IBM DB2 10.1
for Linux, UNIX, and Windows

データ・リカバリーと高可用性ガイドおよびリファレンス

SA88-4694-00
(英文原典：SC27-3870-00)

IBM DB2 10.1
for Linux, UNIX, and Windows

データ・リカバリーと高可用性ガイドおよびリファレンス

SA88-4694-00
(英文原典：SC27-3870-00)

ご注意

本書および本書で紹介する製品をご使用になる前に、549 ページの『付録 B. 特記事項』に記載されている情報をお読みください。

原典： SC27-3870-00
IBM DB2 10.1
for Linux, UNIX, and Windows
Data Recovery and High Availability
Guide and Reference

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.4

目次

本書について

「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」では、DB2® Database for Linux, UNIX, and Windows データベース・ソリューションの高可用性を維持する方法、およびデータの損失を防ぐ方法について説明しています。

「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」は以下の 2 部で構成されています。

- 第 1 部の『高可用性』では、データベース・ソリューションの高可用性を維持するうえで役立つ、手法および DB2 データベース・フィーチャーと機能を説明します。
- 第 2 部の『データ・リカバリー』では、DB2 のバックアップおよびリストア機能を使用してデータの損失を防ぐ方法について説明します。

第 1 部 高可用性

データベース・ソリューションの可用性とは、ユーザー・アプリケーションが必要なデータベース・タスクをどの程度正常に実行することができるかを示す尺度です。ユーザー・アプリケーションがデータベースに接続できない場合、またはトランザクションがエラーのために失敗する、またはシステム上の負荷のためにタイムアウトになる場合は、データベース・ソリューションの可用性は高くありません。ユーザー・アプリケーションがデータベースに正常に接続して作業を実行している場合は、データベース・ソリューションのは可用性は高いと言えます。

可用性の高いデータベース・ソリューションを設計したり、または既存のソリューションの可用性を高めるには、データベースにアクセスするアプリケーションのニーズを理解しなければなりません。追加のストレージ・スペース、より高速なプロセッサー、またはより多くのソフトウェア・ライセンスに対する投資から最大の利益を得るためには、データベース・ソリューションが、ビジネスのために最も大切なアプリケーションが必要とするときに、必要な可用性を持つようにすることに重点を置いてください。

計画外の停止

ユーザーに対するデータベース・ソリューションの可用性に影響を与えることのある予期されないシステム障害には、電源中断、ネットワーク障害、ハードウェア障害、オペレーティング・システムまたは他のソフトウェアのエラー、災害時のシステム全体の障害などがあります。ユーザーがデータベースでの作業を必要とするときにそのような障害が発生する場合、高可用性のデータベース・ソリューションでは以下を行う必要があります。

- ユーザー・アプリケーションを障害から保護して、ユーザー・アプリケーションが障害を関知しないようにします。例えば、データベース・サーバーに障害が生じた場合、 DB2 Data Server はデータベース・クライアント接続を代替データベース・サーバーにリルートすることができます。
- 障害の影響が広まらないように対応します。例えば、クラスター内の 1 つのマシンで障害が生じた場合、以降のトランザクションが障害の生じたマシンに経路指定されて処理されることがないように、クラスター・マネージャーはそのマシンをクラスターから除去することができます。
- 障害から回復して、システムを正常な動作に戻します。例えば、障害の生じた 1 次データベースのデータベース操作をスタンバイ・データベースがテークオーバーする場合、障害の生じたデータベースは再始動、リカバリーすることによって、再度 1 次データベースとしてテークオーバーする可能性があります。

これら 3 つのタスクは、ユーザー・アプリケーションに対するソリューションの可用性に及ぶ影響を最小にとどめるために実行する必要があります。

計画停止

可用性の高いデータベース・ソリューションでは、保守活動がユーザー・アプリケーションに対するデータベースの可用性に与える影響も最小になるようにする必要があります。

例えば、データベース・ソリューションが午前 9 時に開店して、午後 5 時まで営業する従来型のストアにサービスを提供する場合、保守活動は、ユーザー・アプリケーションのデータベースの可用性に影響を与えない営業時間外にオフラインで行うことができます。データベース・ソリューションが、顧客がインターネットを介して一日 24 時間アクセスすることが予期されるオンライン・バンキング業務にサービスを提供する場合は、保守活動をオンラインで実行するか、または活動がオフピークとなる時間にスケジュールして、顧客のデータベースの可用性に与える影響が最小になるようにする必要があります。

ビジネス上の決定を行い、データベース・ソリューションの可用性に関する設計を選択する場合、以下の 2 つの要素を比較考慮する必要があります。

- 顧客がデータベースを利用できないことから生じるビジネス上のコスト
- 特定のレベルの可用性を実装するためのコスト

例えば、データベース・ソリューションが顧客にサービスを提供しているときには毎時一定量の収入 X を生じさせる、インターネット・ベースのビジネスについて考えます。高可用性ストラテジーによって年間のダウン時間を 10 時間短縮できるなら、ビジネスには年間で X に 10 を乗じた追加収入が生じます。この高可用性ストラテジーを実装するコストが、予想される追加収入よりも低い場合は、実装する価値があります。

第 1 章 停止

停止とは、データベース・ソリューションがユーザー・アプリケーションにサービスを提供できなくなることです。停止は、計画外の停止と計画停止の 2 つのグループに分類できます。

計画外の停止

計画外の停止の例には、以下のものがあります。

- ハードウェア障害やソフトウェア障害などの、システムの 1 つのコンポーネントの障害。
- ビジネスに重要なトランザクションのために必要な表を間違ってドロップしてしまうなど、管理上のまたはユーザー・アプリケーションの無効なアクション。
- 構成が最適でないかハードウェアやソフトウェアが不適切なためにパフォーマンスが悪い。

計画停止

計画停止の例には、以下のものがあります。

- 保守。保守アクティビティーによっては、完全に停止することが必要なものがあります。データベースを停止することなく実行できる保守アクティビティーもありますが、パフォーマンスに悪影響を及ぼす場合があります。後者は、最も一般的なタイプの計画停止です。
- アップグレード。ソフトウェアまたはハードウェアをアップグレードすると、部分的または全体的な停止が必要になることがあります。

可用性について考慮する場合、災害時のシナリオやコンポーネントの障害が中心になりがちです。しかし、耐久力のある高可用性ソリューションを設計するには、これらのすべてのタイプの停止を考慮に入れる必要があります。

停止徴候

停止徴候とは、停止の特徴となる症状や動作の集合のことです。停止の徴候となるものは、エンド・ユーザーへの応答時間が長くなるといった一時的なパフォーマンスの問題から、サイトが完全に停止してしまうことまでさまざまです。停止の回避や最小化、および停止からの回復のための方法を考案するにあたって、このようなさまざまな停止が業務にどのように影響するかを考慮してください。

ブラックアウト

ブラックアウトは、エンド・ユーザーがシステムを完全に使用できなくなってしまう場合に発生する停止のタイプです。このタイプの停止は、ハードウェア、オペレーティング・システム、またはデータベース・レベルでの問題によって引き起こされる場合があります。ブラックアウトが発生した場合、直ちに停止の範囲を見分けることが必要です。停止が単にデータベース・レベルのものなのか、インスタンスのレベルでなのか、オペレーティング・システムやハードウェアのレベルでなのかを見極めてください。

ブラウンアウト

ブラウンアウトは、エンド・ユーザーが効率的に作業できないところまでシステム・パフォーマンスが悪くなったときの停止のタイプです。システムは全体としては稼働中ですが、エンド・ユーザーの立場からすると期待どおりに動いていません。このタイプの停止は、システム保守時間枠内やシステムの使用がピークに達しているときに発生する場合があります。普通はこうした停止の間、CPU やメモリーは限界近くに達しています。不適切にチューニングされたサーバーや過度に使用されるサーバーは、しばしばブラウンアウトの原因となります。

停止の頻度と継続時間

データベースの可用性については、多くの場合、一定の期間内のダウン時間 (逆に考えれば、データベース・システムが使用可能である時間) の合計や割合が中心になって考慮されます。しかしながら、計画停止や計画外の停止の頻度や継続時間は、こうした停止が業務に及ぼす影響に大きな違いをもたらします。

例えば、7 時間かかるアップグレードをデータベース・システムに対して実行する必要があり、ユーザー活動が少ない時間帯に毎日 1 時間ずつデータベース・システムをオフラインにするか、最も忙しい日の最も忙しい時間帯に 7 時間にわたってデータベースをオフラインにするかを選べるとします。明らかに、1 回で 7 時間にわたって停止するよりも何回かに分けて短い時間停止する方が、業務活動が受ける影響や被害は少ないでしょう。次に、断続的に、おそらく毎週合計で数分にわたってネットワーク障害が発生するため、定期的にわずかな数のトランザクションが失敗する場合を考えます。こうした非常に短い停止であっても、大きな収益の損失につながる可能性があり、さらには顧客の信頼を取り返しがつかないほど失うことによって、将来の非常に多くの収益を損失する結果になる可能性があります。

停止 (または使用可能な) 時間の合計だけを考えてはなりません。保守活動について決定するときや、計画外の停止に対応するときには、少数の長時間にわたる停止と複数の短時間の停止の影響をよく見定めてください。停止が発生している最中に、こうした判断を行うことは困難な場合があります。それで、さまざまな停止徴候の影響を計算する定式や方式を作成し、最善の選択ができるようにしてください。

複数の連鎖する障害

停止の回避や最小化、および停止からの回復のためにデータベース・ソリューションを設計する際には、複数のコンポーネントに同時に障害が起こる可能性や、さらには 1 つのコンポーネントの障害が他のコンポーネントの障害を引き起こす可能性を考慮に入れてください。

停止のコスト

停止のコストは企業によって異なります。ベスト・プラクティスは、企業ごとにその基幹業務のビジネス・プロセスに対する停止のコストを分析することです。このような分析の結果は、修復計画を策定するのに使用されます。複数の処理が関係する場合、この計画には、複数の修復活動の間で優先順位を付けることが含まれます。

停止のコスト

データベース・システムが顧客の取り引きを処理することができない状態に直面した場合に、企業に対する顧客のコストを見積もることができます。例えば、データベース・システムが使用不能なために失われる売上高の、時間当たりまたは分当たりの平均コストを計算することができます。顧客の信頼を失う結果としての収益の減少を計算するのはもっと難しいですが、企業の可用性要件を査定するにはこのコストを考慮に入れるべきです。

社内データベース・システムが業務処理に使用できなくなることのコストも考慮してください。 E メールやカレンダー・ソフトウェアが 1 時間使えなくなるといった程度の単純なことでさえ、社員が仕事をできなくなってしまうため、業務を止めてしまう場合があります。

停止の許容度

停止の許容度は企業によって異なります。ベスト・プラクティスは、企業ごとにその基幹業務のビジネス・プロセスに対する停止の影響を分析することです。このような分析の結果は、修復計画を策定するのに使用されます。この計画には、複数のプロセスが関係する場合に修復を行う優先順位が含まれます。

停止の許容度

企業の可用性のニーズを決定する重大な要素は、業務または業務内の特定のシステムが停止の発生に対してどれほど許容できるかを考えることです。例えば、あるレストランが主にメニュー情報を公表するために Web サイトを運用している場合、サーバーが時々停止しても売上にはそれほど影響しないでしょう。それに対し、株取引を記録する証券取引所のサーバーが停止したら、その影響は破壊的なものでしょう。このように、多大のリソースを動員してレストランのサーバーの可用性を 99.99% にしてもあまり費用対効果はありませんが、証券取引所の場合は十分に価値があります。

許容度を考慮するに当たり、リカバリーにかかる時間とリカバリー・ポイントという 2 つの概念を考えることが必要です。

リカバリーにかかる時間とは、ビジネス・プロセスまたはシステムをオンラインに戻すのに必要な時間のことです。

リカバリー・ポイントとは、ビジネス・プロセスまたはシステムをリストアする時系列上のポイントのことです。データベースの観点から言うと、計画では、いくらかのトランザクションを失うものの迅速にリストアすることの利点と、トランザクションをまったく失わずに長い時間をかけて完全リストアを実行することを比較考慮する必要があるでしょう。

リカバリーと回避のストラテジー

可用性に関して購入やシステム設計の選択を考慮するとき、高可用性フィーチャーやテクノロジーの長いリストに飛びついてしまう傾向があるかもしれません。しかしながら、システムに高可用性を与えてそれを維持するためのベスト・プラクティスは、テクノロジーを購入することよりも、設計と構成に関してよい選択をし、健全な管理手順と緊急時計画を設計して実践することが関係しています。

投資に見合った最も包括的な高可用性を手に入れるためには、まず企業の必要に最適の高可用性ストラテジーを識別します。それから、最も適切なテクノロジーを選択して、そのストラテジーを実施することができます。

高可用性のためにデータベース・ソリューションを設計または構成するときには、停止をどのように回避するか、停止の影響をどのように最小化するか、システムを迅速に回復するにはどうしたらよいかを考慮してください。

停止の回避

可能な限り、停止を回避します。例えば、計画外の停止を回避するために単一障害点を除去したり、計画停止を回避するために保守活動をオンラインで実行する方法を調査したりすることができます。データベース・システムをモニターして、問題を示すシステムの振る舞いの傾向を識別し、停止を引き起こす前に問題を解決します。

停止の影響の最小化

計画停止や計画外の停止の影響が最小になるように、データベース・ソリューションを設計し構成することができます。例えば、データベース・ソリューションを分散してコンポーネントと機能がローカルにあるようにすれば、1 つのコンポーネントがオフラインになってもユーザー・アプリケーションによってはトランザクションの処理を継続できます。

計画外の停止からの迅速なリカバリー

リカバリー計画を作成します。計画外の停止の場合に、管理者たちが簡単かつ迅速に実行できるはっきりした手順を分かりやすく文書化します。関係するシステムのコンポーネントすべてを記述した分かりやすい設計書を作成します。サービス契約および連絡先情報をよく整理してすぐ分かるようにしておきます。迅速なリカバリーは非常に重要ですが、それと同時に、停止の根本原因を識別して将来の再発を避けるためにどんな診断情報を収集すべきかも分かっていなければなりません。

第 2 章 高可用性ストラテジー

ユーザーにとっては、データベース要求が失敗した理由は重要ではありません。パフォーマンスが低いためにトランザクションがタイムアウトになったとしても、ソリューションのコンポーネントに障害が生じたとしても、または管理者が保守を行うためにデータベースをオフラインにしたとしても、ユーザーにとって結果は同じです。要求の処理にデータベースを使用できなくなります。

データベース・ソリューションの可用性を改良するためのストラテジーには、以下のものがあります。

冗長度 ソリューションの各コンポーネントの 2 次コピーを用意し、障害の発生時にワークロードをテークオーバーできるようにします。

システム・モニター
ソリューションのコンポーネントに関する統計を収集して、ワークロード・バランシングやコンポーネント障害の検出を促進します。

ロード・バランシング
ソリューションの過負荷となったコンポーネントから、ソリューションの負荷の軽い別のコンポーネントに、ワークロードの一部を転送します。

フェイルオーバー
ソリューションの障害の生じたコンポーネントから 2 次コンポーネントにすべてのワークロードを転送します。

パフォーマンスの最大化
トランザクションが完了に非常に長い時間を要したり、タイムアウトしたりする可能性を低くします。

保守の影響の最小化
自動保守活動および手動保守活動をスケジューリングして、ユーザー・アプリケーションに対する影響を最小にします。

冗長度による高可用性

高可用性を維持するための重要なストラテジーは、冗長コンポーネントを持つことです。コンポーネントに障害が発生した場合、そのコンポーネントの 2 次つまりバックアップ・コピーがテークオーバーできるので、データベースはユーザー・アプリケーションに対して引き続き使用可能になります。システムのコンポーネントに冗長度がない場合、そのコンポーネントはシステムの Single Point of Failure になるおそれがあります。

冗長性をもたせることはシステム設計において一般的です。

- 中断されない、またはバックアップ機能を持つ電源機構
- 各コンポーネント間での複数のネットワーク・ファイバー
- ネットワーク・カードの結合またはロード・バランシング
- 冗長度のある配列での複数のハード・ディスク
- CPU のクラスター

システムのこれらのコンポーネントのいずれか 1 つに冗長度がない場合、そのコンポーネントはシステム全体にとっての障害原因となることがあります。

2 つのデータベースを用意することにより、データベース・レベルで冗長性を持たせることができます。つまり、1 次データベースで、全てまたは大部分のアプリケーション・ワークロードを通常に処理させ、もし 1 次データベースに障害が生じた場合に、2 次データベースでワークロードをテークオーバーさせる方法です。 DB2 高可用性災害時リカバリー (HADR) 環境で、この 2 次データベースはスタンバイ・データベースと呼ばれます。

DB2 Connect™ クライアントの場合、DB2 for z/OS® サーバー上の Sysplex ワークロード・バランシング機能は、データ共有グループに直接接続するクライアント・アプリケーションのための高可用性を実現します。 Sysplex ワークロード・バランシング機能は、ワークロード・バランシングおよびシームレスな自動クライアント・リルート機能を提供します。このサポートは、Java クライアント (JDBC、SQLJ、または pureQuery) あるいは他のクライアント (ODBC、CLI、.NET、OLE DB、PHP、Ruby、または組み込み SQL) を使用するアプリケーションで使用できます。

