料は、特定のプラットフォーム上の DB2 ごとに用意されています。たとえば、 OS/2、Windows、および UNIX ベースのプラットフォームで稼働するそれぞれの DB2 用に、別個の*概説*および*インストール* 資料が用意されています。

プラットフォーム共通のサンプル・プログラム (HTML 形式)
これらのサンプルは、アプリケーション開発クライアントとともにインストールされるサンプル・プログラムの HTML 版です。これらのサンプルは参考用であり、実際のプログラムに代わるものではありません。

リリース情報
これらのファイルには、DB2 ブックには含められなかった最新の情報が記載されています。

インストール情報、リリース情報、およびチュートリアルは、製品 CD-ROM から HTML 形式で参照することができます。ほとんどの資料は、製品 CD-ROM から HTML 形式で表示できますし、 DB2 の資料 CD-ROM から Adobe Acrobat (PDF) 形

式で表示し印刷することができます。 IBM にハードコピー版の資料を注文したい場合は、 513ページの『印刷資料の注文方法』を参照してください。注文可能な資料については、以下の表をご覧ください。

OS/2 および Windows プラットフォームの場合、 HTML ファイルは `sqllib¥doc¥html` ディレクトリーにインストールできます。 DB2 情報はいくつかの言語で提供されています。しかし、すべての言語に翻訳されているわけではありません。ある言語で情報が提供されていない場合は、英語版の情報が提供されます。

UNIX プラットフォームの場合、言語ごとに異なる複数の HTML ファイルを `doc/%L/html` ディレクトリーにインストールできます。ここで、*%L* は地域を表しています。詳細については、適切な *概説およびインストールの手引き* を参照してください。

DB2 ブックを入手して情報を利用するには、次のようなさまざまな方法があります。

- 516ページの『オンライン情報の表示』
- 521ページの『オンライン情報の検索』
- 513ページの『印刷資料の注文方法』
- 513ページの『PDF 資料の印刷』

表 36. DB2 情報

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
	DB2 の手引きおよび解説書情報		
管理の手引き	*管理の手引き: 計画* は、データベース概念について概説し、設計 (たとえば、論理および物理データベース設計) に関する情報を提供し、高い可用性について解説しています。	SC88-8513 db2d1x70	db2d0
	管理の手引き: インプリメンテーション は、設計、データベースへのアクセス、監査、バックアップ、およびリカバリーなどのインプリメンテーションについて説明しています。	SC88-8511 db2d2x70	
	管理の手引き: パフォーマンス は、データベース環境について解説し、さらにアプリケーションのパフォーマンスの評価と調整の方法について説明しています。	SC88-8512 db2d3x70	
管理 API 解説書	データベースの管理に使用できる DB2 アプリケーション・プログラミング・インターフェース (API) およびデータ構造について説明します。また、この資料は、アプリケーションから API を呼び出す方法も示します。	SC88-8514 db2b0x70	db2b0
アプリケーション構築の手引き	環境設定に関する情報を提供し、Windows、OS/2、および UNIX ベースのプラットフォームでの DB2 アプリケーションのコンパイル、リンク、実行の各ステップについて説明します。	SC88-8515 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	DB2 ユニバーサル・データベース製品をご使用中に発生する可能性のあるセンス・コード APPC、CPI-C、および SNA についての一般情報を提供します。 HTML 形式でのみご利用いただけます。	資料番号なし db2apx70	db2ap

表 36. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
アプリケーション開発の手引き	DB2 データベースにアクセスするアプリケーションを、組み込み SQL または Java (JDBC および SQLJ) を使用して開発する方法について説明します。さらに、ストアード・プロシージャーの作成方法、ユーザー定義関数の作成方法、ユーザー定義タイプの作成方法、トリガーの使用方法、区画化されている環境または統合されているシステムでのアプリケーションの開発方法などについて解説されています。	SC88-8516 db2a0x70	db2a0
コール・レベル・インターフェースの手引きおよび解説書	DB2 データベースにアクセスするアプリケーションを、 DB2 コール・レベル・インターフェース (Microsoft ODBC 仕様互換の呼び出し可能 SQL) を使用して開発する方法について説明します。	SC88-8517 db2l0x70	db2l0
コマンド解説書	コマンド行プロセッサーの使用法について説明し、データベースの管理に使用できる DB2 コマンドについて解説しています。	SC88-8518 db2n0x70	db2n0
コネクティビティー 補足	DB2 (AS/400 版)、DB2 (OS/390 版)、DB2 (MVS 版)、または DB2 (VM 版) を DRDA アプリケーション・リクエスターとして DB2 ユニバーサル・データベース とともに使用するためのセットアップ情報および参照情報を提供します。また、この資料は DRDA アプリケーション・サーバーを DB2 コネクト アプリケーション・リクエスターとともに使用する方法の詳細を示します。 HTML と PDF でのみ利用可能	資料番号なし db2h1x70	db2h1
データ移動ユーティリティー 手引きおよび解説書	データの移動を行う DB2 ユーティリティー (インポート、エクスポート、ロード、AutoLoader、および DPROP など) の使用法について説明しています。	SC88-8522 db2dmx70	db2dm