フェイルオーバーによる高可用性

フェイルオーバーは、1 次システム上で障害が発生したときに、ワークロードを 1 次システムから 2 次システムに転送することです。ワークロードがこのように転送された場合、 2 次システムは障害の生じた 1 次システムのワークロードをテークオーバーしたといいます。

例 1

クラスター環境で、クラスター内の 1 つのマシンに障害が生じた場合、クラスター管理用のソフトウェアは障害の生じたマシン上で実行していたプロセスをクラスター内の別のマシンに移動できます。

例 2

複数の IBM® Data Server を使用するデータベース・ソリューションでは、1 つのデータベースが使用不可となった場合、データベース・マネージャーは使用できなくなったデータベース・サーバーに接続されていたデータベース・アプリケーションを 2 次データベース・サーバーに転送します。

現在市場で使用されている最も一般的な 2 つのフェイルオーバー・ストラテジーは、アイドル・スタンバイおよび相互テークオーバーとして知られています。

アイドル・スタンバイ

この構成では、1 次システムがすべてのワークロードを処理する一方で、2 次またはスタンバイ・システムは、 1 次システムに障害が生じたときにワークロードをテークオーバーできるようにアイドルまたはスタンバイ・モードになります。高可用性災害時リカバリー (HADR) セットアップでは、最高 3 つのスタンバイを持つことができ、各スタンバイで読み取り専用ワークロードを許可するように構成できます。

相互テークオーバー

この構成では、複数のシステムが存在して、各システムがそれぞれ別のシステムの指定された 2 次システムになっています。システムに障害が生じると、障害が生じたシステムに対する 2 次システムはそれ自体のワークロードの処理を続行しながら障害が生じたシステムのワークロードも処理する必要があるため、全体的なパフォーマンスは負の影響を受けます。

クラスタリングによる高可用性

クラスターとは、単一システムとして協調して稼働する、接続された複数のマシンのグループのことです。クラスター内のマシンに障害が生じると、クラスター管理用のソフトウェアは障害の生じたマシンのワークロードを他のマシン上に転送します。

ハートビート・モニター

クラスター内の 1 つのマシンに生じた障害を検出するには、フェイルオーバー・ソフトウェアは、ハートビート監視またはキープアライブ・パケットをマシン間で使用して、クラスター内のシステムが稼働していることを確認することができます。ハートビート監視は、クラスター内のすべてのマシン間で常時通信し続けるシステム・サービスを含んでいます。ハートビートが検出されない場合、バックアップ・マシンへのフェイルオーバーが開始します。

IP アドレス・テークオーバー

クラスター内の 1 つのマシンに障害が生じた場合、クラスター・マネージャーは IP アドレスを一方のマシンから他方のマシンに転送することによって、ワークロードを一方のマシンから他方のマシンに転送できます。これは IP アドレス・テークオーバー、または IP テークオーバーと呼ばれます。この転送はクライアント・アプリケーションからは認識されないので、クライアント・アプリケーションは IP アドレスのマップ先となる物理マシンの変更に気付くことなく、元の IP アドレスを継続して使用します。

DB2 高可用性 (HA) フィーチャーによって、IBM Data Server とクラスター管理ソフトウェアの統合が可能になります。

データベース・ロギング

データベース・ログによって障害からのリカバリーが可能になり、1 次データベースと 2 次データベースの同期が可能になるため、データベース・ログは高可用性データベース・ソリューション設計での重要な部分を成します。

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

2 つのタイプ (*循環* およびアーカイブ) のデータベース・ロギングがサポートされています。それぞれのタイプは、異なるレベルのリカバリー機能を提供します。

- 10 ページの『循環ロギング』
- 11 ページの『アーカイブ・ロギング』

アーカイブ・ロギングを選択する利点は、ロールフォワード・リカバリーでは、アーカイブ・ログとアクティブ・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、データベースをリストアできるという点です。アーカイブ・ログ・ファイルは、バックアップを取ったあとに変更内容をリカバリーするために使用されます。これは、バックアップ時までの状態しかリカバリーできず、その後のすべての変更内容が失われる循環ロギングとは異なります。

循環ロギング

循環ロギングは、新規のデータベースが作成されるときのデフォルト動作です。(**logarchmeth1** および **logarchmeth2** データベース構成パラメーターは OFF に設定されます。) このタイプのロギングでは、データベースの全オフライン・バックアップのみが許可されます。全バックアップを作成するとき、データベースはオフライン (ユーザーからアクセス不能) でなければなりません。

名前から分かるとおり、循環ロギングはオンライン・ロギングの*輪*を使用して、トランザクション障害およびシステム破損からのリカバリーを提供します。ログは、現行トランザクションの整合性を保証するためにのみ使用および保持されます。循環ロギングでは、最後に行った全バックアップ操作より後に実行されたトランザクションを使用してデータベースをロールフォワードすることはできません。最後のバックアップ操作以降に加えられた変更はすべて失われます。このタイプのリストア操作では、全バックアップが取られた時点までデータがリカバリーされるため、これは、*バージョン・リカバリー* と呼ばれています。

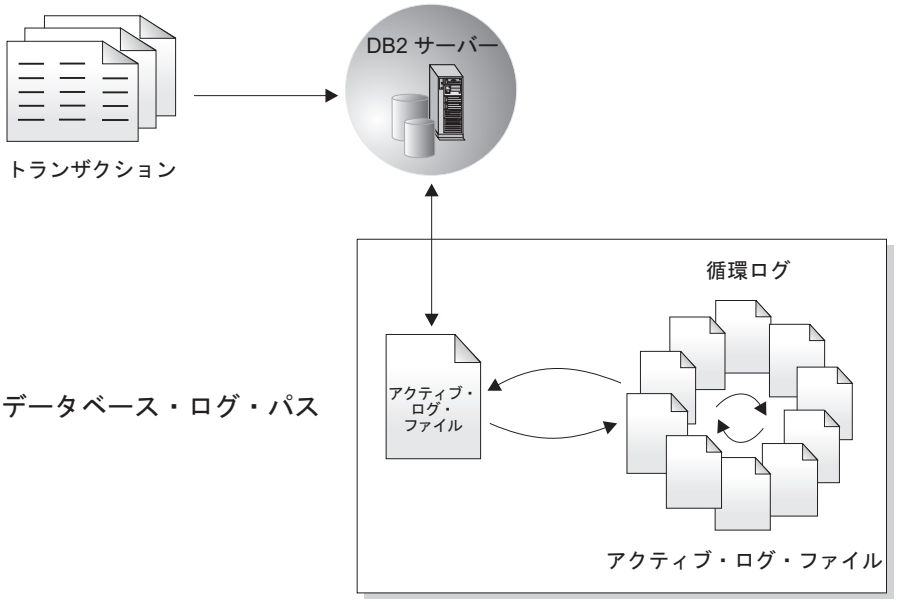


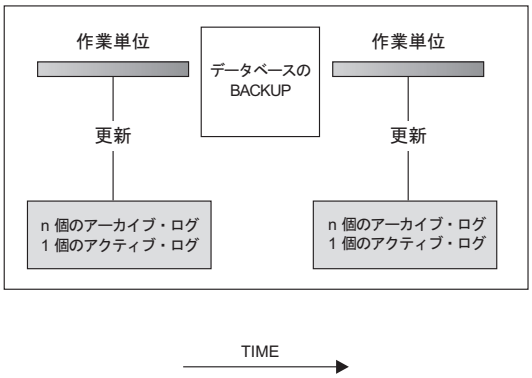
図 1. 循環ロギング

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

アーカイブ・ロギング

アーカイブ・ロギングは、特にロールフォワード・リカバリーに使用されます。アーカイブ・ログとは、現行のログ・パス、またはミラー・ログ・パスから別の場所にコピーされたログ・ファイルのことです。

logarchmeth1 データベース構成パラメーターまたは **logarchmeth2** データベース構成パラメーター、またはその両方を使用して、ユーザーまたはデータベース・マネージャーがログのアーカイブ処理を管理することができます。



ログは、データベースへの変更をトラッキングするためにバックアップ間で使用される。

図 2. ロールフォワード・リカバリーでのアクティブ・ログおよびアーカイブ・データベース・ログ: 長時間実行しているトランザクションの場合には、複数のアクティブ・ログが生じる可能性があります。

データベースをアーカイブ・ロギング用に構成している場合にのみ、オンライン・バックアップを取ることがサポートされます。オンライン・バックアップ操作時には、データベースに対するすべてのアクティビティーがログに記録されます。オンライン・バックアップの完了後に、データベース・マネージャーが現行のアクティブ・ログを強制的にクローズすることにより、そのアーカイブが行われます。このプロセスにより、オンライン・バックアップに、リカバリーに使用できるアーカイブ・ログの完全セットが揃うことになります。オンライン・バックアップ・イメージがリストアされるとき、これらのログは少なくともバックアップ操作が完了した時点までロールフォワードされなければなりません。この操作を円滑に行うには、データベースのリストア時にアーカイブ・ログを使用可能にする必要があります。

logarchmeth1 および **logarchmeth2** データベース構成パラメーターを使用して、アーカイブ・ログの保管場所を指定することができます。 **logarchmeth1** パラメーターを使用して、**logpath** 構成パラメーターで設定されているアクティブ・ログ・パスから、ログ・ファイルをアーカイブすることができます。 **logarchmeth2** パラメーターを使用して、ログ・ファイルの追加のコピーを、アクティブ・ログ・パスから別の場所にアーカイブすることができます。ミラー・ロギングを構成していない場合、**logarchmeth1** パラメーターで使用されたログ・パスと同じログ・パスから、追加のコピーが取られます。ミラー・ロギングを **mirrorlogpath** 構成パラメーターによって構成している場合、**logarchmeth2** 構成パラメーターは、代わりにミラー・ログ・パスからログ・ファイルをアーカイブします。これにより、ロールフォワード・リカバリー時の回復力が向上します。 **newlogpath** パラメーターは、アクティブ・ログの保管場所に影響を与えます。

特定のシナリオでは、アーカイブ・ログ・ファイルを圧縮することが、これらのファイルに関連するストレージ・コストの削減につながります。**logarchmeth1** および **logarchmeth2** 構成パラメーターが DISK、TSM、または VENDOR に設定されている場合は、**logarchcompr1** および **logarchcompr2** 構成パラメーターを ON に設定して、アーカイブ・ログ・ファイルの圧縮を使用可能にすることができます。**logarchcompr1** および **logarchcompr2** が動的に設定されている場合、既にアーカイブされているログ・ファイルは圧縮されません。

LOGRETAIN オプションを使用して、アクティブ・ログを管理することを示す値を指定した場合、データベース・マネージャーは、ログ・ファイルをアーカイブしてそれらのファイルがクラッシュ・リカバリーに必要なくなると、アクティブ・ログ・パスのこれらのファイルを名前変更します。無限ロギングを有効にした場合、さらに多くのアクティブ・ログ・ファイルのためにスペースを空ける必要が生じると、データベース・マネージャーは、ログ・ファイルをアーカイブした後にそのログ・ファイルの名前を変更します。

ログ制御ファイル

データベースが障害の後に再始動すると、データベース・マネージャーはログ・ファイルに格納されたトランザクション情報を適用して、データベースを整合性のある状態に戻します。ログ・ファイルからデータベースに適用される必要のあるレコードを判別するために、データベース・マネージャーはログ・コントロール・ファイルに記録された情報を使用します。

データベース回復力のための冗長度

データベース・マネージャーは、各メンバーのログ制御ファイルの 2 つのコピー SQLOGCTL.LFH.1 と SQLOGCTL.LFH.2、およびグローバル・ログ制御ファイルの 2 つのコピー SQLOGCTL.GLFH.1 と SQLOGCTL.GLFH.2 を保守して、一方のコピーが損傷した場合でも、データベース・マネージャーが他方のコピーを使用できるようにします。

パフォーマンスの考慮

ログ制御ファイルに含まれるトランザクション情報を適用すると、障害の後にデータベースを再始動するときのオーバーヘッドが増大します。「データベース: 管理の概念および構成リファレンス」の『softmax - 「リカバリー範囲およびソフト・チェックポイント・インターバル」構成パラメーター』を使用して、データベース・マネージャーがバッファー・プール・ページをディスクに書き込む頻度を構成することにより、クラッシュ・リカバリーの際に処理する必要のあるログ・レコードの数を少なくすることができます。

第 3 章 IBM Data Server における高可用性

IBM Data Server には、多数の高可用性ストラテジーをサポートする機能が組み込まれています。

自動クライアント・リルートのロードマップ

自動クライアント・リルートは、障害が発生したサーバーから代替サーバーにクライアント・アプリケーションをリダイレクトすることにより最小の中断でアプリケーションが作業を継続できるようにする IBM Data Server のフィーチャーの 1 つです。自動クライアント・リルートは、障害が発生する前に代替サーバーが指定されている場合のみ行えます。

表 1 は、各カテゴリーの関連トピックをリストしています。

表 1. 自動クライアント・リルートの情報のロードマップ

カテゴリー	関連トピック
一般情報	• 27 ページの『自動クライアント・リルートの制約事項』 • 23 ページの『自動クライアント・リルートについての説明およびセットアップ』 • 「*DB2 Connect DB2 Connect サーバー機能 インストールおよび構成*」の『自動クライアント・リルートについての説明およびセットアップ (DB2 Connect)』
構成	• 27 ページの『自動クライアント・リルート用の代替サーバーの識別』 • 「Java アプリケーションの開発」の『Java クライアント用の DB2 Database for Linux, UNIX, and Windows 高可用性サポートの構成』
例	• 258 ページの『自動クライアント・リルートの例』
他の DB2 フィーチャーとの相互作用	• 38 ページの『自動クライアント・リルートおよび高可用性災害時リカバリー (HADR) の構成』 • 「Java アプリケーションの開発」の『Java クライアント用の DB2 Database for Linux, UNIX, and Windows 高可用性サポートの構成』
トラブルシューティング	• 25 ページの『クライアント接続ディストリビューター・テクノロジーに対する自動クライアント・リルートの構成』

注: DB2 for z/OS シスプレックスの自動クライアント・リルートは、IBM データ・サーバー・クライアントと非 Java IBM データ・サーバー・ドライバーでも使用可能です。このサポートがあるので、DB2 for z/OS シスプレックスにアクセスするアプリケーションは、クライアントが提供する自動クライアント・リルート機能を使用でき、DB2 Connect サーバーを経由する必要がありません。このフィーチャーについて詳しくは、*DB2 Information Center* の自動クライアント・リルート (クライアント・サイド) のトピックを参照してください。

Linux および UNIX 用の DB2 障害モニター機能

これは UNIX ベースのシステムでのみ使用可能です。 DB2 障害モニター機能を使用すると、DB2 データベース・マネージャー・インスタンスをモニターし、不完全な状態で終了するインスタンスをすべて再始動することによって、DB2 Data Server データベースの稼働状態を簡単に維持することができます。

障害モニター・コーディネーター (FMC) は、 UNIX ブート・シーケンスで開始される障害モニター機能のプロセスです。*init* デーモンは FMC を開始し、それが異常終了した場合には再開します。 FMC は、DB2 インスタンスごとに 1 つの障害モニターを開始します。それぞれの障害モニターは、デーモン・プロセスとして実行され、 DB2 インスタンスと同等のユーザー特権を持っています。

障害モニターが開始すると、それが不完全な状態で終了することがないようにモニターされます。障害モニターに障害が起きた場合には、それは FMC により再開されます。さらに障害モニターは、それぞれ 1 つの DB2 インスタンスのモニターを受け持ちます。DB2 インスタンスか不完全な状態で終了する場合には、障害モニターがそれを再開します。障害モニターは、**db2stop** コマンドが実行された場合のみ非アクティブになります。DB2 インスタンスがその他の仕方でシャットダウンした場合には、障害モニターはそれを再開します。

DB2 障害モニターに関する制約事項

高可用性クラスタリング製品 (IBM Tivoli® System Automation for Multiplatforms、IBM PowerHA® SystemMirror for AIX®、または MSCS など) を使用している場合、インスタンスの始動とシャットダウンはそのクラスタリング製品により制御されるため、障害モニター機能をオフにする必要があります。

DB2 障害モニターと DB2 ヘルス・モニターの違い

ヘルス・モニターと障害モニターは、単一のデータベース・インスタンス上で作動するツールです。ヘルス・モニターは、ヘルス・インディケーター を使用して、データベース・マネージャーやデータベースのパフォーマンスの特定の性質の正常性を評価します。ヘルス・インディケーターは、表スペースなどの、データベース・オブジェクトの特定のクラスの何らかの面の正常性を測定します。特定の基準に対してヘルス・インディケーターを評価して、データベース・オブジェクトのそのクラスの正常性を判別できます。さらにヘルス・インディケーターは、アラートを生成して、インディケーターがしきい値を超過した時点を通知したり、データベース・オブジェクトの状態が通常でないことを示したりすることができます。

比較すると、障害モニターのみが、モニター対象のインスタンスが稼働し続けるようにすることを担当します。障害モニターがモニターしている DB2 インスタンスが不慮に終了した場合は、障害モニターはそのインスタンスを再始動します。障害モニターは、Windows では使用できません。

高可用性災害時リカバリー (HADR)

高可用性災害時リカバリー (HADR) フィーチャーは、部分的なサイト障害と全体的なサイト障害の両方に関する可用性の高い解決方法を提供します。 HADR は、*1 次データベース* と呼ばれる 1 つのソース・データベースから、*スタンバイ・データベース* と呼ばれる 1 つ以上のターゲット・データベースにデータの変更内容を複製して、データの損失を防ぎます。

部分的なサイト障害は、ハードウェア、ネットワーク、またはソフトウェア (DB2 データベース・システムまたはオペレーティング・システム) 障害によって生じることがあります。HADR を使用しない場合、部分サイト障害では、データベースの入ったデータベース管理システム (DBMS) サーバーを再始動する必要があります。データベースおよびそれが置かれているサーバーを再始動するのにかかる時間の長さは、予測できません。データベースが整合した状態に戻って使用可能になるまでに、数分かかることがあります。 HADR を使用すると、スタンバイ・データベースが数秒で処理を引き継ぐことができます。さらに、自動クライアント・リルートを使用するか、アプリケーションで再試行ロジックを使用することで、元の 1 次データベースを使用していたクライアントを新しい 1 次データベースにリダイレクトできます。

全サイト障害は、火災などの災害によってサイト全体が破壊される場合に生じ得ます。ただし、HADR では、1 次データベースとスタンバイ・データベースとの通信に TCP/IP を使用するため、それぞれ別の場所に置くことができます。例えば、1 次データベースをある都市の本社に置き、スタンバイ・データベースを別の都市の営業所に置くことができます。プライマリー・サイトで災害が生じる場合、リモートのスタンバイ・データベースが、完全な DB2 機能を備えた 1 次データベースとしてサイトを引き継ぎ、データの可用性が維持されます。テークオーバー操作が行われた後で、元の 1 次データベースのバックアップを使用して、 1 次データベースの状態に戻すことができます。このことを、*フェイルバック*といいます。古い 1 次データベースと新しい 1 次データベースを一致させることが可能であれば、フェイルバックを開始できます。古い 1 次データベースをスタンバイ・データベースとして HADR セットアップに再統合したら、データベースの役割を切り替えて、元の 1 次データベースをもう一度 1 次データベースにすることができます。

HADR を使用して、構成およびトポロジーの選択項目に基づいて、潜在的なデータの損失を防ぐレベルを指定します。選択する必要がある主な項目をいくつか以下に示します。

使用する同期レベル

1 次データベースで生成され、スタンバイ・データベースに送られるログ・データによって、スタンバイ・データベースが 1 次データベースと同期されます。スタンバイは常にログを使用してロールフォワードを行います。4 つの異なる同期モードから選択できます。保護レベルは、最高が SYNC であり、以後 NEARSYNC、ASYNC、SUPERASYNC の順に低くなります。詳しくは、 64 ページの『高可用性災害時リカバリー (HADR) 同期モード』を参照してください。

ピア・ウィンドウの使用

ピア・ウィンドウ・フィーチャーは、1 次データベースがピア状態で HADR から切断されても、構成された時間の間は 1 次データベースとスタ

ンバイ・データベースを引き続きピア状態にあるかのように動作するよう指定します。ピア状態またはこの「切断済みピア」状態で 1 次に障害が発生した場合、スタンバイにフェイルオーバーするため、データが損失することはありません。このフィーチャーにより最高度の保護が提供されます。詳しくは、 49 ページの『hadr_timeout および hadr_peer_window データベース構成パラメーターの設定』を参照してください。

デプロイするスタンバイの数

HADR では、単一スタンバイ・モードまたは複数スタンバイ・モードを使用できます。複数スタンバイを使用した場合は、1 つのテクノロジーで高可用性と災害時リカバリーの両方の目標を達成することができます。詳しくは、 219 ページの『HADR 複数スタンバイ・データベース』を参照してください。

HADR スタンバイは、HA や DR としての役割以外にも、以下のような多数の用途があります。

スタンバイ・データベースの読み取り

スタンバイ・データベースの読み取りフィーチャーを使用すれば、スタンバイ・データベースの HA や DR の動作に影響することなく、読み取り専用ワークロードを 1 つ以上のスタンバイに送信できます。このフィーチャーは、スタンバイの主な役割に影響を与えることなく、1 次のワークロードを削減するのに役立ちます。このトピックについて詳しくは、 244 ページの『HADR スタンバイ・データベースの読み取りフィーチャー』を参照してください。

スタンバイ・データベースの読み取りを使用可能にしない限り、アプリケーションは、現行 1 次データベースにしかアクセスできません。スタンバイ・データベースの読み取りを使用可能にした場合は、読み取り専用アプリケーションをスタンバイにリダイレクトできます。スタンバイ・データベースに接続しているアプリケーションが、フェイルオーバー時のスタンバイの可用性に影響を与えることはありません。

遅延再生

遅延再生を使用して、スタンバイ・データベースのログの再生時点が、必ず 1 次データベースの再生時点よりも前になるように指定できます。1 次でデータが失われた場合や破損した場合、遅延時間を指定したスタンバイでこのデータをリカバリーできます。詳しくは、 212 ページの『HADR 遅延再生』を参照してください。

ローリング更新とローリング・アップグレード

HADR セットアップを使用すると、データベースに対して、さまざまなタイプのアップグレードおよび DB2 フィックスパック更新を障害を発生することなく実行できます。複数スタンバイ・モードを有効にして使用している場合は、HADR によって保護された状態のままでアップグレードを実行できます。詳しくは、 191 ページの『DB2 高可用性災害時リカバリー (HADR) 環境でのローリング更新とローリング・アップグレードの実行』を参照してください。

データベース内のほとんどのデータまたはすべてのデータで保護が必要な場合や、スタンバイ・データベースで自動的に複製する必要がある DDL 操作を実行する場合は、HADR が最善のオプションかもしれません。ただし、HADR は、DB2 製品

ファミリーで提供されている、いくつかのレプリケーション・ソリューションの 1 つに過ぎません。InfoSphere® Federation Server ソフトウェアおよび DB2 データベース・システムには、SQL レプリケーション・ソリューションおよび Q レプリケーション・ソリューションが含まれています。これらは、特定の構成で高可用性を実現するために使用することもできます。これらのソリューションは、論理的に整合したデータベース表のコピーを、複数の場所で維持するものです。さらに、列および行のフィルター操作、データ形式変更、表のコピーの更新のサポートなど、柔軟かつ複雑な機能を提供します。また、これらのソリューションはパーティション・データベース環境で使用することもできます。

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます: HADR のセットアップ. タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、およびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

DB2 高可用性 (HA) フィーチャー

DB2 高可用性 (HA) フィーチャーによって、IBM Data Server とクラスター管理ソフトウェアの統合が可能になります。

クラスター環境でデータベース・マネージャー・インスタンスを停止する場合、当該インスタンスの停止をクラスター・マネージャーに知らせる必要があります。クラスター・マネージャーは、当該インスタンスの停止を知らない場合、停止したインスタンスにフェイルオーバーなどの操作を試みる可能性があります。 DB2 高可用性 (HA) フィーチャーは、データベース・マネージャー・インスタンスの停止などのインスタンス構成の変更によってクラスター変更が必要な場合に、データベース・マネージャーがクラスター・マネージャーと通信できるようにするためのインフラストラクチャーを提供します。

インスタンスの変更によってクラスターの変更が必要な場合は常にデータベース・マネージャーがクラスター・マネージャーと通信すれば、インスタンス構成の変更を実行した後に、別個のクラスター操作を実行する必要がなくなります。

DB2 HA フィーチャー は、以下のエレメントで構成されています。

- IBM Tivoli System Automation for Multiplatforms (SA MP) は、AIX および Linux 上の IBM Data Server に DB2 高可用性 (HA) フィーチャーの一部としてバンドルされており、DB2 インストーラーに統合されています。DB2 インストーラーまたは IBM Data Server のインストール・メディアに組み込まれている **installSAM** および **uninstallSAM** スクリプトを使用して、SA MP をインストール、アップグレード、またはアンインストールすることができます。
- クラスター化された環境では、一部のデータベース・マネージャーのインスタンス構成および管理操作で、関連するクラスター構成の変更が必要となります。DB2 High Availability Feature (HA) フィーチャーは、特定のデータベース・マネージャーのインスタンス構成および管理操作を実行する度に、データベース・マネージャーがクラスター・マネージャーの構成変更を自動的に要求できるようにします。 99 ページの『DB2 高可用性 (HA) フィーチャーを使用したクラスターの自動構成』 を参照。

- DB2 高可用性インスタンス構成ユーティリティー (db2haicu) は、クラスター環境の高可用性データベースを構成および管理するために使用できるテキスト・ベースのユーティリティーです。 **db2haicu** は、データベース・インスタンス、クラスター環境、およびクラスター・マネージャーに関する情報を、システムを照会して収集します。ユーザーは、**db2haicu** 呼び出しへのパラメーターおよび入力ファイルを介して、または実行時に **db2haicu** プロンプトで、詳細情報を提供します。 110 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu)』 を参照。
- DB2 クラスター・マネージャー API は、データベース・マネージャーが構成変更をクラスター・マネージャーに通信するために必要な関数のセットを定義します。 150 ページの『DB2 クラスター・マネージャー API』 を参照。

ログ・シッピングによる高可用性

ログ・シッピングとは、すべてのログ・ファイルをスタンバイ・マシンにコピーする処理のことです。これは、1 次データベースにおいて、アーカイブ・ログを保持しているストレージ装置から直接コピー、または、アーカイブ・ログをコピーするユーザー出口プログラムを利用して行います。

スタンバイ・データベースは、稼働マシンにより作成されたログ・ファイルを連続的にロールフォワードしています。稼働マシンに障害が起こった時、フェイルオーバーが発生し、以下のことがなされます。

- 残ったログがスタンバイ・マシンに転送される。
- スタンバイ・データベースにおいて to end of logs and stop のオプションを指定した ROLLFORWARD が実行される。
- クライアントがスタンバイ・データベースに再接続し、操作を再開する。

スタンバイ・マシンには、それ自体のリソース (ディスクなど) がありますが、稼働データベースと同じ物理的および論理的定義を持っている必要があります。この方法を使用する場合、1 次データベースは、リストア・ユーティリティーまたはスプリット・ミラー機能を使用してスタンバイ・マシンにリストアされます。

災害時リカバリーの状況でデータベースを確実にリカバリーできるようにするために、以下の点を考慮してください。

- アーカイブの場所は、プライマリー・サイトとは地理的に分離しているべきである。
- 1 次データベースのログをスタンバイ・データベース・サイトへ遠隔ミラーする。
- データ損失ゼロとするためには、同期ミラーを使用する。このことは、最新のディスク・サブシステム (ESS や EMC など) や、別のリモート・ミラーリング・テクノロジーを使用して行えます。 NVRAM キャッシュ (ローカルとリモートの両方) も、災害時リカバリーのパフォーマンスへの影響を最小化するためにお勧めします。

スタンバイ・マシン上でどのログ・ファイルをロールフォワードするか制御する場合には、**ROLLFORWARD DATABASE** コマンドで **NORETRIEVE** オプションを使用することにより、アーカイブ・ログの取得を無効にすることができます。これには、以下のような場合に役立ちます。

- ロールフォワードするログ・ファイルを制御することで、スタンバイ・マシンを実動マシンより *X* 時間遅れになるようにして、両方のシステムに影響を与えないようにすることができます。
- スタンバイ・システムにアーカイブへのアクセス権限がない場合 (例えば、TSM がアーカイブである場合、ファイルをリトリーブできるのは元のマシンのみです)。
- 実動システムがファイルをアーカイブしているときに、スタンバイ・システムが同じファイルをリトリーブして不完全なログ・ファイルを取得する可能性もあります。この問題は、**NORETRIEVE** を使用して解決できます。

注:

1. 索引の再作成が 1 次データベースで行われたことを示すログ・レコードをスタンバイ・データベースが処理する時、スタンバイ・サーバー上の索引は自動的に再作成されません。データベースへの最初の接続時、またはスタンバイ・サーバーがロールフォワード・ペンディング状態から解放された後に初めて索引にアクセスしようとした時に、索引はスタンバイ・サーバーに再作成されます。 1 次サーバーのいずれかの索引が再作成された場合には、スタンバイ・サーバーが 1 次サーバーと再同期を取ることをお勧めします。**logindexbuild** データベース構成パラメーターを設定すると、ロールフォワード操作中に索引を再作成できます。
2. ロード・ユーティリティーが、**COPY YES** オプションを指定した状態で 1 次データベース上で実行される場合、スタンバイ・データベースは、ロールフォワード時に、そのコピー・イメージへアクセスできる必要があります。
3. ロード・ユーティリティーが、**COPY NO** オプションを指定した状態で 1 次データベース上で実行される場合、スタンバイ・データベースは再同期を取る必要があります。再同期しない場合、表スペースはリストア・ペンディング状態に置かれます。
4. スタンバイ・マシンを初期化するには、2 つの方法があります。
   a. バックアップ・イメージからそこにリストアすることにより。
   b. 実動システムのスプリット・ミラーを作成し、 **STANDBY** オプションで **db2inidb** コマンドを実行することにより。

   スタンバイ・マシンを初期化した後にのみ **ROLLFORWARD** コマンドをスタンバイ・システム上で実行することができます。
5. ログに記録されていない操作は、スタンバイ・データベースでは再実行できません。結果として、そうした操作をした後には、スタンバイ・データベースを再同期することをお勧めします。このことは、オンライン・スプリット・ミラーおよびサスペンド入出力サポートを通して行えます。

ログのミラーリング

IBM Data Server は、データベース・レベルでのログ・ミラーリングをサポートします。ログ・ファイルをミラーリングすると、アクティブ・ログの不慮の削除、およびハードウェア障害によるデータ破損といった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・ログが (ディスク・クラッシュの結果) 損傷する可能性があることを懸念している場合は、**mirrorlogpath** 構成パラメーターを使用し、データベースがアクティブ・ログのコピーを管理するための 2 次パスを指定することにより、ログの保管先のボリュームをミラーリングすることを検討してください。

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

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

アクティブ・ログ・パスまたはミラー・ログ・パスへのエラーの書き込みが生じると、データベースにより、障害が起きたパスに「不良」のマークが付けられ、管理通知ログにメッセージが書き込まれ、以後ログ・レコードを残りの「良好な」ログ・パスだけに書き込まれます。現行のログ・ファイルが満杯になるか切り捨てられるまで、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 により維持されるので、ログ・ファイルがアーカイブされる際には正しいパスが使用されます。残りの「良好な」パスへの書き込み中に障害が起きると、データベースはシャットダウンします。

サスペンド入出力とオンライン・スプリット・ミラー・サポートによる高可用性

IBM Data Server のサスペンド入出力サポートにより、データベースをオフラインにしないで 1 次データベースのスプリット・ミラー・コピーを行うことができます。これを使用すると、1 次データベースで障害が発生した場合にテークオーバーするスタンバイ・データベースを非常に短い時間で作成することができます。

ディスク・ミラーリングは、データを 2 つの異なるハード・ディスクに同時に書き込む処理です。データの一方のコピーは、他方のミラーと呼ばれます。ミラーの分割とは、2 つのコピーを分離する処理のことです。

ディスク・ミラーリングを使用することにより、1 次データベースの 2 次コピーを保持することができます。 IBM Data Server のサスペンド入出力機能を使用すると、データベースをオフラインにしないでデータベースの 1 次ミラー・コピーと 2 次ミラー・コピーを分割することができます。 1 次データベース・コピーと 2 次データベース・コピーが分割されると、2 次データベースは 1 次データベースで障害が起きた場合に操作をテークオーバーできるようになります。

大きなデータベースは IBM Data Server バックアップ・ユーティリティーを使用してバックアップしないという場合、サスペンド入出力およびスプリット・ミラー機能を使用してミラー・イメージからコピーを作成することができます。この方法では以下の利点もあります。

- 稼働マシンからバックアップ操作のオーバーヘッドを除去します。
- 高速にシステムを複製します。
- アイドル・スタンバイ・フェイルオーバーを高速にインプリメントできます。初期のリストア操作は不要です。また、ロールフォワードが遅すぎたりエラーが発生する場合に、再初期化を高速に実行してこれらに対応することが可能です。

db2inidb コマンドは、スプリット・ミラーを初期化して以下のように使用できるようにします。

- クローン・データベースとして
- スタンバイ・データベースとして
- バックアップ・イメージとして

このコマンドは、スプリット・ミラーに対してのみ発行することができます。そのコマンドは、スプリット・ミラーを使用する前に実行しなければなりません。

パーティション・データベース環境では、入出力を中断してすべてのデータベース・パーティションに同時に書き込む必要はありません。 1 つ以上のデータベース・パーティションのサブセットを中断して、オフライン・バックアップを実行するためにスプリット・ミラーを作成できます。カタログ・パーティションがサブセットに含まれる場合は、そのパーティションを最後に中断するデータベース・パーティションにする必要があります。