表 36. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
データウェアハウスセンター 管理の手引き	データウェアハウスセンターを使用してデータウェアハウスを構築および保守する方法を説明します。	SC88-8545 db2ddx70	db2dd
データウェアハウスセンター アプリケーション統合の手引き	プログラマーがアプリケーションをデータウェアハウスセンターおよび情報カタログ・マネージャーと統合するのに役立つ情報を提供します。	SC88-8546 db2adx70	db2ad
DB2 コネクト 使用者の手引き	DB2 コネクト製品の概念、プログラミング、および一般的な使用方法に関する情報を提供します。	SC88-8521 db2c0x70	db2c0
DB2 クエリー・パトローラー 管理の手引き	DB2 クエリー・パトローラー・システムの運用の概説を行い、運用および管理に関する詳細情報、および管理用グラフィカル・ユーザー・インターフェース・ユーティリティーについてのタスク情報を提供します。	SC88-8525 db2dwx70	db2dw
DB2 クエリー・パトローラー 使用者の手引き	DB2 クエリー・パトローラーのツールや関数の使用方法を説明します。	SC88-8527 db2wwx70	db2ww
用語集	DB2 およびそのコンポーネントで使用される用語の定義を示します。 HTML 形式と *SQL 解説書* で利用可能	資料番号なし db2t0x70	db2t0
イメージ、オーディオ、およびビデオ・エクステンダー 管理およびプログラミングの手引き	DB2 エクステンダーの一般情報について提供し、画像、音声、およびビデオ (IAV) エクステンダーの管理と構成について、および IAV エクステンダーを使用したプログラミングについて説明しています。さらに、参照情報、診断情報 (メッセージ解説)、およびサンプルも収録されています。	SC88-8609 dmbu7x70	dmbu7
情報カタログ・マネージャー 管理の手引き	情報カタログを管理するためのガイドです。	SC88-8547 db2dix70	db2di
情報カタログ・マネージャー プログラミングの手引きおよび解説書	情報カタログ・マネージャー用の体系化されたインターフェースの定義を示します。	SC88-8549 db2bix70	db2bi

表 36. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
情報カタログ・マネージャー 使用者の手引き	情報カタログ・マネージャー・ユーザー・インターフェースの使用に関する情報を提供します。	SC88-8548 db2aix70	db2ai
インストールおよび構成補足	プラットフォーム固有の DB2 クライアントの計画、インストール、およびセットアップのガイドです。この補足資料には、バインド、クライアント / サーバー通信の設定、 DB2 GUI ツール、DRDA AS、分散インストール、分散要求の構成、および異機種データ・ソースへのアクセスについても説明されています。	GC88-8524 db2iyx70	db2iy
メッセージ解説書	DB2、情報カタログ・マネージャー、およびデータウェアハウスセンターから出されるメッセージとコードをリストし、取るべき処置を解説しています。	第 1 巻 GC88-8543 db2m1x70 第 2 巻 GC88-8544 db2m2x70	db2m0
OLAP Integration Server Administration Guide	OLAP Integration Server の Administration Manager コンポーネントの使用方法を説明します。	SC27-0787 db2dpx70	n/a
OLAP Integration Server Metaoutline User's Guide	標準の OLAP Metaoutline インターフェースを使用して (Metaoutline Assistant を使用するのではなく) OLAP metaoutline を作成しデータを取り込む方法を説明しています。	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	(Model Assistant ではなく) 標準的な OLAP Model Interface を使用して OLAP モデルを作成する方法を説明します。	SC27-0783 db2lpx70	n/a
OLAP のセットアップおよびユーザーズ・ガイド	OLAP スターター・キットの構成およびセットアップに関する情報を提供します。	SC88-8652 db2ipx70	db2ip
Hyperion Essbase スプレッドシート アドイン ユーザーズ ガイド for Excel	Excel 作表計算プログラムを使用して OLAP データを分析する方法を説明します。	SC88-8724 db2epx70	db2ep