パーティション・データベース環境では、**db2inidb** コマンドは各データベース・パーティション上でそれぞれ実行する必要があります。それから、それらのデータベース・パーティションのスプリット・イメージを使用することができます。db2inidb コマンドは、**db2_all** コマンドを使用してすべてのデータベース・パーティションで同時に実行することができます。しかし、RELOCATE USING オプションを使用する場合は、**db2_all** コマンドを使用して全データベース・パーティションに対して同時に **db2inidb** を実行することはできません。データベース・パーティションごとにそれぞれ別個の構成ファイル (変更するデータベース・パーティションの NODENUM 値が含まれる) を用意する必要があります。例えば、データベースの名前を変更する場合は、すべてのデータベース・パーティションが影響を受けることになり、各データベース・パーティションごとに別個の構成ファイルを用意して **db2relocatedb** コマンドを実行する必要があります。単一データベース・パーティションに属するコンテナーを移動する場合は、そのデータベース・パーティションに対して一度だけ **db2relocatedb** コマンドを実行することが必要です。

注: スプリット・ミラーが、データベースを構成するすべてのコンテナーおよびディレクトリー (ボリューム・ディレクトリーを含む) を含んでいることを確認してください。この情報を収集するには、DBPATHS 管理ビューを参照してください。このビューは、分割する必要のあるデータベースのすべてのファイルとディレクトリーを表示します。

第 4 章 高可用性のための構成

DB2 データベース・ソリューションを高可用性のために構成するには、データベース保守活動をスケジュールすること、1 次およびスタンバイ・データベース・サーバーが相互に認識して障害発生時のそれぞれの役割を知るように構成すること、およびクラスター管理用のソフトウェアがワークロードを障害が生じたクラスター・ノードから転送するように構成することが必要です。

始める前に

データベース・ソリューションを構成する前に行うこと。

- ソリューションを構成する基礎となるハードウェアおよびソフトウェア・コンポーネントを集めてインストールします。これらの基礎となるコンポーネントには、電源機構、ネットワーク接続、ネットワーク・カード、ディスクその他のストレージ・デバイス、オペレーティング・システム、クラスター管理用のソフトウェアなどが含まれることがあります。
- これらの基礎となるコンポーネントをデータベース・ロード・バランシング、フェイルオーバー、またはリカバリー操作で使用することを試行する前に、データベース・ワークロードなしでテストして、それらが正常に機能していることを確認します。

このタスクについて

冗長度は、高可用性ソリューションで重要な部分となります。ただし、保守を賢明にスケジュールしない場合、必要なリカバリー・ログ用のストレージ・スペースが不足する場合、またはクラスター管理用のソフトウェアが正しく構成されていない場合は、ユーザーがデータベースを使用して重要な作業を行うときにソリューションが使用可能ではないことがあります。

手順

高可用性のための構成には、次のものが含まれます。

- クライアント・リルートの構成
- 障害モニターの構成
- DB2 高可用性災害時リカバリーの構成
- 保守活動のスケジュール
- ロギングの構成
- クラスター管理用のソフトウェアの構成

自動クライアント・リルートについての説明およびセットアップ

自動クライアント・リルート・フィーチャーの主な目的は、IBM Data Server Client アプリケーションが通信の消失からリカバリーして最小限の中断で動作を続行できるようにすることです。名前が示すように、リルートは連続稼働サポートの要です。ただし、リルートが可能なのは、クライアント接続のための代替ロケーションが示されている場合のみです。

サーバーが DB2 Database for Linux, UNIX, and Windows である場合、自動クライアント・リルート・フィーチャーは、例えば以下のような構成可能な環境で使用できます。

1. DB2 Database Partitioning Feature を使用する DB2 Enterprise Server Edition
2. IBM DB2 pureScale® Feature を使用する DB2 Enterprise Server Edition
3. WebSphere® Replication Server
4. IBM PowerHA SystemMirror for AIX
5. 高可用性災害時リカバリー (HADR)

   自動クライアント・リルートが HADR または DB2 pureScale Feature と連動することにより、クライアント・アプリケーションは、機能の中断を最小限に抑えて、アクセス先データベースのフェイルオーバー後に稼働を続けることができます。

データベース・サーバーが System i® または System z® 上にある場合、シームレス自動クライアント・リルート・フィーチャーは、以下の構成で使用されます。

1. IBM Data Server Client が、代替サーバーを持つ DB2 Connect サーバーを通して z/OS または i5/OS® システムに接続する。IBM Data Server Client と 2 つの DB2 Connect サーバーの間で自動クライアント・リルートが使用されます。
2. DB2 for z/OS シスプレックス・データ共有環境にアクセスする DB2 Connect クライアントまたはサーバー製品。DB2 Connect と z/OS シスプレックス・システムの間で自動クライアント・リルートが使用されます。自動クライアント・リルート・フィーチャーは、DB2 Connect ライセンス・クライアントとシスプレックスの間のシームレスなフェイルオーバーをサポートします。シームレス・フェイルオーバーについて詳しくは、*DB2* インフォメーション・センターで自動クライアント・リルート (クライアント・サイド) に関するトピックを参照してください。

DB2 Connect サーバーとその代替サーバーの場合、ローカル・データベースの同期に関する要件はないので、オリジナルと代替の両方の DB2 Connect サーバーで、同一のデータベースの別名を使用してアクセスできるように、ターゲット・ホストまたは System i データベースをカタログするだけで済みます。

DB2 データベース・システムが通信障害からリカバリーできるようにするためには、通信障害が発生する前に、代替サーバーの場所を指定しておく必要があります。 **UPDATE ALTERNATE SERVER FOR DATABASE** コマンドを使用して、特定のデータベース上に代替サーバー・ロケーションを定義します。

サーバー・インスタンスの特定のデータベース上にある代替サーバー・ロケーションを指定した後、接続プロセスの一環として、代替サーバー・ロケーション情報が IBM Data Server Client に戻されます。DB2 Connect クライアントまたはサーバー製品とホストまたは System i データベース・サーバーの間で自動クライアント・リルートを使用する場合、リモート・サーバーは自身の代替アドレスを 1 つ以上提供する必要があります。DB2 for z/OS の場合、データベースがシスプレックス・データ共有環境のときは、複数のアドレスが分かっています。このため、DB2 Connect に代替サーバーをカタログする必要はありません。何らかの理由でクライアントとサーバーとの間の通信が失われた場合、代替サーバーの情報を持っている IBM Data Server Client は代替サーバー情報を使用して接続の再確立を試行します。IBM Data

Server Client が再接続を試みるデータベース・サーバーは、元のサーバーと、元のサーバーのデータベース・ディレクトリー・ファイルにリストされた代替サーバーか、z/OS シスプレックス・システムが返すサーバー・リストに含まれる代替サーバーのいずれかになります。接続を再確立しようとするこうした試行は、最初は非常に短い時間間隔で行われ、徐々に試行の間隔が長くなります。

接続が成功すると、通信障害の後にデータベース接続が再確立されたことを示す SQLCODE -30108 が戻されます。ホスト名または IP アドレスと、サービス名またはポート番号が戻されます。クライアント通信の再確立が元のサーバーに対しても代替サーバーに対しても不可能な場合、IBM Data Server Client は元の通信障害に関するエラーだけをアプリケーションに戻します。

DB2 Connect サーバー環境での代替サーバー接続に関する以下の考慮事項にも注意する必要があります。

- リモート・クライアントとローカル・クライアントの両方のために DB2 Connect サーバーを使用してホストまたは System i データベースへのアクセスを提供する場合、システム・データベースのディレクトリー項目にある代替サーバーの接続情報に関して混乱が生じる可能性があります。この混乱を最小限に抑えるために、同じホストまたは System i データベースを表すように、システム・データベースのディレクトリーで 2 つの項目をカタログすることを検討してください。リモート・クライアント用に 1 つの項目、ローカル・クライアント用にもう 1 つの項目をカタログします。
- ターゲットの DB2 for z/OS サーバーから返されるシスプレックス情報は、DB2 Connect サーバーのキャッシュでのみ維持されます。ディスクに書き込まれる代替サーバーは 1 つのみです。複数の代替サーバーまたは複数のアクティブ・サーバーが存在する場合、その情報はメモリーでのみ維持され、プロセスが終了すると失われます。

代替サーバーが指定されている場合、一般的に、通信エラー (sqlcode -30081) またはデータベース・エージェントの終了 (sqlcode -1224) が検出されたときに自動クライアント・リルートが使用可能になります。ただし、高可用性災害時リカバリー (HADR) 環境では、 sqlcode -1776 が HADR スタンバイ・サーバーから戻された場合にもまた、これが使用可能になります。

ワークロード・バランシングおよび自動クライアント・リルートのためには、/etc/hosts ファイルにあるクラスターのメンバーごとにクライアントの項目が必要です。以下に例を示します。

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

クライアント接続ディストリビューター・テクノロジーに対する自動クライアント・リルートの構成

1 次データベース・サーバーで障害が発生した場合、ディストリビューターまたはディスパッチャー・テクノロジー (WebSphere Edge Server Load Balancer など) により、システムの定義済みセットに対するクライアント・アプリケーション再接続要求が分散されます。ディストリビューター・テクノロジーと DB2 自動クライアント・リルートを併用している場合、ディストリビューター自体を DB2 自動クライアント・リルート先の代替サーバーと見なす必要があります。

ディストリビューター・テクノロジーを次のような環境で使用していることがあります。

クライアント -> ディストリビューター・テクノロジー -> (DB2 Connect サーバー 1 または DB2 Connect サーバー 2) -> DB2 for z/OS

ここで、

- ディストリビューター・テクノロジー・コンポーネントの TCP/IP ホスト名は DThostname
- DB2 Connect サーバー 1 の TCP/IP ホスト名は GWYhostname1
- DB2 Connect サーバー 2 の TCP/IP ホスト名は GWYhostname2
- DB2 for z/OS サーバーの TCP/IP ホスト名は zOShostname

ディストリビューター・テクノロジーを使用していずれかの DB2 Connect サーバーにアクセスするために、クライアントは **DThostname** を使用してカタログされます。 ディストリビューター・テクノロジーの介入により、**GWYhostname1** または **GWYhostname2** を使用することが決定します。決定後、クライアントはこれら 2 つの DB2 Connect ゲートウェイのうちのいずれかに直接ソケット接続します。選択された DB2 Connect サーバーへのソケット接続が確立されると、標準クライアント --> DB2 Connect サーバー --> DB2 for z/OS の接続が成立します。

例えば、ディストリビューターが **GWYhostname2** を選択すると想定します。これにより、次の環境が生成されます。

クライアント —> DB2 Connect サーバー 2 —> DB2 for z/OS

ディストリビューターは、何らかの通信障害があると、接続を再試行しません。このような環境においてデータベースでの自動クライアント・リルート・フィーチャーを使用可能にする場合、DB2 Connect サーバー (DB2 Connect サーバー 1 または DB2 Connect サーバー 2) 内の 1 つ以上の関連データベースの代替サーバーを、ディストリビューター (DThostname) としてセットアップする必要があります。 DB2 Connect サーバー 1 が何らかの理由でロックされる場合、自動クライアント・リルートがトリガーされ、ディストリビューターを 1 次サーバーおよび代替サーバーの両方として使用してクライアント接続が再試行されます。このオプションを使用すると、ディストリビューター・フィーチャーと DB2 自動クライアント・リルート・フィーチャーを結合して保守できます。代替サーバーをディストリビューターのホスト名以外のホストに設定することによっても、クライアントに自動クライアント・リルート・フィーチャーが提供されます。ただし、クライアントは定義済み代替サーバーへの直接接続を確立して、ディストリビューター・テクノロジーをバイパスします。これによりディストリビューターとその価値が無効になります。

自動クライアント・リルート・フィーチャーは次の SQL コードをインターセプトします。

- SQL20157N
- SQL1768N (理由コード: 7)

注: 「TCP キープアライブ」オペレーティング・システム構成パラメーターの設定値が高すぎる場合、クライアント・リルートはソケット障害に関して即時に知らされないことがあります。 (この構成パラメーターの名前はプラットフォームによって異なることに注意してください。)

自動クライアント・リルート用の代替サーバーの識別

DB2 サーバーまたは DB2 Connect サーバーでクラッシュが発生すると、そのサーバーに接続している各クライアントは通信エラーを受け取り、接続が終了してアプリケーション・エラーが発生します。可用性が重視される場合は、重複セットアップ、またはサーバーをスタンバイ・ノードにフェイルオーバーする機能を実装します。どちらの場合も、DB2 クライアント・コードは、フェイルオーバー・ノードで稼働している可能性のある元のサーバーとの接続を再確立するか (IP アドレスもまたフェイルオーバーします)、新しいサーバーとの接続を再確立しようとします。

手順

新しいサーバーまたは代替サーバーを定義するには、**UPDATE ALTERNATE SERVER FOR DATABASE** コマンドまたは **UPDATE ALTERNATE SERVER FOR LDAP DATABASE** コマンドを使用します。
これらのコマンドは、システム・データベース・ディレクトリー内で、データベース別名の代替サーバー情報を更新します。

自動クライアント・リルートの制約事項

高可用性 DB2 データベース・ソリューションを設計する際は、DB2 データベース・クライアント・リルートの制約事項を考慮します。

以下に、DB2 データベース自動クライアント・リルート・フィーチャーの制限をリストします。

- 自動クライアント・リルートは、DB2 データベース・サーバーまたは DB2 Connect Serverへの接続に使用される通信プロトコルが TCP/IP の場合にのみ、サポートされます。つまり、TCP/IP 以外のプロトコルを使って接続している場合、自動クライアント・リルート・フィーチャーは使用できません。 DB2 データベースでループバックがセットアップされている場合でも、自動クライアント・リルート・フィーチャーを利用するためには、TCP/IP 通信プロトコルを使用する必要があります。
- DB2 Connect クライアントまたはサーバー製品とホストまたは System i データベース・サーバーの間で自動リルートを使用するとき、以下の場合は関連する考慮点があります。
  - リモート・クライアントとローカル・クライアントの両方のために DB2 Connect Serverを使用してホストまたは System i データベースへのアクセスを提供する場合、システム・データベースのディレクトリー項目にある代替サーバーの接続情報に関して混乱が生じる可能性があります。この混乱を最小限に抑えるために、同じホストまたは System i データベースを表すように、システム・データベースのディレクトリーで 2 つの項目をカタログすることを検討してください。リモート・クライアント用に 1 つの項目、ローカル・クライアント用にもう 1 つの項目をカタログします。

- ターゲットの DB2 for z/OS サーバーから返されるシスプレックス情報は、DB2 Connect Serverのキャッシュでのみ維持されます。ディスクに書き込まれる代替サーバーは 1 つのみです。複数の代替サーバーまたは複数のアクティブ・サーバーが存在する場合、その情報はメモリーでのみ維持され、プロセスが終了すると失われます。

- 代替サーバーのロケーションへの通信が再確立された場合、同じデータベース別名への新しい接続はすべて、代替サーバーのロケーションに接続されます。元のロケーションの問題が修正され、新しい接続を元のロケーションとの間で確立する場合には、以下のいくつかのオプションから選択できます。
  - 代替サーバー・ロケーションをオフラインにして、接続が元のサーバーに再びフェイルオーバーするようにします。 (この場合、元のサーバーを代替サーバーのロケーションとして設定するために、 **UPDATE ALTERNATE SERVER** コマンドを使って元のサーバーがすでにカタログされていることを想定します。)
  - 新しい接続によって使用される新しいデータベース別名をカタログすることができます。
  - データベース項目をアンカタログして、再びカタログすることができます。
- クライアントとサーバーの両方がこのフィーチャーをサポートする場合、 DB2 Database for Linux, UNIX, and Windowsはそのクライアントとサーバーの両方で自動クライアント・リルート・フィーチャーをサポートします。その他の DB2 データベース製品ファミリーは、現時点では、このフィーチャーをサポートしません。
- 自動クライアント・リルート・フィーチャーの動作と DB2 for z/OS シスプレックス環境における自動クライアント・リルートの動作は若干異なります。異なるのは、主に次の点です。
  - 自動クライアント・リルート・フィーチャーでは、1 次サーバーは 1 つの代替サーバーを指定する必要があります。これは、1 次サーバーで発行される **UPDATE ALTERNATE SERVER FOR DATABASE** または **UPDATE ALTERNATE SERVER FOR LDAP DATABASE** コマンドを使用して行われます。このコマンドは、ローカル・データベース・ディレクトリーを代替サーバーの情報で更新し、同一のクライアントにある他のアプリケーションがこの情報にアクセスできるようにします。これに対して DB2 for z/OS で使用されるデータ共有シスプレックスは、クライアントが接続できる 1 つ以上のサーバーのリストをメモリーに保持します。通信障害が発生した場合、クライアントはそのサーバーのリストを使用して、適切な代替サーバーのロケーションを判別します。
  - 自動クライアント・リルート・フィーチャーでは、特殊レジスターの設定が変更されるたびに、サーバーが特殊レジスターの最新の設定をクライアントに通知します。これによりクライアントは、転送が行われた後ランタイム環境を可能な限り再確立できることになります。これに対して DB2 for z/OS で使用されるシスプレックスはコミット境界でクライアントに特殊レジスターの設定を返すので、転送された作業単位内で変更された特殊レジスターは再生する必要があります。その他はすべて自動的に再生されます。

  自動クライアント・リルートのフルサポートは、Linux、UNIX、または Windows クライアントと Linux、UNIX、または Windows サーバーの間でのみ使用可能です。それは、Linux、UNIX、または Windows クライアントと DB2 for z/OS シスプレックス・サーバー (サポートされている任意のバージョン) の間では使用できません。この場合は転送機能のみがサポートされます。

- 代替ホスト・サーバーにインストールされた DB2 データベース・サーバーと元のホスト・サーバーにインストールされた DB2 データベース・インスタンスは同じバージョンでなければなりません (ただし DB2 データベース・サーバーのフィックスパックは DB2 データベース・インスタンスのものより高くても問題ありません)。
- クライアント・マシンでデータベース・ディレクトリーを更新する権限があるかどうかにかかわらず、代替サーバー・ロケーション情報は常にメモリーに保持されます。言い換えると、データベース・ディレクトリーの更新権限がない場合 (または、読み取り専用のデータベース・ディレクトリーである場合)、他のサーバーとの間でメモリーが共有されないため、他のアプリケーションは代替サーバー・ロケーションを判別して使用することができません。
- すべての代替ロケーションの間で、同じ認証が適用されます。つまり、代替ロケーションの認証タイプが元のロケーションと異なる場合、クライアントはデータベース接続を再確立できません。
- 通信障害が発生した場合、グローバル一時表、ID、シーケンス、カーソル、フェデレート処理用のサーバー・オプション (SET SERVER OPTION)、および特殊レジスターなど、すべてのセッション・リソースが失われます。処理を続行するためには、アプリケーションがセッション・リソースを再確立しなければなりません。接続が再確立された後、特殊レジスター・ステートメントを実行する必要はありません。通信エラーの前に発行された特殊レジスター・ステートメントを DB2 データベースが再生するからです。ただし、一部の特殊レジスターは再生されません。以下のものが該当します。
  - SET ENCRYPTPW
  - SET EVENT MONITOR STATE
  - SET SESSION AUTHORIZATION
  - SET TRANSFORM GROUP

  DB2 Connect に問題がある場合は、データ・サーバー上の DB2 Connect 製品に固有の限定された特殊レジスターのリストを参照する必要があります。
- 通信障害の発生後に接続が再確立され、クライアントが CLI、JCC Type 2 または Type 4 ドライバーを使用している場合は、元のサーバーに対して準備された SQL および XQuery ステートメントは新しいサーバーで暗黙的に再び準備されます。ただし、組み込み SQL ルーチン (例えば SQC または SQX アプリケーション) は、新しいサーバーでは再び準備されません。
- クライアント・リルート可能なデータベース別名に対して、高可用性災害時リカバリー (HADR) コマンド (**START HADR**、**STOP HADR**、または **TAKEOVER HADR**) を実行しないでください。HADR コマンドは、データベース別名を介してターゲット・データベースを識別するようにインプリメントされています。その結果、ターゲット・データベースで代替データベースが定義されていると、HADR コマンドは実際に操作しているデータベースを判別することが困難になります。クライアントはクライアント・リルート可能な別名を使用して接続しなければならない場合がありますが、HADR コマンドは特定のデータベースに対して適用される必要があります。これに対応するため、1 次およびスタンバイ・データベースに固有の別名を定義し、HADR コマンドをこれらの別名に対してのみ実行することができます。

- 各データベース・サーバーに定義できる代替サーバーは 1 つだけであるため、HADR 複数スタンバイ・セットアップがある場合は、1 次の代替サーバーとして 1 つのスタンバイ・データベース (おそらく、プリンシパル・スタンバイ・データベース) を選択する必要があります。

自動クライアント・リルートをインプリメントする別の方法は、DNS エントリーを使用して、 DNS エントリー用の代替 IP アドレスを指定することです。この基本にある考え方は、2 番目の IP アドレス (代替サーバーのロケーション) を DNS エントリーに指定することです。クライアントは代替サーバーを認識しませんが、接続時に DB2 データベース・システムが DNS エントリー用の複数の IP アドレス間を切り替えます。

TCP/IP キープアライブ・パラメーターの構成

クライアントとサーバー間の DB2 接続は、TCP/IP プロトコルを使用して通信を行います。TCP/IP 層におけるタイムアウトが原因で生じる可能性のあるフェイルオーバーの問題を回避するには、クライアント側で TCP/IP キープアライブ・パラメーターを調整する必要があります。クライアント側でキープアライブ値を小さくすると、サーバー障害をいっそう適時に検出できるようになります。

クライアントの TCP/IP キープアライブ・パラメーターを更新するには、幾つかの方法があります。どの方法を選択するかは、クライアント接続が IBM Data Server Driver for JDBC and SQLJ に基づいているかどうかによって異なります。

高可用性クライアントの TCP/IP キープアライブ・パラメーターの構成 (JDBC)

このタスクについて

IBM Data Server Driver for JDBC and SQLJ を使用するクライアント・システムの場合、TCP/IP キープアライブ設定は、以下のパラメーターを調整してオペレーティング・システム・レベルで設定します。こうしたコマンドでは推奨値が提供されますが、ご使用のネットワークやサーバーの能力に応じてそれらの設定を微調整してください。

注: オペレーティング・システム・レベルでこれらの設定を変更すると、クライアントのすべての TCP/IP 通信に影響が及びます。

手順

1. **AIX の更新**

   AIX クライアントの場合、以下の 3 つのオペレーティング・システム・キープアライブ・パラメーターを変更します。

   - **tcp_keepidle** - アイドル状態の TCP 接続をアクティブに保つ期間
   - **tcp_keepintvl** - TCP 接続の妥当性検査を行うためにパケットを送信する間隔
   - **tcp_keepcnt** - 送信したキープアライブ・プローブ数がこの数を超えると接続が終了します

AIX オペレーティング・システムでは、「ネットワーク・オプション」コマンドを使用して、こうしたパラメーターを更新します。

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

tcp_keepidle 値と **tcp_keepintvl** 値は、0.5 秒単位で表されます。

2. **Linux の更新**

   Linux クライアントの場合、以下の 4 つのオペレーティング・システム・キープアライブ・パラメーターを変更します。

   - **tcp_keepalive_probes** - 送信したものの応答しないプローブ数がこの数を超えると、クライアントによって接続が切れていると見なされ、アプリケーション層に通知されます
   - **tcp_keepalive_time** - 最後のデータ・パケットが送信されてから、最初のキープアライブ・プローブまでの間隔
   - **tcp_keepalive_intvl** - 後続のキープアライブ・プローブ間の間隔
   - **tcp_retries2** - 試行を中止するまでに、パケットを再送する最大回数

   Linux オペレーティング・システムでは、「echo」コマンドを使用して、こうしたパラメーターを更新します。

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

   tcp_keepalive_time 値と **tcp_keepalive_intvl** 値は、秒単位で表されます。システムの再始動後にもこれらの値を保持するには、/etc/sysctl.conf ファイルに追加する必要があります。

次のタスク

その他のクライアント・プラットフォームの場合の TCP/IP キープアライブ値の設定方法については、ご使用のオペレーティング・システム資料を参照してください。

関連情報:

AIX ネットワーク・オプション・コマンド

Linux における TCP/IP キープアライブの使用法

Microsoft Windows TCP/IP レジストリー項目

JDBC 以外の高可用性クライアントの TCP/IP キープアライブ・パラメーターの構成 (AIX、HP-UX、Linux、Windows)

クライアントでキープアライブ・パラメーターを設定するために推奨されている方法は、db2dsdriver.cfg 構成ファイルで **keepAliveTimeout** パラメーターを使用する方法です。

このタスクについて

こうしたコマンドでは推奨値が提供されますが、ご使用のネットワークやサーバーの能力に応じてそれらの設定を微調整してください。

手順

IBM Data Server Driver for JDBC and SQLJ を使用しないクライアントの場合、2 通りの方法で TCP/IP キープアライブ・パラメーターを更新できます。

- db2dsdriver.cfg ファイルを変更します。

  このパラメーターを設定するには、db2dsdriver.cfg ファイルを編集し、 <acr> セクションの外に **keepAliveTimeout** 行を配置します。ただし、引き続き <databases> 親セクション内に存在するようにしてください。以下に例を示します。

  ```
  <configuration>
    <dsncollection>
      <dsn alias="D3D" name="D3D" host="DB2PS-member0" port="5912" />
    </dsncollection>
    <databases>
      <database name="D3D" host="DB2PS-member0" port="5912">
      <parameter name="keepAliveTimeout" value="20"/>
      <acr>
        <parameter name="enableAcr" value="true"/>
        <parameter name="enableSeamlessAcr" value="true"/>
        <parameter name="affinityFailbackInterval" value="15"/>
    </databases>
  ...
  </configuration>
  ```

  この方法は、インスタンスに基づくクライアントと、インスタンスを使用しないドライバーのどちらにも使用できるので推奨されています。また、db2dsdriver.cfg ファイルを使用すると、それぞれのデータベースの **keepAliveTimeout** 設定を別々にすることが可能です。

- **DB2TCP_CLIENT_KEEPALIVE_TIMEOUT** パラメーターを変更します。

  このタイプのクライアントでキープアライブ・パラメーターを更新するための 2 番目の方法は、**DB2TCP_CLIENT_KEEPALIVE_TIMEOUT** パラメーターを設定して、TCP/IP 通信層で障害を検出するという方法です。

  このパラメーターを更新するには、クライアントのコマンド・ウィンドウまたは端末から、以下のコマンドを発行します。

  ```
  db2set DB2TCP_CLIENT_KEEPALIVE_TIMEOUT=20
  ```

  この値は、秒単位で指定します。

  注: TCP/IP タイムアウト・キープアライブはインスタンス接続でもサポートされていますが、**DB2TCP_CLIENT_KEEPALIVE_TIMEOUT** パラメーターの値を指定するというこの 2 番目の方法を使用してのみ設定が可能です。インスタンス接続の場合、自動クライアント・リルート (ACR) は適用されません。

DB2 障害モニター・レジストリー・ファイル

障害モニター・レジストリー・ファイルは、障害モニター・デーモンが開始される時、各物理マシン上のすべての DB2 データベース・マネージャー・インスタンスごとに作成されます。このファイルのキーワードと値により、障害モニターの動作を指定します。

この障害モニター・レジストリー・ファイルは、/sqllib/ ディレクトリーにあり、fm.*machine_name*.reg という名前が付いています。このファイルは、**db2fm** コマンドを使用して変更できます。

障害モニター・レジストリー・ファイルが存在しない場合には、デフォルト値が使用されます。

以下は、障害モニター・レジストリー・ファイルの内容の例です。

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

障害モニター・レジストリー・ファイルのキーワード

FM_ON

障害モニターを始動するかしないかを指定します。値が NO に設定されている場合、障害モニター・デーモンは始動しないか、すでに始動していた場合には停止されます。デフォルト値は NO です。

FM_ACTIVE

障害モニターがアクティブであるかそうでないかの指定をします。障害モニターは、**FM_ON** および **FM_ACTIVE** が両方とも YES に設定されている時にのみ作動します。 **FM_ON** が YES に設定されており、**FM_ACTIVE** が NO に設定されている場合には、障害モニター・デーモンは始動しますが、アクティブにはなりません。これは、DB2 がシャットダウンした場合でもそれをオンラインに戻そうとしないことを意味します。デフォルト値は YES です。

START_TIMEOUT

障害モニターがモニターしているサービスを障害モニターが始動するまでの時間の最大の長さを指定します。デフォルト値は 600 秒です。

STOP_TIMEOUT

障害モニターがモニターしているサービスを障害モニターが停止するまでの時間の最大の長さを指定します。デフォルト値は 600 秒です。

STATUS_TIMEOUT

障害モニターがモニターしているサービスの状況を障害モニターが入手するまでの時間の最大の長さを指定します。デフォルト値は 20 秒です。

STATUS_INTERVAL

モニターしているサービスの状況を取得するために行う、連続した 2 回の呼び出しの間の最小時間を指定します。デフォルト値は 20 秒です。

RESTART_RETRIES

モニターしているサービスの状況を取得する際、状況が取得できなかった場合に障害モニターが試行を繰り返す回数を指定します。この数に達すると、障害モニターは、そのサービスがオフラインになっていると判断して、そのサービスをオンラインに戻そうとします。デフォルト値は 3 です。

ACTION_RETRIES

サービスをオンラインにするために障害モニターが試行する回数を指定します。デフォルト値は 3 です。

NOTIFY_ADDRESS

障害モニターが通知メッセージを送信する相手の E メール・アドレスを指定します。デフォルトは *instance_name@machine_name* です。

db2fm コマンドを使った DB2 障害モニターの構成

DB2 障害モニター・レジストリー・ファイルの変更は、**db2fm** コマンドを使用して行えます。

db2fm コマンドを使用して障害モニター・レジストリー・ファイルを更新する例をいくつか示します。

例 1: START_TIMEOUT を更新する

インスタンス DB2INST1 の START_TIMEOUT の値を 100 秒に更新するには、DB2 データベース・コマンド・ウィンドウから以下のコマンドを入力します。

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

例 2: STOP_TIMEOUT を更新する

インスタンス DB2INST1 の STOP_TIMEOUT の値を 200 秒に更新するには、以下のコマンドを入力します。

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

例 3: START_TIMEOUT および STOP_TIMEOUT を更新する

インスタンス DB2INST1 の START_TIMEOUT の値を 100 秒に更新し、STOP_TIMEOUT の値を 200 秒に更新するには、以下のコマンドを入力します。

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

例 4: 障害モニターをオンにする

インスタンス DB2INST1 の障害モニターを始動するには、以下のコマンドを入力します。

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

例 5: 障害モニターをオフにする

インスタンス DB2INST1 の障害モニターをオフにするには、以下のコマンドを入力します。

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

DB2INST1 の障害モニターが実行されなくなったことを確認するには、UNIX システムで以下のコマンドを入力します。

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

Linux では、以下のコマンドを入力します。

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

db2fmd と DB2INST1 を示す項目は、そのインスタンスに対する障害モニターが依然として実行中であることを示します。障害モニターをオフにするには、インスタンス所有者として以下のコマンドを入力します。

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

db2fmc およびシステム・コマンドを使った DB2 障害モニターの構成

DB2 障害モニターの構成は、DB2 Fault Monitor Controller Utility (FMCU) コマンドの **db2fmcu**、またはシステム・コマンドを使用して行えます。

db2fmcu およびシステム・コマンドを使用して障害モニターを構成する例をいくつか示します。

例 1: FMC が起動されないようにする

DB2 Fault Monitor Controller Utility (FMCU) を使用して FMC が起動されないようにすることもできます。FMCU はシステムの inittab ファイルにアクセスするので、ルートとして実行しなければなりません。FMC の実行をブロックするには、ルートとして以下のコマンドを入力します。

```
db2fmcu -d
```

注: DB2 Data Server フィックスパックを適用する場合は、inittab が FMC を組み込むように再構成されるので、このコマンドはリセットされます。フィックスパックの適用後に FMC が起動されないようにするには、この例に示すコマンドを再発行する必要があります。

例 2: FMC が起動されるように組み込む

db2fmcu -d コマンドの逆に、FMC を組み込むように inittab を再構成するには、以下のコマンドを入力します。

```
db2fmcu -u -p fullpath
```

fullpath は、**db2fmcd** オブジェクトの完全パスです (例えば /opt/IBM/db2/bin/db2fmcd)。

例 3: DB2 データベース・マネージャー・インスタンスを自動的に開始する

システムが最初にブートされる際に FMC がインスタンスを自動的に開始できるようにすることもできます。インスタンス DB2INST1 に関してこのフィーチャーを使用可能にするには、以下のコマンドを入力します。

```
db2iauto -on db2inst1
```

例 4: インスタンスの自動始動を無効にする

自動始動の動作をオフにするには、以下のコマンドを入力します。

```
db2iauto -off db2inst1
```

例 5: 障害モニター・プロセスが起動されないようにする

特定のインスタンスのグローバル・レジストリー・レコード中のフィールドに変更を加えて、システム上でそのインスタンスに関する障害モニター・プロセスが起動しないようにすることもできます。グローバル・レジストリー・フィールドに変更を加えて、インスタンス DB2INST1 に関する障害モニターを使用不可にするには、ルートとして以下のコマンドを入力します。

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

このコマンドを逆にして、インスタンス DB2INST1 に関する障害モニターを再び使用できるようにするには、ルートから以下のコマンドを入力します。

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

高可用性災害時リカバリーの初期設定 (HADR)

1 次データベースとスタンバイ・データベースを、単一スタンバイ・モードで DB2 高可用性災害時リカバリー (HADR) 用にセットアップして初期設定するには、以下の手順を使用します。

単一スタンバイ・モードで HADR を初期設定する場合は、220 ページの『複数スタンバイ・モードでの HADR の初期化』を参照してください。

このタスクについて

HADR は、コマンド行プロセッサー (CLP) を使用するか、db2HADRStart API を呼び出して初期設定できます。

手順

CLP を使用して、初めてシステム上で HADR を初期設定する場合、次のようにします。

1. 各 HADR データベースのホスト名、ホスト IP アドレス、およびサービス名かポート番号を判別します。

   ホストに複数のネットワーク・インターフェースがある場合、 HADR ホスト名または IP アドレスが目的のインターフェースにマッピングされるようにします。保護されるデータベースごとに、/etc/services で個別の HADR ポートを割り振る必要があります。これらは、インスタンスに割り振られるポートと同じであってはなりません。ホスト名は、1 つの IP アドレスにだけマッピング可能です。

   注: 1 次データベースとスタンバイ・データベースのインスタンス名は、同じである必要はありません。
2. 1 次データベースとして設定する予定の既存のデータベースに基づき、バックアップ・イメージをリストアするか、スプリット・ミラーを初期設定して、スタンバイ・データベースを作成します。