表 36. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
Hyperion Essbase スプレッドシート アドイン ユーザーズ ガイド for 1-2-3	ロータス 1-2-3 作表計算プログラムを使用して OLAP データを分析する方法を説明します。	SC88-8723 db2tpx70	db2tp
レプリケーションの手引きおよび解説書	DB2 に付属の IBM レプリケーション・ツールの計画、構成、管理、および使用方法に関する情報を提供します。	SC88-8550 db2e0x70	db2e0
地理情報エクステンダー使用者の手引きおよび解説書	地理情報エクステンダーのインストール、構成、管理、プログラミング、およびトラブルシューティングに関する情報を提供します。また、地理情報データの概念についての重要事項を示し、地理情報エクステンダー固有の参照情報 (メッセージおよび SQL) を提供します。	SC88-8624 db2sbx70	db2sb
SQL 概説	SQL の概念を紹介し、構造体とタスクの例を多数提供しています。	SC88-8539 db2y0x70	db2y0
SQL 解説書	SQL の構文、セマンティクス、および言語規則について説明します。また、この資料には、各リリース間の互換性、製品の制限事項、およびカタログ・ビューも含まれます。	第 1 巻 SC88-8540 db2s1x70 第 2 巻 SC88-8657 db2s2x70	db2s0
システム・モニター 手引きおよび解説書	データベースおよびデータベース・マネージャーに関連したさまざまな情報を収集する方法を示します。この資料は、この情報を利用して、データベース活動の把握、パフォーマンス向上、および問題原因の判別を行う方法を説明しています。	SC88-8523 db2f0x70	db2f0

表 36. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
テキスト・エクステンダー *管理およびプログラミング*	DB2 エクステンダーの一般情報、テキスト・エクステンダーの管理および構成情報、およびテキスト・エクステンダーを使用したプログラミングの方法について解説します。この資料には、参照情報、診断情報 (メッセージ解説)、およびサンプルが含まれています。	SC88-8610 desu9x70	desu9
問題判別の手引き	エラーの原因の判別、問題からの回復、および DB2 カスタマー・サービスの支援の下での診断ツールの使用法を記載しています。	GD88-7271 db2p0x70	db2p0
新機能	DB2 ユニバーサル・データベース バージョン 7 の新しい機能および拡張機能について説明します。	SC88-8541 db2q0x70	db2q0
DB2 のインストールおよび構成の情報			
DB2 コネクト エンタープライズ・エディション (OS/2 および Windows 版) 概説およびインストール	OS/2 および Windows 32 ビット オペレーティング・システム版の DB2 コネクト エンタープライズ・エディションで、計画、移行、インストール、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8520 db2c6x70	db2c6
DB2 コネクト エンタープライズ・エディション (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 コネクト エンタープライズ・エディションの計画、移行、インストール、構成、およびタスクに関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8519 db2cyx70	db2cy

表 36. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
DB2 コネクト パーソナル・エディション 概説およびインストール	OS/2 および Windows 32 ビット オペレーティング・システムの DB2 コネクト パーソナル・エディションで、計画、移行、インストール、および構成を行う場合のタスク情報を提供します。また、この資料はサポートされているすべてのクライアントのインストールおよびセットアップについても説明します。	GC88-8533 db2c1x70	db2c1
DB2 コネクト パーソナル・エディション (Linux 版) 概説およびインストール	サポートされる Linux 配布プログラムの DB2 コネクト パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8528 db2c4x70	db2c4
DB2 データ・リンク・マネージャー 概説およびインストール	AIX および Windows 32 ビット オペレーティング・システムの DB2 データ・リンク・マネージャーで、計画、インストール、構成を行う場合の情報を提供します。	GC88-8532 db2z6x70	db2z6
DB2 エンタープライズ拡張エディション (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 エンタープライズ拡張エディションの計画、インストール、および構成に関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8530 db2v3x70	db2v3
DB2 エンタープライズ拡張エディション (Windows 版) 概説およびインストール	Windows 32 ビット オペレーティング・システムの DB2 エンタープライズ拡張エディションで、計画、インストール、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8529 db2v6x70	db2v6