次の例では、**BACKUP DATABASE** および **RESTORE DATABASE** コマンドが使用され、データベース SOCKS がスタンバイ・データベースとして初期設定されます。この場合、NFS がマウントされたファイル・システムは、両方のサイトでアクセス可能であるものとします。

1 次データベースで次のコマンドを発行します。

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

スタンバイ・データベースで次のコマンドを発行します。

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

次の例には、**db2inidb** ユーティリティーを使用して、 1 次データベースのスプリット・ミラーを使用したスタンバイ・データベースを初期設定する方法が示されています。この手順は、前述のバックアップおよびリストアの手順の代わりに実行できるものです。

スタンバイ・データベースで次のコマンドを発行します。

```
db2inidb socks as standby
```

注:

a. 1 次データベースとスタンバイ・データベースのデータベース名は、同じでなければなりません。

b. リストア操作後またはスプリット・ミラー初期設定後は、スタンバイ・データベースで **ROLLFORWARD DATABASE** コマンドを発行しないでください。ロールフォワード操作を使用したときの結果は、スタンバイ・データベースで HADR を使用してログを再生する場合とは、若干異なる場合があります。データベースが同じでない場合は、スタンバイ・データベースを開始しようとすると失敗します。

c. **RESTORE DATABASE** コマンドで REPLACE HISTORY FILE オプションを使用してください。

d. **RESTORE DATABASE** コマンドを使用してスタンバイ・データベースを作成する場合は、スタンバイ・データベースがロールフォワード・ペンディング・モードまたはロールフォワード進行中モードのままになるようにしてください。つまり、**ROLLFORWARD DATABASE** コマンドに COMPLETE オプションまたは STOP オプションを指定して発行することはできないということです。ロールフォワードを停止した後で、AS STANDBY オプションを指定した **START HADR** コマンドをデータベースに対して試行すると、エラーが戻されます。

e. スタンバイ・データベースを設定するときには、以下の **RESTORE DATABASE** コマンド・オプションを避ける必要があります。 TABLESPACE、INTO、REDIRECT、および WITHOUT ROLLING FORWARD。

f. **db2inidb** ユーティリティーを使用してスタンバイ・データベースを設定する場合、 SNAPSHOT または MIRROR オプションは使用しないでください。構成属性である、インスタンス名、ログ・パス、およびデータベース・パスの 1 つ以上を変更するには、RELOCATE USING オプションを指定できます。しかし、データベース名または表スペース・コンテナー・パスを変更してはなりません。

3. 1 次データベースおよびスタンバイ・データベースで、以下の HADR 構成パラメーターを設定します。

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

これらの構成パラメーターは、スタンバイ・データベースを作成した後に設定する必要があります。これらを、スタンバイ・データベースの作成前に設定する場合、スタンバイ・データベース側での設定は、 1 次データベースでの設定内容を反映します。

注: これは一般的な HADR セットアップです。より詳細な構成オプションおよび設定については、次のリンクを参照してください。

4. スタンバイ・インスタンスに接続し、次の例のように、スタンバイ・データベースで HADR を開始します。

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

注: 通常は、スタンバイ・データベースが最初に開始されます。 1 次データベースを最初に開始する場合、**hadr_timeout** データベース構成パラメーターで指定した時間間隔内にスタンバイ・データベースが開始されなければ、この開始手順は失敗します。
スタンバイ・データベースは開始後に、ローカルに使用可能なログ・ファイルが読み取られて再生されるローカル・キャッチアップ 状態になります。ローカル・ログをすべて再生した後、リモート・キャッチアップ・ペンディング 状態になります。

5. 1 次インスタンスに接続し、次の例のように、1 次データベースで HADR を開始します。

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

1 次データベースの開始後、スタンバイ・データベースはリモート・キャッチアップ 状態になり、1 次データベースからログ・ページを受け取って再生します。 1 次データベース・マシンのディスク上にあるログ・ファイルをすべて再生すると、両方のデータベースがピア 状態になります。

関連情報:

自動クライアント・リルートおよび高可用性災害時リカバリー (HADR) の構成

自動クライアント・リルート・フィーチャーと高可用性災害時リカバリー (HADR) フィーチャーを併用することにより、障害が発生したデータベース・サーバーからスタンバイ・データベース・サーバーにクライアント・アプリケーション要求を転送することができます。

制約事項

- リルートは、サーバーで代替データベース・ロケーションが指定されている場合にのみ可能です。
- 自動クライアント・リルートは、TCP/IP プロトコルでのみサポートされています。

構成に関する詳細

- **UPDATE ALTERNATE SERVER FOR DATABASE** コマンドを使用して、自動クライアント・リルートを使用できるようにします。
- 自動クライアント・リルートでは、**hadr_remote_host** および **hadr_remote_svc** データベース構成パラメーターを使用しません。
- 複数スタンバイ・モードでは、自動クライアント・リルートに指定できるスタンバイ・データベースは 1 つだけです。
- 代替ホストのロケーションは、サーバーのシステム・データベース・ディレクトリー・ファイルに保管されています。
- 自動クライアント・リルートが使用可能になっていない場合、クライアント・アプリケーションは、エラー・メッセージ SQL30081N を受け取り、サーバーとの接続を確立するための処置は行われません。

UPDATE ALTERNATE SERVER FOR DATABASE コマンドを使用した自動クライアント・リルートと HADR のセットアップ

システムは、以下のようにしてセットアップします。

- データベース MUSIC がホスト HORNET に存在するものとしてカタログされているクライアントがあります。
- データベース MUSIC は 1 次データベースであり、対応するスタンバイ・データベース MUSIC は、ホスト MONTERO のポート番号 456 (**svcename** 構成パラメーターで割り当てられたもの) に存在します。

自動クライアント・リルートを使用可能にするために、以下のようにしてホスト HORNET のデータベース MUSIC の代替サーバーを更新します。

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

このコマンドの発行後、クライアントは、ホスト HORNET へ正常に接続して、代替サーバー情報を入手することになります。次に、クライアントとホスト HORNET のデータベース MUSIC との間で通信エラーが発生する場合、クライアントは、まずホスト HORNET のデータベース MUSIC へ再接続しようとします。これが失敗する場合、次にクライアントは、ホスト MONTERO のスタンバイ・データベース MUSIC との接続を確立しようとします。

索引ロギングおよび高可用性災害時リカバリー (HADR)

高可用性災害時リカバリー (HADR) データベースのためのデータベース構成パラメーター **logindexbuild** および **indexrec** を設定することを検討してください。

logindexbuild データベース構成パラメーターの使用

推奨: HADR データベースについては、**logindexbuild** データベース構成パラメーター ON に設定し、索引作成、再作成、および再編成のために完全な情報がログに記録されるようにしてください。索引作成は 1 次システムで時間がかかることがあり、より多くのログ・スペースが必要になる場合がありますが、索引は、HADR ログの再生時にスタンバイ・システムで再作成され、フェイルオーバーが生じるときに使用できるようになります。この推奨事項に従わないと、索引の作成や再作成のイベントを再生する際に、ログ・レコード内の情報が不完全で新しい索引にデータを追加できないので、スタンバイ・システムは索引に無効としてマークを付けま

す。 1 次システムでの索引作成がログに記録されていない場合で、フェイルオーバーが生じた場合、フェイルオーバーの完了後に無効の状態になっている索引を、アクセスされる前に再作成する必要があります。索引の再作成中には、アプリケーションでアクセスすることはできません。

注: LOG INDEX BUILD 表属性がデフォルト値である NULL に設定される場合、DB2 は、 **logindexbuild** データベース構成パラメーターに指定された値を使用します。LOG INDEX BUILD 表属性が ON または OFF に設定される場合、**logindexbuild** データベース構成パラメーターに指定された値は無視されます。

以下のいずれかの理由により、1 つ以上の表で LOG INDEX BUILD 表属性を OFF に設定するよう選択できます。

- 索引作成のロギングをサポートするための十分なアクティブ・ログ・スペースがない。
- 索引データが非常に大きく、表が頻繁にアクセスされないため、テークオーバー操作の終了時に索引を再作成できる。この場合、**indexrec** 構成パラメーターを RESTART に設定します。表が頻繁にアクセスされないため、この設定では、システムは、テークオーバー操作後に初めて表がアクセスされることを待つのではなく、テークオーバー操作の終了時に索引を再作成します。

1 つ以上の表で LOG INDEX BUILD 表属性が OFF に設定される場合、それらの表に索引作成操作を実行すると、テークオーバー操作を実行するときに索引が再作成されます。同様に、LOG INDEX BUILD 表属性がデフォルト値の NULL に設定され、**logindexbuild** データベース構成パラメーターが OFF に設定される場合、表に対して索引作成操作を実行すると、その表の索引はテークオーバー操作を実行するときに再作成されます。以下のいずれかのアクションを実行することで、索引が再作成されることを防げます。

- 新しい 1 次データベースですべての無効な索引が再作成された後で、データベースのバックアップをとり、スタンバイ・データベースに適用します。 1 次データベース上における無効な索引は通常スタンバイ・データベースで再作成が必要であるとみなされますが、このように 1 次データベースのバックアップをスタンバイ・データベースに適用することで、スタンバイ・データベースは、1 次データベース上で無効な索引作成するために使用するログを適用する必要がなくなります。
- LOG INDEX BUILD 表属性を ON に設定するか、LOG INDEX BUILD 表属性を NULL に設定し、スタンバイ・データベースで **logindexbuild** 構成パラメーターを ON に設定し、索引再作成がログに記録されるようにします。

indexrec データベース構成パラメーターの使用

推奨: 1 次データベースとスタンバイ・データベースの両方で、**indexrec** データベース構成パラメーターを RESTART (デフォルト) に設定してください。これにより、テークオーバー操作の完了後に、無効な索引が再作成されます。索引作成がログに記録されていない場合、この設定により、 DB2 は、無効な索引を調べて再作成できます。このプロセスは、バックグラウンドで生じるものであり、テークオーバー操作が正常に完了した後でも、データベースにアクセスできます。

索引がバックグラウンド索引再作成プロセスによって再作成される前に、特定のトランザクションが無効な索引を含む表にアクセスする場合、無効な索引は、それに

アクセスした最初のトランザクションによって再作成されます。

高可用性災害時リカバリー用のデータベース構成 (HADR)

データベース構成パラメーターは、DB2 HADR での最良のパフォーマンスを得るために役立ちます。

ほとんどの場合、データベース構成パラメーターの設定とデータベース・マネージャー構成パラメーターの設定は、1 次データベースとスタンバイ・データベースが置かれているシステムで同一にする必要があります。スタンバイ・データベースの構成パラメーターの設定が 1 次の設定と異なる場合、以下の問題が発生する可能性があります。

- 1 次データベースから送られたログ・ファイルの再生中に、スタンバイ・データベースにエラー・メッセージが戻されることがあります。
- テークオーバー操作後に、新しい 1 次データベースはワークロードを処理できない可能性があるため、パフォーマンス上の問題が生じるか、アプリケーションが元の 1 次データベースに接続されていたときには受け取ることのなかったエラー・メッセージを受け取ります。

1 次データベースの構成パラメーターの変更内容は、自動的にスタンバイ・データベースに伝搬されません。スタンバイ・データベースで手動で変更する必要があります。動的構成パラメーターの場合、データベース管理システム (DBMS) またはデータベースをシャットダウンして再始動しなくても、変更は有効になります。動的構成パラメーターではない場合、変更はスタンバイ・データベースの再始動後に有効になります。

以下に、HADR に関する特定の構成トピックのセクションを示します。

- 『スタンバイ・データベースでのログ・ファイルのサイズ構成パラメーター』
- 42 ページの『スタンバイ・データベースでのログ受信バッファー・サイズ』
- 42 ページの『ロード操作および HADR』
- 43 ページの『DB2_HADR_PEER_WAIT_LIMIT レジストリー変数』
- 44 ページの『HADR 構成パラメーター』

スタンバイ・データベースでのログ・ファイルのサイズ構成パラメーター

前の段落で説明した構成パラメーターの動作に関する 1 つの例外として、**logfilsiz** データベース構成パラメーターの動作があります。このパラメーターの値はスタンバイ・データベースに複製されませんが、両方のデータベースでログ・ファイルを同一にするために、スタンバイの **logfilsiz** 構成パラメーターの設定は無視されます。代わりに、スタンバイ・データベースでは、1 次データベースのログ・ファイルのサイズと一致するサイズのローカル・ログ・ファイルが作成されます。

テークオーバー後、元のスタンバイ (新しい 1 次) は、元の 1 次で設定された **logfilsiz** パラメーター値を、データベースを再始動するまで使用します。その時点で、新しい 1 次は、ローカルに設定された値を再び使用します。さらに、新しい 1 次では、現在のログ・ファイルが切り捨てられ、以前に作成されたログ・ファイルのサイズが変更されます。

データベースが非強制テークオーバーの結果として役割の切り替えを維持し、どちらのデータベースも非アクティブにされない場合、使用されるログ・ファイル・サイズは常に元の 1 次データベースのサイズになります。ただし、元のスタンバイ (新しい 1 次) が非アクティブにされてから再始動されると、新しい 1 次ではローカルに構成されたログ・ファイル・サイズが使用されます。元の 1 次が再びテークオーバーする場合、このログ・ファイル・サイズが継続して使用されます。ログ・ファイル・サイズが元の 1 次の設定に戻るのは、元の 1 次の非アクティブ化と再始動が終わってからです。

スタンバイ・データベースでのログ受信バッファー・サイズ

デフォルトでは、スタンバイ・データベースでのログ受信バッファー・サイズは、1 次データベースの **logbufsz** 構成パラメーターに指定した値の 2 倍です。このサイズでは十分でない可能性があります。例えば、HADR 同期モードが ASYNC に設定されており、1 次データベースとスタンバイ・データベースがピア状態である場合に何が起こり得るかを考えます。また、1 次データベースのトランザクション負荷が高くなっている場合、スタンバイ・データベースのログ受信バッファーの容量がいっぱいになる可能性があり、1 次データベースからのログ・シッピング操作が停止する可能性があります。これらの一時的なピークを管理するために、以下の構成変更のいずれかを行うことができます。

- **DB2_HADR_BUF_SIZE** レジストリー変数の値を変更して、スタンバイ・データベースのログ受信バッファーのサイズを増やす。
- **hadr_spool_limit** 構成パラメーターを設定して、スタンバイ・データベースでログ・スプーリングを有効にする。

ロード操作および HADR

1 次データベースで **COPY YES** パラメーターを指定した **LOAD** コマンドを発行すると、コマンドは 1 次データベースで実行され、コマンドで指定したパスまたは装置経由でロード・コピーにアクセスできる場合は、データがスタンバイ・データベースに複製されます。スタンバイ・データベースからロード・コピー・データにアクセスできない場合、表が保管される表スペースがスタンバイ・データベース上で無効としてマークされます。この表スペースに関係する以降のログ・レコードはスキップされます。ロード操作がスタンバイ・データベース上のロード・コピーに確実にアクセスできるようにするため、**COPY YES** パラメーターからの出力ファイルに共有ロケーションを使用します。別の方法として、1 次でのロードの実行中に、スタンバイ・データベースを非アクティブにして、出力ファイルのコピーをスタンバイ・パスに置いてから、スタンバイ・データベースをアクティブにすることもできます。

1 次データベースで **NONRECOVERABLE** パラメーターを指定した **LOAD** コマンドを発行すると、そのコマンドは 1 次データベースで実行され、スタンバイ・データベースの表は無効としてマークされます。この表に関係する以降のログ・レコードはスキップされます。**COPY YES** および **REPLACE** パラメーターを指定した **LOAD** コマンドを発行して表を戻すことも、表をドロップしてスペースをリカバリーすることもできます。

COPY NO パラメーターを指定したロード操作は HADR ではサポートされないため、操作は **NONRECOVERABLE** パラメーターを指定したロード操作に自動的に変換さ

れます。 **COPY NO** パラメーターを指定したロード操作を、**COPY YES** パラメーターを指定したロード操作へ変換できるようにするには、 1 次データベースで **DB2_LOAD_COPY_NO_OVERRIDE** レジストリー変数を設定します。このレジストリー変数はスタンバイ・データベースでは無視されます。 1 次データベースに指定した装置またはディレクトリーが、スタンバイ・データベースで、同じパス、装置、またはロード・ライブラリーを使用してアクセス可能であることを確認してください。

Tivoli Storage Manager (TSM) ソフトウェアを使用して、**COPY YES** パラメーターを指定したロード操作を実行する場合、1 次データベースおよびスタンバイ・データベースで **vendoropt** 構成パラメーターを設定しなければならない場合があります。TSM の構成方法に応じて、1 次データベースの値とスタンバイ・データベースの値が同じでない場合があります。さらに、TSM を使用して、**COPY YES** パラメーターを指定したロード操作を実行するときには、**GRANT** パラメーターを指定した **db2adutl** コマンドを発行し、スタンバイ・データベースに、ロードされるファイルの読み取り権限を与える必要があります。

表データが、**COPY YES** パラメーターをロード操作によって複製される場合、索引は、次のようにして複製されます。

- **LOAD** コマンドに REBUILD 索引付けモード・オプションを指定する場合、LOG INDEX BUILD 表属性が ON に設定 (ALTER TABLE ステートメントを使用して) されているか、または NULL に設定されていて **logindexbuild** データベース構成パラメーターが ON に設定されていると、1 次データベースは、再作成された索引オブジェクト (つまり、表に対して定義されているすべての索引) をコピー・ファイルに組み込み、スタンバイ・データベースが索引オブジェクトを複製できるようにします。ロード操作の前に、スタンバイ・データベースの索引オブジェクトが無効としてマークされる場合、索引の再作成の結果として、ロード操作後に再び使用可能になります。
- **LOAD** コマンドに INCREMENTAL 索引付けモード・オプションを指定する場合、LOG INDEX BUILD 表属性が ON に設定 (ALTER TABLE ステートメントを使用して) されているか、または NULL に設定されていて 1 次データベース上の **logindexbuild** データベース構成パラメーターが ON に設定されていると、スタンバイ・データベース上の索引オブジェクトは、ロード操作の前に無効のマークが付けられていない場合にのみ更新されます。それ以外の場合には、索引はスタンバイ・データベース上で無効としてマークされます。

DB2_HADR_PEER_WAIT_LIMIT レジストリー変数

制約事項: 複数スタンバイ・モードでは、このセクションのいずれも補助スタンバイには適用されません。補助スタンバイは SUPERASYNC 同期モードであるため、ピア状態にはならないためです。

DB2_HADR_PEER_WAIT_LIMIT レジストリー変数を設定すると、スタンバイへのログ・レプリケーションのために HADR 1 次データベースでのロギングが指定秒数の間ブロックされた場合、その 1 次データベースのピア状態が解除されます。この限度に達すると、1 次データベースはスタンバイ・データベースへの接続を切断します。 **hadr_peer_window** 構成パラメーターを 0 に設定して、ピア・ウィンドウを使用不可にすると、1 次は切断済み状態になり、ロギングが再開します。ピア・ウィンドウを使用可能にすると、1 次データベースは切断済みピア状態になり、ロギングのブロックが継続されます。1 次データベースは、再接続時またはピア・ウィン

ドウの有効期限が切れた時に切断済みピア状態ではなくなります。 1 次データベースが切断済みピア状態でなくなると、ロギングが再開します。

データベースがピア状態から解除されるときのピア・ウィンドウの遷移を考慮すると、すべての事例で安全にテークオーバーが行われるようにピア・ウィンドウのセマンティクスが守られます。遷移の際に 1 次に障害が発生した場合でも、通常のピア・ウィンドウ保護が引き続き適用されます (引き続き切断済みピア状態である場合、スタンバイから安全にテークオーバーされます)。

スタンバイ・データベースの側では、切断の後、データベースは既に受け取ったログの再生を続行します。受け取ったログが再生されると、スタンバイ・データベースは 1 次データベースに再接続します。受け取ったログが再生されると、スタンバイは 1 次に再接続します。再接続時に、通常の状態遷移が行われます (まず、リモート・キャッチアップ状態になってから、ピア状態になります)。

hadr_timeout データベース構成パラメーターとの関係

1 次データベースがブロック中にスタンバイ・データベースからのハートビート・メッセージを受け取り続けている場合、**hadr_timeout** データベース構成パラメーターによって 1 次データベースがピア状態から解除されることはありません。 **hadr_timeout** データベース構成パラメーターでは、HADR ネットワーク層のタイムアウト値を指定します。HADR データベースは、パートナー・データベースから **hadr_timeout** 構成パラメーターで指定された期間にまったくメッセージを受け取らなかった場合、そのパートナー・データベースへの接続を切断します。このタイムアウトでログ・シッピングや ACK (確認通知) シグナルなどの高位層の操作に対するタイムアウトは制御されません。スタンバイ・データベースでのログ再生がロードや再編成などの大規模な操作で滞っている場合でも、HADR コンポーネントは通常のスケジュールで 1 次データベースにハートビート・メッセージを送信します。そのようなシナリオでは、スタンバイの再生がブロックされている間、1 次はブロックされます (**DB2_HADR_PEER_WAIT_LIMIT** レジストリー変数を設定した場合を除く)。

DB2_HADR_PEER_WAIT_LIMIT レジストリー変数を設定すると、接続状況に関係なく、1 次のロギングのブロックが解除されます。
DB2_HADR_PEER_WAIT_LIMIT レジストリー変数を設定していない場合でも、ネットワーク・エラーが検出された場合、または接続がクローズされた場合 (**hadr_timeout** 構成パラメーターの結果として発生する可能性がある) には、常に 1 次のピア状態が解除されます。

HADR 構成パラメーター

一部の HADR 構成パラメーター (**hadr_local_host** や **hadr_remote_host** など) は静的です。静的パラメーターはデータベースの始動時にロードされ、変更内容は実行時には無視されます。HADR パラメーターは、**START HADR** コマンドの完了時にもロードされます。1 次データベースの場合、データベースをオンラインにしたままの状態で、HADR を動的に開始および停止できます。したがって、データベースをシャットダウンせずに HADR 構成パラメーターの実効値をリフレッシュする 1 つの方法として、HADR を停止して再始動することができます。一方、**STOP HADR** をスタンバイで実行するとデータベースが停止するため、スタンバイのパラメーターはデータベースをオンラインにした状態ではリフレッシュできません。

ホスト名パラメーターおよびサービス名とポート名のパラメーター (単一スタンバイ・モード)

HADR ペアでは、以下のような、相互に関連する 5 つの構成パラメーターを設定する必要があります。

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

TCP 接続は、1 次データベースとスタンバイ・データベース間で通信する際に使用されます。「local」パラメーターはローカル・アドレスを指定し、「remote」パラメーターはリモート・アドレスを指定します。 1 次データベースは、ローカル・アドレスで新しい接続を listen します。 1 次データベースに接続されていないスタンバイ・データベースは、リモート・アドレスへの接続を再試行します。

スタンバイ・データベースもローカル・アドレスで listen します。一部のシナリオでは、別の HADR データベースがこのアドレスでスタンバイ・データベースに接続して、メッセージを送信できます。

HADR_NO_IP_CHECK レジストリー変数が設定されている場合を除き、HADR では以下のローカル・アドレスとリモート・アドレスでの接続のクロスチェックを行います。

```
my local address = your remote address
```

および

```
my remote address = your local address
```

このチェックは、構成パラメーターのリテラル・ストリングではなく、IP アドレスとポート番号を使用して行われます。チェックを回避するには、NAT (ネットワーク・アドレス変換) 環境で **HADR_NO_IP_CHECK** レジストリー変数を設定する必要があります。

HADR データベースは、IPv4 または IPv6 のどちらかを使用して、そのパートナー・データベースを見つけるように構成することができます。ホスト・サーバーが IPv6 をサポートしていない場合は、IPv4 を使用する必要があります。サーバーが IPv6 をサポートしている場合、データベースが IPv4 か IPv6 のどちらを使用するかは、**hadr_local_host** および **hadr_remote_host** 構成パラメーターに指定されるアドレスのフォーマットによって決まります。データベースは、2 つのパラメーターを同一の IP フォーマットに解決し、可能な場合は IPv6 の使用を試みます。以下の表は、IPv6 が使用可能なサーバーの場合に IP モードが決定される方法を示します。

hadr_local_host パラメーターに使用される IP モード	**hadr_remote_host パラメーターに使用される IP モード**	**HADR 通信に使用される IP モード**
IPv4 アドレス	IPv4 アドレス	IPv4
IPv4 アドレス	IPv6 アドレス	Error

hadr_local_host パラメーターに使用される IP モード	hadr_remote_host パラメーターに使用される IP モード	HADR 通信に使用される IP モード
IPv4 アドレス	ホスト名、IPv4 のみにマップ	IPv4
IPv4 アドレス	ホスト名、IPv6 のみにマップ	Error
IPv4 アドレス	ホスト名、IPv4 と v6 にマップ	IPv4
IPv6 アドレス	IPv4 アドレス	Error
IPv6 アドレス	IPv6 アドレス	IPv6
IPv6 アドレス	ホスト名、IPv4 のみにマップ	Error
IPv6 アドレス	ホスト名、IPv6 のみにマップ	IPv6
IPv6 アドレス	ホスト名、IPv4 と IPv6 にマップ	IPv6
ホスト名、IPv4 のみにマップ	IPv4 アドレス	IPv4
ホスト名、IPv4 のみにマップ	IPv6 アドレス	Error
ホスト名、IPv4 のみにマップ	ホスト名、IPv4 のみにマップ	IPv4
ホスト名、IPv4 のみにマップ	ホスト名、IPv6 のみにマップ	Error
ホスト名、IPv4 のみにマップ	ホスト名、IPv4 と IPv6 にマップ	IPv4
ホスト名、IPv6 のみにマップ	IPv4 アドレス	Error
ホスト名、IPv6 のみにマップ	IPv6 アドレス	IPv6
ホスト名、IPv6 のみにマップ	ホスト名、IPv4 のみにマップ	Error
ホスト名、IPv6 のみにマップ	ホスト名、IPv6 のみにマップ	IPv6
ホスト名、IPv6 のみにマップ	ホスト名、IPv4 と IPv6 にマップ	IPv6
ホスト名、IPv4 と IPv6 にマップ	IPv4 アドレス	IPv4
ホスト名、IPv4 と IPv6 にマップ	IPv6 アドレス	IPv6
ホスト名、IPv4 と IPv6 にマップ	ホスト名、IPv4 のみにマップ	IPv4
ホスト名、IPv4 と IPv6 にマップ	ホスト名、IPv6 のみにマップ	IPv6
ホスト名、IPv4 と IPv6 にマップ	ホスト名、IPv4 と IPv6 にマップ	IPv6

1 次およびスタンバイ・データベースが同じ IPv4 または IPv6 フォーマットを使用する場合にのみ、両者は HADR 接続を作成できます。一方のサーバーが IPv6 に対応 (ただし、IPv4 もサポート) していて、もう一方のサーバーが IPv4 しかサポートしていない場合、IPv6 サーバー上の **hadr_local_host** および **hadr_remote_host** パラメーターのうち少なくとも 1 つは IPv4 アドレスを指定して、このサーバーのデータベースでの IPv4 の使用を強制する必要があります。

HADR のローカル・サービスとリモート・サービスの各パラメーター (**hadr_local_svc** および **hadr_remote_svc**) は、ポート番号またはサービス名のいずれかに設定できます。指定した値は、他のサービス (他の DB2 コンポーネントや他の HADR データベースを含む) で使用されていないポートにマップする必要があります。特に、どちらのパラメーター値も、サーバーがリモート・クライアントからの通信を待機するために使用する TCP/IP ポート (**svcename** データベース・マネージャー構成パラメーターの値)、またはその次のポート (**svcename** パラメーターの値 + 1) に設定することはできません。

1 次データベースとスタンバイ・データベースが異なるサーバー上にある場合、両方のデータベースで同じポート番号またはサービス名を使用できます。それ以外の場合は、異なる値を使用する必要があります。

ホスト名、サービス名またはポート名、およびターゲット・リストのパラメーター (複数スタンバイ・モード)

複数スタンバイ・モードでは、**hadr_local_host**、**hadr_local_svc**、**hadr_remote_host**、**hadr_remote_host**、および **hadr_remote_inst** の各構成パラメーターを引き続き設定する必要があります。これらのパラメーターを正しく設定しないと、**hadr_target_list** 構成パラメーターの設定を使用して、1 次がスタンバイに接続された後で自動的に更新されます。このパラメーターは、すべてのスタンバイのホスト名とポート名を指定します。ターゲット・リストに指定した最初のスタンバイが、*プリンシパル HADR スタンバイ・データベース* と見なされます。

複数スタンバイ・モードでは、**hadr_local_host**、**hadr_local_svc**、**hadr_remote_host**、**hadr_remote_host**、および **hadr_remote_inst** の各構成パラメーターを引き続き設定する必要があります。 **hadr_local_host** パラメーターと **hadr_local_svc** パラメーターは、単一スタンバイ・モードと同じ意味を持ちます。1 次データベースで、**hadr_remote_host**、**hadr_remote_host**、および **hadr_remote_inst** を設定して、そのプリンシパル・スタンバイを示します。新しいパラメーター **hadr_target_list** は、プリンシパル・スタンバイである最初の項目を含む、すべてのスタンバイをリストするために使用されます。スタンバイでは、「remote」パラメーターを設定して 1 次を示します。特定の条件では、(1 次とスタンバイの両方の) 「remote」パラメーターを自動的に更新することができます。詳しくは、226 ページの『複数 HADR スタンバイ・データベース用のデータベース構成』の『HADR パラメーターの自動再構成』を参照してください。

同期モード

単一スタンバイ・モードでは、同期モード (**hadr_syncmode** 構成パラメーターで指定) は 1 次データベースとスタンバイ・データベースで同じでなければなりません。この構成パラメーターの値の整合性は、HADR ペアが接続を確立するときに検査されます。

複数スタンバイ・モードでは、同期モードが同じである必要はありません。すべてのスタンバイには、スタンバイのタイプによって判別される*有効同期モード* が用意されています。プリンシパル・スタンバイでは 1 次の同期モードを使用し、補助スタンバイでは SUPERASYNC を使用します。すべてのスタンバイに*構成済み同期モード* が用意されています。このモードは、**hadr_syncmode** の明示的な設定であり、スタンバイが新しい 1 次になる場合に使用されます。

詳しくは、『DB2 高可用性災害時リカバリー (HADR) 同期モード』を参照してください。

HADR のタイムアウトとピア・ウィンドウ

タイムアウト期間 (**hadr_timeout** 構成パラメーターで指定) は 1 次データベースとスタンバイ・データベースで同じでなければなりません。これらの構成パラメーターの値の整合性は、HADR ペアが接続を確立するときに検査されます。

ただし、1 つの例外があります。1 次データベースは始動時に、以下の 2 つのうち長い方の時間の間、スタンバイの接続を待機します。

- 最低 30 秒
- **hadr_timeout** データベース構成パラメーターで指定される秒数。

指定された時間内にスタンバイが接続しない場合、開始は失敗します。この動作に対する例外が 1 つあります。それは、**BY FORCE** パラメーターを指定して **START HADR** コマンドを発行する場合です。この場合、1 次データベースは、スタンバイ・データベースが接続されるのを待機せずに始動します。

複数スタンバイ・モードでは、1 次が待機するのは、プリンシパル・スタンバイへの接続のみです。補助スタンバイへの接続はオプションです。

HADR のペアが接続を確立した後、両者はハートビート・メッセージを交換します。ハートビート・インターバルは、**hadr_timeout** や **hadr_peer_window** 構成パラメーターのような要因から計算されます。これは、MON_GET_HADR 表関数の HEARTBEAT_INTERVAL フィールドで報告されます。あるデータベースが、**hadr_timeout** 構成パラメーターにより指定された秒数内に別のデータベースからメッセージを受け取らないと、切断が開始されます。この動作は、パートナー・データベースまたは介在するネットワークのいずれかの障害を検出するために、最大で、HADR データベースの **hadr_timeout** 構成パラメーターで指定される秒数を要することを意味します。**hadr_timeout** の構成パラメーターの値の設定が低すぎると、偽のアラームが出され、頻繁に切断が起きます。

hadr_peer_window 構成パラメーターをゼロ以外の値に設定した場合に、1 次がピア状態でスタンバイとの接続を失うと、1 次データベースは、スタンバイ・データベースとの接続が復元されるか、**hadr_peer_window** 構成パラメーターの時間値が経過するかのいずれか早い時点まで、トランザクションをコミットしません。

最大の可用性を得るために、**hadr_peer_window** データベース構成パラメーターのデフォルト値は 0 に設定されています。このパラメーターが 0 の場合に、1 次とスタンバイの間の接続がクローズされると、1 次データベースはすぐにピア状態からドロップしてトランザクションがブロックされないようにします。接続は、スタンバイが接続をクローズしたか、ネットワーク・エラーが検出されたか、あるいはタイムアウトに達したことが原因でクローズする可能性があります。データの整合性を高めつつ、可用性を低くするために、**hadr_peer_window** データベース構成パラメーターをゼロ以外の値に設定できます。

詳しくは、『**hadr_timeout** および **hadr_peer_window** データベース構成パラメーターの設定』を参照してください。

次のサンプル構成は、1 次データベースおよびスタンバイ・データベース用です。

1 次データベース:

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

スタンバイ・データベース:

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

hadr_timeout および hadr_peer_window データベース構成パラメーターの設定

hadr_timeout および **hadr_peer_window** データベース構成パラメーターを構成して、接続障害に対する応答を最適化できます。

hadr_timeout データベース構成パラメーター

hadr_timeout データベース構成パラメーターで指定したよりも長い時間、HADR データベースがパートナー・データベースからの通信を受け取らなかった場合、データベースは、パートナー・データベースとの接続が失われたと判断します。データベースがピア状態であるときに接続が消失した場合、**hadr_peer_window** データベース構成パラメーターが 0 より大きければ、切断済みピア状態に移り、**hadr_peer_window** データベース構成パラメーターが 0 以下であれば、リモート・キャッチアップ・ペンディング状態に移ります。状態変更は 1 次データベースおよびスタンバイ・データベースの両方に適用されます。

hadr_peer_window データベース構成パラメーター

hadr_peer_window 構成パラメーターは、**hadr_timeout** 構成パラメーターに取って代わるものではありません。**hadr_timeout** 構成パラメーターは、パ