表 36. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
DB2 ユニバーサル・データベース (OS/2 版) 概説およびインストール	OS/2 オペレーティング・システムでの DB2 ユニバーサル・データベースの計画、インストール、移行、および構成に関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8534 db2i2x70	db2i2
DB2 ユニバーサル・データベース (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 ユニバーサル・データベースの計画、インストール、移行、および構成に関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8536 db2ixx70	db2ix
DB2 ユニバーサル・データベース (Windows 版) 概説およびインストール	Windows 32 ビット オペレーティング・システムの DB2 ユニバーサル・データベースで、計画、インストール、移行、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8537 db2i6x70	db2i6
DB2 パーソナル・エディション 概説およびインストール	OS/2 および Windows 32 ビット オペレーティング・システム 版の DB2 ユニバーサル・データベース パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8535 db2i1x70	db2i1
DB2 パーソナル・エディション (Linux 版) 概説およびインストール	サポートされる Linux 配布プログラムの DB2 ユニバーサル・データベース パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8538 db2i4x70	db2i4
DB2 クエリー・パトローラー インストールの手引き	DB2 クエリー・パトローラーのインストール情報を提供します。	GC88-8526 db2iwx70	db2iw

表 36. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
ウェアハウス・マネージャー *インストールの手引き*	ウェアハウス・エージェント、ウェアハウス・トランスフォーマー、および情報カタログ・マネージャーのインストール情報を提供します。	GC88-8572 db2idx70	db2id
プラットフォーム共通のサンプル・プログラム (HTML 形式)			
サンプル・プログラム (HTML)	DB2 のサポートするすべてのプラットフォームでのプログラム言語用に、サンプル・プログラム (HTML 形式) を提供します。これらのサンプル・プログラムは、参照用としてのみ提供されています。サンプルは、すべてのプログラミング言語で利用できるわけではありません。 HTML サンプルが利用できるのは、 DB2 アプリケーション開発クライアントがインストールされている場合だけです。 プログラムの詳細については、*アプリケーション構築の手引き* を参照してください。	資料番号なし	db2hs
リリース情報			
DB2 コネクト リリース情報	DB2 コネクトの資料には含められなかった最新の情報が収録されています。	注 #2 を参照してください。	db2cr
DB2 インストール情報	DB2 ブックには含められなかったインストールに関する最新の情報が収録されています。	製品 CD-ROM からのみ利用できます。	
DB2 リリース情報	DB2 ブックには含められなかった DB2 製品とその機能に関する最新の情報が収録されています。	注 #2 を参照してください。	db2ir

注:

1. ファイル名の 6 桁目の文字 *x* は、その資料の言語を表します。たとえば、ファイル名 `db2d0e70` は、*管理の手引き* の英語版であることを示し、ファイル名 `db2d0f70` は同じ資料のフランス語版を示します。資料の言語を表すためにファイル名の 6 桁目で使用されている文字は以下のとおりです。

言語	ID
ブラジル・ポルトガル語	b
ブルガリア語	u
チェコ語	x
デンマーク語	d
オランダ語	q
英語	e
フィンランド語	y
フランス語	f
ドイツ語	g
ギリシャ語	a
ハンガリー語	h
イタリア語	i
日本語	j
韓国語	k
ノルウェー語	n
ポーランド語	p
ポルトガル語	v
ロシア語	r
簡体字中国語	c
スロベニア語	l
スペイン語	z
スウェーデン語	s
繁体字中国語	t
トルコ語	m

2. DB2 ブックには含められなかった最新の情報が、「リリース情報」で HTML 形式および ASCII ファイルとして利用できます。 HTML 版は、インフォメーション・センターおよび製品 CD-ROM からご利用になれます。 ASCII ファイルの参照方法:
   - UNIX ベースのプラットフォームでは、ファイル `Release.Notes` を参照してください。このファイルは `DB2DIR/Readme/%L` ディレクトリーにあります。ここで `%L` はロケール名を、`DB2DIR` は以下のものを表します。
     - `/usr/lpp/db2_07_01` (AIX の場合)
     - `/opt/IBMdb2/V7.1` (HP-UX、DYNIX/ptx、Solaris、および Silicon Graphics IRIX の場合)
     - `/usr/IBMdb2/V7.1` (Linux の場合)
   - これ以外のプラットフォームでは、ファイル `RELEASE.TXT` を参照してください。このファイルは、製品がインストールされているディレクトリーにあります。OS/2 プラットフォームでは、**IBM DB2** フォルダーをダブルクリックし、**Release Notes** アイコンをダブルクリックすることもできます。

PDF 資料の印刷

資料のハードコピー版が必要な場合、 DB2 の資料 CD-ROM にある PDF ファイル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することができます。ライブラリー内の各資料のファイルについては、503ページの表36 を参照してください。

Adobe Acrobat Reader の最新版は、Adobe の Web サイト http://www.adobe.co.jp/ から入手できます。

PDF ファイルは、DB2 の資料 CD-ROM に収録されており、ファイル拡張子 PDF が付いています。 PDF ファイルにアクセスするには以下のようにします。

1. DB2 の資料 CD-ROM を挿入します。 UNIX ベースのプラットフォームの場合は、DB2 資料 CD-ROM をマウントします。マウントの手順については、*概説およびインストール* を参照してください。
2. Acrobat Reader を起動します。
3. 以下に示すいずれかの位置から必要な PDF ファイルを開きます。
   - OS/2 および Windows プラットフォームでは:

     x:¥doc¥*language* ディレクトリー。ここで、*x* は CD-ROM ドライブを、*language* は 2 桁の言語を表す国コード (たとえば、EN は英語) を示します。
   - UNIX ベースのプラットフォームでは:

     CD-ROM の */cdrom*/doc/*%L* ディレクトリー。ここで、*/cdrom* は CD-ROM のマウント・ポイントを、*%L* はロケール名を表します。

さらに、PDF ファイルを CD-ROM からローカル・ドライブまたはネットワーク・ドライブにコピーし、そこから参照することもできます。

印刷資料の注文方法

ハードコピー版の DB2 ブックは、個別に注文することができます。資料を注文するには、IBM 承認の販売業者または営業担当員に連絡してください。

DB2 オンライン文書

オンライン・ヘルプへのアクセス

すべての DB2 コンポーネントで、オンライン・ヘルプを利用できます。以下の表に、さまざまな種類のヘルプを示します。

ヘルプの種類	内容	利用方法
コマンド・ヘルプ	コマンド行プロセッサーのコマンド構文について説明します。	コマンド行プロセッサーの対話モードから、次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 コマンドのヘルプが表示されます。
クライアント構成アシスタントのヘルプ コマンド・センターのヘルプ コントロール・センターのヘルプ データウェアハウスセンターのヘルプ イベント・アナライザーのヘルプ 情報カタログ・マネージャーのヘルプ サテライト管理センターのヘルプ スクリプト・センターのヘルプ	そのウィンドウまたはノートブックで実行できるタスクについて説明します。このヘルプは、知っておく必要のある概説および前提条件に関する情報を含みます。また、ウィンドウやノートブックの制御の使用方法を示します。	ウィンドウまたはノートブックから、**「ヘルプ (Help)」**プッシュボタンをクリックするか、または **F1** キーを押します。

ヘルプの種類	内容	利用方法

メッセージ・ヘルプ	メッセージの原因、および取るべき処置を説明します。	コマンド行プロセッサーの対話モードから、次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。
		一度に 1 画面分のメッセージ・ヘルプを表示させるには、次のように入力します。
		`? XXXnnnnn | more`
		メッセージ・ヘルプをファイルに保管するには、次のように入力します。
		`? XXXnnnnn > filename.ext`
		ここで、*filename.ext* はメッセージ・ヘルプを保管するファイルを表します。

SQL ヘルプ	SQL ステートメントの構文について説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `help statement` ここで、*statement* は SQL ステートメントを表します。 たとえば、`help SELECT` と入力すると、 SELECT ステートメントのヘルプが表示されます。 **注:** UNIX ベースのプラットフォームでは、SQL ヘルプを利用できません。
SQLSTATE ヘルプ	SQL 状態およびクラス・コードについて説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `? sqlstate or ? 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` によって SQL 状態 08003 のヘルプが表示され、 `? 08` によってクラス・コード 08 のヘルプが表示されます。

オンライン情報の表示

この製品に付属のブックは、ハイパーテキスト・マークアップ言語 (HTML) ソフトコピー形式です。ソフトコピー形式では情報を検索または表示したり、ハイパーテキスト・リンクを利用して関連情報に移動したりすることができます。また、1 つの端末を超えてライブラリーを容易に共用することができます。

オンライン・ブックやサンプル・プログラムは、 HTML バージョン 3.2 仕様に準拠するすべてのブラウザーを使って表示できます。

オンライン・ブックまたはサンプル・プログラムは、次のようにして表示します。

- DB2 管理ツールを実行している場合、インフォメーション・センターを使用します。
- ブラウザーで、**「ファイル (File)」 —>「ページを開く (Open Page)」** をクリックします。次のようなページを開いて、DB2 情報に関する説明とリンクを表示してください。
  - UNIX ベースのプラットフォームでは、以下のページを開きます。

    INSTHOME/sqllib/doc/%*L*/html/index.htm

    ここで *%L* はロケール名です。
  - その他のプラットフォームでは、以下のページを開きます。

    sqllib¥doc¥html¥index.htm

    パスは DB2 がインストールされているドライブです。

    インフォメーション・センターをインストールしていない場合、 **DB2 Information** アイコンをダブルクリックしてページを開くことができます。このアイコンは、ご使用のシステムに応じて、製品のメイン・フォルダー内または Windows 「スタート」メニューにあります。

Netscape ブラウザーのインストール

システムに Web ブラウザーがインストールされていない場合、製品の箱の中にある Netscape CD-ROM から Netscape をインストールすることができます。インストールに関する詳細な説明については、以下を参照してください。

1. Netscape CD-ROM を挿入します。
2. UNIX ベースのプラットフォームでは、CD-ROM をマウントします。マウントの手順については、*概説およびインストール* を参照してください。
3. インストールの手順については、CDNAV*nn*.txt ファイルを参照します。ここで、*nn* は 2 桁の言語 ID を表します。ファイルは CD-ROM のルート・ディレクトリーにあります。

インフォメーション・センターを使用した情報へのアクセス

インフォメーション・センターを使用すると、DB2 製品情報にす早くアクセスすることができます。インフォメーション・センターは、DB2 管理ツールを使用できるすべてのプラットフォームで利用できます。

インフォメーション・センターは「インフォメーション・センター (Information Center)」アイコンをダブルクリックすることによってオープンできます。このアイコンのある場所はシステムによって異なります。メイン・プロダクト・フォルダーか Windows の**「スタート」**メニューのどちらかです。

Windows プラットフォームの DB2 では、ツールバーおよび**ヘルプ**・メニューを使用して、インフォメーション・センターにアクセスすることもできます。

インフォメーション・センターは 6 種類の情報を提供します。適切なタブをクリックすると、種類ごとに提供されているトピックが表示されます。

タスク (Tasks) DB2 を使用して実行できる主要なタスク。

参照 (Reference)
DB2 参照情報 (キーワード、コマンド、API など)。

ブック (Books) DB2 ブック。

トラブルシューティング (Troubleshooting)
エラー・メッセージのカテゴリーと、メッセージに対するリカバリー処置。

サンプル・プログラム (Sample Programs)
DB2 アプリケーション開発クライアントに付属のサンプル・プログラム。 DB2 アプリケーション開発クライアントをインストールしていない場合、このタブは表示されません。

Web WWW 上にある DB2 情報。この情報にアクセスするには、ご使用のシステムから Web への接続が必要です。

リストから項目を 1 つ選択すると、インフォメーション・センターはビューアーを立ち上げて情報を表示します。選択した情報の種類に応じて、ビューアーはシステム・ヘルプ・ビューアー、エディター、または Web ブラウザーです。

インフォメーション・センターには検索機能が備わっており、リストを参照せずに特定のトピックを探すことができます。

テキストの全検索を行うには、インフォメーション・センター内のハイパーテキスト・リンク「**DB2 オンライン情報の検索 (Search DB2 Online Information)**」検索フォームに従います。

通常、HTML 検索サーバーは自動的に始動します。 HTML 情報の検索がうまくいかない場合は、以下の方法の 1 つを使用して、検索サーバーを始動しなければならない場合もあります。

Windows では

「スタート」をクリックし、**「プログラム」—>「IBM DB2」—> 「Information」—> 「Start HTML Search Server」**を選択します。

OS/2 では

「DB2 (OS/2 版)」フォルダーをダブルクリックして、**「Start HTML Search Server」**アイコンをダブルクリックします。

HTML 情報の検索でこの他の問題が発生した場合は、リリース情報を参照してください。

注: 検索機能は、Linux、DYNIX/ptx、および Silicon Graphics IRIX 環境では利用できません。

DB2 ウィザードの使用

ウィザードを使用すると、各タスクをステップごとに進めることによって、さまざまな管理タスクを遂行することができます。ウィザードは、コントロール・センターおよびクライアント構成アシスタントを通して使用できます。以下の表では、ウィザードとその目的をリストしています。

注: データベース作成、索引作成、マルチサイト更新の構成、およびパフォーマンス構成ウィザードは、区分データベース環境で使用できます。

ウィザード	内容	利用方法
データベース追加 *(Add Database)*	クライアント・ワークステーション上にデータベースのカタログを作成します。	クライアント構成アシスタントから、「**追加 (Add)**」をクリックします。
データベース・バックアップ *(Back up Database)*	バックアップ計画を決定、作成、およびスケジュールします。	「コントロール・センター (Control Center)」からバックアップするデータベースを右クリックし、**「バックアップ (Backup)」—> 「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。

ウィザード	内容	利用方法
マルチサイト更新の構成 (Configure Multisite Update)	マルチサイト更新、分散トランザクション、または 2 フェーズ・コミットを構成します。	「コントロール・センター (Control Center)」から、**「データベース (Databases)」**フォルダーを右クリックして、**「マルチサイト更新 (Multisite Update)」**を選択します。
データベース作成 (Create Database)	データベースを作成し、いくつかの基本的な構成タスクを実行します。	「コントロール・センター (Control Center)」から、**「データベース (Databases)」**フォルダーを右クリックして、**「作成 (Create)」—> 「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。
表作成 (Create Table)	基本的なデータ・タイプを選択して、表の基本キーを作成します。	「コントロール・センター (Control Center)」から、**「表 (Tables)」**アイコンを右クリックして、**「作成 (Create)」—>「ウィザードを使用する表 (Table Using Wizard)」**を選択します。
表スペース作成 (Create Table Space)	新しい表スペースを作成します。	「コントロール・センター (Control Center)」から、**「表スペース (Table Spaces)」**アイコンを右クリックして、**「作成 (Create)」—> 「ウィザードを使用する表スペース (Table Space Using Wizard)」**を選択します。
索引作成 (Create Index)	すべての照会について、作成すべき索引および除去すべき索引を提案します。	「コントロール・センター (Control Center)」から、**「索引 (Index)」**アイコンを右クリックして、**「作成 (Create)」—> 「ウィザードを使用する索引 (Index Using Wizard)」**を選択します。

ウィザード	内容	利用方法
パフォーマンス構成 *(Performance Configuration)*	ビジネス要件に適合するように構成パラメーターを更新して、データベースのパフォーマンスを調整します。	「コントロール・センター (Control Center)」から、調整したいデータベースを右クリックして、**「ウィザードを使用するパフォーマンスの構成 (Configure Performance Using Wizard)」**を選択します。 区分データベース環境では、「Database Partitions」視点から、調整したい最初のデータベース区画を右クリックして、**「ウィザードを使用するパフォーマンスの構成 (Configure Performance Using Wizard)」**を選択します。
データベース復元 *(Restore Database)*	障害の後、データベースを回復します。どのバックアップを使用し、どのログを再生するかの判別を支援します。	「コントロール・センター (Control Center)」から復元するデータベースを右クリックし、**「復元 (Restore)」—>「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。

文書サーバーのセットアップ

デフォルトでは、DB2 情報はローカル・システムにインストールされます。つまり、DB2 情報にアクセスする必要のある各担当者が同じファイルをインストールする必要があります。 DB2 情報を 1 か所に格納するには、次のようにします。

1. `¥sqllib¥doc¥html` のすべてのファイルとサブディレクトリーを、ローカル・システムから Web サーバーにコピーします。各ブックには独自のサブディレクトリーがあり、そのブックを構成する必要な HTML および GIF ファイルが入っています。ディレクトリー構造は常に同じ状態に保つ必要があります。
2. Web サーバーを構成して、ファイルを新しい場所で検索するようにします。さらに詳しい情報については、 *インストールおよび構成 補足* の NetQuestion 付録を参照してください。
3. インフォメーション・センターの Java バージョンをご使用の場合は、すべての HTML ファイルのベース URL を指定できます。この URL はブックのリストに使用してください。
4. 資料ファイルが表示されるようになったなら、よく使うトピックにはブックマークを付けておいてください。ブックマークを付けるページは、たとえば以下のものがあります。
   - ブックのリスト

- 頻繁に使用されるブックの目次
- 頻繁に参照する情報 (たとえば、ALTER TABLE トピックなど)
- 検索フォーム

中央のマシンから DB2 ユニバーサル・データベース オンライン文書ファイルを提供する方法については、 *インストールおよび構成 補足* の NetQuestion 付録を参照してください。

オンライン情報の検索

HTML ファイルの情報を検索するには、以下の方法のどれか 1 つを使用してください。

- 最上部にある**「検索 (Search)」**をクリックします。検索フォームを使用して特定のトピックを見つけます。この機能は、Linux、DYNIX/ptx、または Silicon Graphics IRIX 環境ではご利用になれません。
- 最上部にある**「索引 (Index)」**をクリックします。索引を使用して、ブック内の特定のトピックを見つけます。
- HTML 資料またはヘルプの目次あるいは索引を表示してから、 Web ブラウザーの検索機能を利用して資料内の特定のトピックを見つけます。
- Web ブラウザーのブックマーク機能を使用して、特定のトピックにす早く戻ります。
- インフォメーション・センターの検索機能を使用して、特定のトピックを検索します。詳しくは、517ページの『インフォメーション・センターを使用した情報へのアクセス』を参照してください。

付録K. 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、プログラムまたはサービスの操作性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。実施権の許諾については、下記の宛先に書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
AP 事業所
IBM World Trade Asia Corporation
Intellectual Property Law & 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。

本書は定期的に見直され、必要な変更 (たとえば、技術的に不適切な表現や誤植など) 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 Ltd.
Office of the Lab Director
1150 Eglinton Avenue. East
Toronto, Ontario
M3C 1H7
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・プログラムのすべての部分、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。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All Rights Reserved.

商標

アスタリスク (*) 付きの以下の用語は、IBM Corporation の商標です。

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

以下は、それぞれ各社の商標または登録商標です。

Tivoli および NetView は Tivoli Systems Inc. の商標です。

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。

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。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の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]

T

U

V

W

X

IBM と連絡をとる

技術上の問題がある場合は、時間をとって「問題判別の手引き」に定義されている処置を検討し、それらの提案を実行した後で、お客様サポートに連絡をとってください。この資料には、お客様サポートがお客様を支援するために必要とする情報が説明されています。

製品情報

以下の情報は英語で提供されます。内容は英語版製品に関する情報です。

http://www.ibm.com/software/data/

DB2 World Wide Web ページには、ニュース、製品説明、研修スケジュールなどの DB2 に関する最新情報が提供されています。ただし、提供されている情報は英語です。

http://www.ibm.com/software/data/db2/library/

「DB2 Product and Service Technical Library」では、よくされる質問 (FAQ)、修正内容、資料、および最新の DB2 技術情報などの情報へのアクセスが提供されています。

注: この情報のご提供は英語のみとなりますのでご注意ください。

http://www.elink.ibmlink.ibm.com/pbl/pbl/

「International Publications」注文用 Web サイトでは、マニュアルの注文方法についての情報を提供しています。ただし、提供されている情報は英語です。

http://www.ibm.com/education/certify/

IBM の「Professional Certification Program」Web サイトでは、 DB2 を含むさまざまな IBM 製品の認証テストの情報を提供しています。ただし、提供されている情報は英語です。

ftp.software.ibm.com

匿名でログオンしてください。ディレクトリー `/ps/products/db2` には、 DB2 および多数の他製品に関連したデモ、修正プログラム、情報、およびツールがあります。ただし、提供されている情報は英語です。

comp.databases.ibm-db2, bit.listserv.db2-l

これらのインターネット・ニュースグループは、ユーザーが DB2 製品に関する自分の経験について話し合うために利用できます。ただし、提供されている情報は英語です。

Compuserve: GO IBMDB2

このコマンドを入力すると、IBM DB2 Family forum にアクセスできます。すべての DB2 製品が、このフォーラムでサポートされています。ただし、提供されている情報は英語です。

米国以外の国で IBM に連絡する方法については、 *IBM Software Support Handbook* の Appendix A を参照してください。この資料にアクセスするには、Web ページ: http://www.ibm.com/support/ にアクセスし、ページの最下部にある「IBM Software Support Handbook」リンク・ボタンを選択します。

注: 国によっては、IBM が承認している販売業者が、 IBM サポート・センターの代わりにそれら販売業者のサポート・センターに連絡する場合があります。

Printed in Japan

SC88-9019-00

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