ートナー・データベースとの接続が失われたと判断するまでに、HADR データベースが待機する時間を決定します。**hadr_peer_window** 構成パラメーターは、接続が失われた後にデータベースを切断済みピア状態にするかどうか、およびデータベースがその状態を維持する時間を決定します。HADR は、TCP ソケットでの送信、受信、またはポーリング中にネットワーク・エラーが検出されるとすぐに、接続を切断します。 HADR は 100 ミリ秒間隔でソケットをポーリングします。このため、OS によって検出されるネットワーク・エラーに対して即座に応答できます。最悪のケースの場合にのみ、HADR はタイムアウトになるまで待機し、不良な接続を切断します。この場合、障害時に実行中であったデータベース・アプリケーションは、**hadr_timeout** と **hadr_peer_window** のデータベース構成パラメーターの和と等しい時間ブロックされる可能性があります。

hadr_timeout および hadr_peer_window データベース構成パラメーターの設定

データベース・アプリケーションの待機時間は最小に抑えることが望まれます。**hadr_timeout** および **hadr_peer_window** 構成パラメーターに小さい値を設定することによって、HADR スタンバイ・データベースが 1 次データベースとの接続を失った場合に、データベース・アプリケーションが待機する必要のある時間を削減できます。ただし、**hadr_timeout** および **hadr_peer_window** 構成パラメーターに割り当てる値を選択する際に、考慮すべき詳細事項が他に 2 点あります。

- **hadr_timeout** データベース構成パラメーターは、一時的な短いネットワーク中断によって HADR 接続に対する誤ったアラームが生成されないように、十分な長さの値を設定する必要があります。例えば、**hadr_timeout** のデフォルト値は 120 秒ですが、これは多くのネットワークにおいて妥当な値です。
- **hadr_peer_window** データベース構成パラメーターは、自動化された障害対応をシステムが実行できるように、十分な長さの値を設定する必要があります。例えばクラスター・マネージャーのような HA システムによって、切断済みピア状態が終わる前に 1 次データベースの障害が検出された場合、スタンバイ・データベースへのフェイルオーバーが発生します。すべてのデータは、古い 1 次データベースから新しい 1 次データベースに複製されるため、フェイルオーバー中にデータが失われることはありません。**hadr_peer_window** が短すぎると、HA システムが障害を検出して対応する時間が十分にない可能性があります。

  注: HADR 複数スタンバイ・モードでは、プリンシパル・スタンバイは 1 次の **hadr_peer_window** (*有効ピア・ウィンドウ*) の設定を使用します。補助スタンバイ上の **hadr_peer_window** の設定は無意味です。このタイプのスタンバイは常に SUPERASYNC モードで実行されるためです。

HADR ログ・スプーリング

高可用性災害時リカバリー (HADR) ログ・スプーリング・フィーチャーにより、スタンバイでのログ再生を待たなくても、1 次でトランザクションを進行させることができます。このフィーチャーを使用可能にすると、1 次が送信したログ・データはスタンバイ上のディスクにスプール され (つまり書き込まれ)、そのログ・データは後でログ再生時に読み取られます。

ログ・スプーリング (**hadr_spool_limit** データベース構成パラメーターを設定することで使用可能になります) は HADR フィーチャーを改善したものです。再生に長い時間がかかると、1 次の新しいトランザクションは、スタンバイ・システムにログ・データを送信できないため (バッファーにデータを受信するためのスペースがない場合)、ブロックされる可能性があります。ログ・スプーリング・フィーチャーでは、スタンバイがそのバッファーのサイズによって制限されません。バッファーに収まりきらない受信データが増えると、ログ再生でディスクからデータが読み取られます。これにより、1 次でのトランザクション・ボリュームのスパイク、またはスタンバイでの (特定のタイプのログ・レコードの再生による) ログ再生のスローダウンに対するシステムの耐性をより高くすることができます。

このフィーチャーにより、1 次のログ位置とスタンバイのログ再生の間により大きなギャップ (これにより、テークオーバーの時間がより長くなる場合があります) が生じる可能性があります。スプールされたログの再生が終わるまでスタンバイは新しい 1 次として始動してトランザクションを受信できないため、スプール制限の設定は慎重に検討する必要があります。

DB2 高可用性災害時リカバリー (HADR) のログ・アーカイブ構成

DB2 高可用性災害時リカバリー(HADR) でログ・アーカイブを使用するには、すべてのログ・アーカイブ・ロケーションから自動ログ検索機能が実行できるように 1 次データベースとスタンバイ・データベースの両方を構成します。複数スタンバイ・システムの場合は、1 次データベースとすべてのスタンバイ・データベースでアーカイブを構成します。

現在の 1 次データベースだけがログ・アーカイブを実行できます。セットアップされたアーカイブ・ロケーションが 1 次データベースとスタンバイ・データベースで別々の場合、ログは 1 次データベースのアーカイブ・ロケーションにしかアーカイブされません。テークオーバーが行われると、スタンバイ・データベースが新しい 1 次データベースになります。その時以降アーカイブされるログは、元のスタンバイ・データベースのアーカイブ・ロケーションに保管されます。このような構成の場合、ログのアーカイブ先は一方のロケーションかもう一方のロケーションのどちらかであり、両方にアーカイブされることはありません。ただし、すでに元の 1 次データベースがアーカイブを済ませたいくつかのログを、テークオーバーの後で新しい 1 次データベースがアーカイブする場合は別です。複数スタンバイ・システムでは、アーカイブ・ログ・ファイルをすべてのデータベース (1 次とスタンバイ) のアーカイブ装置間で分散できます。すべてのファイルが 1 つの場所に保管されるため、共有アーカイブが推奨されます。

多くの操作で、アーカイブ・ログ・ファイルのリトリーブを行う必要があります。これらの操作には、データベースのロールフォワード、リモート・キャッチアップで HADR のスタンバイ・データベースに送るログ・ファイルの 1 次データベースでのリトリーブ、およびレプリケーション・プログラム (Q レプリケーションなど) によるログの読み取りがあります。そのため、HADR システムの共有アーカイブが推奨されます。それ以外の場合は、必要なファイルを複数のアーカイブ装置に分散でき、必要なファイルを見つけて要求元のデータベースにコピーするためにユーザー介入が必要になります。コピー先として、アーカイブ装置が推奨されます。アーカイブにコピーできない場合は、ログをオーバーフロー・ログ・パスにコピーします。最後の手段として、ログをログ・パスにコピーする方法がありますが、アクティブ・ログ・ファイルが損傷するリスクがあることに注意してください。DB2 で

は、オーバーフロー・ログ・パスとアクティブ・ログ・パスにユーザーがコピーしたファイルが自動的に削除されないため、HADR のスタンバイやアプリケーションで必要なくなった場合に、手動でファイルを削除する必要があります。

特定のシナリオとして、複数スタンバイ・モードでのテークオーバーがあります。テークオーバーの後、あるスタンバイが以前のログ位置にあることが原因で、他のスタンバイで必要なログ・ファイルの一部が新しい 1 次に含まれていない場合があります。1 次は、要求されたログ・ファイルを検出できない場合、スタンバイに通知します。これにより、スタンバイは接続をクローズしてから数秒で再接続し、再試行します。再試行期間は数分に制限されます。再試行時間が過ぎると、スタンバイはシャットダウンします。この場合、1 次にファイルをコピーして、最初の欠落ファイルから現在のログ・ファイルまでのファイルがあることを確認します。ファイルがコピーされたら、必要に応じてスタンバイを再始動してください。

スタンバイ・データベースはそのログ・パスにあるログ・ファイルを自動的に管理します。スタンバイ・データベースは、1 次データベースがログ・ファイルをアーカイブしたという通知を 1 次データベースから受けるまでは、自分のローカル・ログ・パスからログ・ファイルを削除しません。この動作は、ログ・ファイルの消失に対して付加的な保護を提供します。特定のログ・ファイルが 1 次データベースにアーカイブされる前に 1 次データベースに障害が起き、ログ・ディスクが破損した場合、スタンバイ・データベースは自分のディスクからそのログ・ファイルを削除しません。なぜなら、正常にログ・ファイルをアーカイブしたという通知を 1 次データベースから受け取っていないからです。その後スタンバイ・データベースが新しい 1 次データベースとしてテークオーバーすると、スタンバイ・データベースはリサイクルの前にそのログ・ファイルをアーカイブします。**logarchmeth1** および **logarchmeth2** 構成パラメーターの両方を使用している場合、1 次データベースが両方の方式でログ・ファイルをアーカイブするまで、スタンバイ・データベースはそのログ・ファイルをリサイクルしません。

以前にリストした利点に加え、共有ログ・アーカイブ装置は、スタンバイ・データベースが、リモート・キャッチアップ状態で 1 次を介して間接的に古いログ・ファイルをリトリーブするのではなく、ローカル・キャッチアップ状態でアーカイブからそれらのファイルを直接リトリーブできるようにすることで、キャッチアップ処理を改善します。ただし、磁気テープ・ドライブのようなシリアルのアーカイブ装置は、HADR データベースに使用しないように推奨します。シリアル装置では、読み取り操作と書き込み操作が混在するため、 1 次データベースとスタンバイ・データベースの両方で性能低下が生じる可能性があります。 1 次データベースはログ・ファイルをアーカイブする際に装置への書き込みを行い、スタンバイ・データベースは装置から読み取りを行ってログを適用します。共有するように装置を構成していない場合でも、パフォーマンスへの影響は生じる可能性があります。

高可用性災害時リカバリー (HADR) のパフォーマンス

データベース・システムのさまざまな側面、例えばネットワーク帯域幅、CPU の能力、バッファー・サイズなどを構成することにより、DB2 高可用性災害時リカバリー (HADR) データベースのパフォーマンスを改善することができます。

ネットワークは HADR セットアップの重要な部分です。 1 次からスタンバイにデータベース変更を複製して 2 つのデータベースの同期を維持するためには、ネットワーク接続を必要とするからです。

ネットワーク・パフォーマンスの最大化のための推奨事項:

- ネットワーク帯域幅がデータベース・ログ生成速度よりも大きくなるようにしてください。
- HADR 同期モードを選択するときは、ネットワーク遅延を考慮に入れてください。ネットワークの遅延の影響が及ぶのは、SYNC および NEARSYNC モードの 1 次だけです。

  SYNC モードを使用した結果として生じるシステム・パフォーマンスの低下は、他の同期モードの場合と比べて著しく大きくなることがあります。SYNC モードの場合、1 次データベースのログ・ディスクにログ・ページが正常に書き込まれて初めて、1 次データベースからスタンバイ・データベースにログ・ページが送信されます。システム整合性を守るため、1 次データベースはスタンバイからの確認通知を待ってから、アプリケーションにトランザクションが準備またはコミットされたことを通知します。スタンバイ・データベースは、受け取ったログ・ページをスタンバイ・データベース・ディスクに書き込んでから、確認通知を送信します。パフォーマンス・オーバーヘッドは、スタンバイ・データベースのログ・ページを書き込むのに必要な時間と、メッセージを 1 次に送り返すのに必要な時間を加えた時間になります。

  NEARSYNC モードの場合、1 次データベースはログ・ページの書き込みと送信を並行して実行します。その後、1 次はスタンバイからの確認通知を待ちます。スタンバイ・データベースは、ログ・ページをメモリーに受け取るとすぐに確認通知を送信します。高速ネットワークの場合、1 次データベースへのオーバーヘッドは最小限に抑えられます。1 次データベースがローカル・ログの書き込みを終えたときに、確認通知がすでに届いていることもあります。

  ASYNC モードの場合も、ログの書き込みと送信が並行して行われます。ただし、このモードの場合、1 次データベースはスタンバイ・データベースからの確認通知を待ちません。したがって、ネットワーク遅延は問題になりません。ASYNC モードのパフォーマンスのオーバーヘッドは、NEARSYNC モードの場合よりさらに小さくなります。

  SUPERASYNC モードの場合は、ネットワークの中断や輻輳のためにトランザクションがブロックされたり応答時間が長くなったりすることはありません。新しいトランザクションは、既にサブミットされたトランザクションが 1 次データベースに書き込まれると直ちに処理可能になります。したがって、ネットワーク遅延は問題になりません。非強制的テークオーバー操作完了の経過時間は、他のモードの場合よりも長くなる可能性があります。 1 次データベースとスタンバイ・データベース間のログ・ギャップが相対的に大きくなる場合があるからです。
- **DB2_HADR_SOSNDBUF** および **DB2_HADR_SORCVBUF** レジストリー変数の調整を考慮してください。

  HADR のログ・シッピング・ワークロード、ネットワーク帯域幅、送信遅延は、TCP ソケット・バッファー・サイズを調整するときに検討しなければならない重要な要因です。 2 つのレジストリー変数 **DB2_HADR_SOSNDBUF** と **DB2_HADR_SORCVBUF** により、HADR 接続の TCP ソケットの送信バッファーと受信バッファーのサイズだけを調整することができます。その 2 つの変数

の値の範囲は、1024 から 4294967295 であり、デフォルトでは、オペレーティング・システムのソケット・バッファー・サイズが使用されます (その値は、オペレーティング・システムによって異なります)。**DB2_HADR_SOSNDBUF** と **DB2_HADR_SORCVBUF** の設定値を、最小で 16384 (16 K) にするよう強くお勧めします。オペレーティング・システムによっては、ユーザー指定値が自動的に丸められたり、上限値が設定されていたりすることがあります。

HADR シミュレーター (シミュレートされた HADR ワークロードを生成するコマンド行ツール) を使用して、ネットワーク・パフォーマンスを測定したり、各種ネットワーク・チューニング・オプションを試してみたりすることができます。このシミュレーターは、http://www.ibm.com/developerworks/wikis/display/data/HADR_sim からダウンロードできます。

ネットワーク輻輳 (ふくそう)

1 次でのログの書き込みごとに、同じログ・ページがスタンバイにも送信されます。各回の書き込み操作をフラッシュと言います。フラッシュのサイズは、1 次データベースのログ・バッファー・サイズまでに制限されます (データベース構成パラメーター **logbufsz** で制御)。各回のフラッシュの正確なサイズは非決定的です。ログ・バッファーが大きくなればフラッシュ・サイズも大きくなるとは限りません。

スタンバイ・データベースでログ・ページの再生に時間がかかりすぎる場合、ログ受信バッファーがいっぱいで、バッファーがこれ以上ログ・ページを受け取れなくなっている可能性があります。SYNC および NEARSYNC モードの場合、1 次データベースがもう一度ログ・バッファーをフラッシュすると、データのバッファー先がネットワークのパイプライン (1 次マシン、ネットワーク、およびスタンバイ・データベースで構成される) になる恐れがあります。スタンバイ・データベースにはデータを受け取るための空きバッファーがないので確認通知ができず、1 次データベースはスタンバイ・データベースの確認通知を待機している間、ブロックされることになります。

ASYNC モードの場合、1 次データベースはログ・ページの送信を継続し、やがてパイプラインがいっぱいになって、それ以上ログ・ページを送信できなくなります。この状態を*輻輳* と言います。輻輳は、**hadr_connect_status** モニター・エレメントにより報告されます。SYNC および NEARSYNC モードの場合、通常、パイプラインが単一のフラッシュを吸収できるので輻輳は発生しません。しかし 1 次データベースは、フラッシュ操作に関するスタンバイ・データベースからの確認通知を待機して、ブロックされたままとなります。

輻輳は、再生に長い時間のかかるログ・レコード (データベースまたは表の再編成のログ・レコードなど) をスタンバイ・データベースが再生しているときにも発生することがあります。

SUPERASYNC モードでは、1 次データベースのトランザクション・コミット操作は、HADR ネットワークやスタンバイ HADR サーバーが相対的に低速度であることの影響を受けないので、1 次データベースとスタンバイ・データベース間のログ・ギャップが大きくなり続ける可能性があります。ログ・ギャップは、1 次システムで実際に災害が発生した場合に失われる可能性のあるトランザクション数の間接的指標であるため、モニターすることが重要です。災害時リカバリー・シナリオ

では、ログ・ギャップの間にコミットされたトランザクションは、スタンバイ・データベースでは使用できません。そのために、**hadr_log_gap** モニター・エレメントを使用してログ・ギャップをモニターします。許容できるログ・ギャップではないことが分かったら、ネットワークの中断やスタンバイ HADR サーバーの相対速度を調べ、ログ・ギャップを小さくするための修正措置を講じてください。

ネットワーク輻輳を最小化するための推奨事項:

- スタンバイ・データベースは、ログに記録されたデータベース操作の再生を、1 次側でログが生成されるのと同じ程度の速さで行えるだけの強力なものでなければなりません。1 次とスタンバイで同一のハードウェアを使用することをお勧めします。
- スタンバイ・データベースのログ受信バッファーのサイズを調整することを考慮してください (**DB2_HADR_BUF_SIZE** レジストリー変数を使用)。

  バッファーを大きくすると、輻輳の原因がすべて取り除かれるわけではありませんが、輻輳を減らすのに役立ちます。デフォルトでは、スタンバイ・データベースのログ受信バッファーのサイズは、1 次データベースのログ書き込みバッファーのサイズの 2 倍です。データベース構成パラメーター **logbufsz** は、1 次データベース・ログ書き込みバッファーのサイズを指定します。

  -hadr オプションを指定した **db2pd** コマンドを使用することにより、スタンバイのログ受信バッファーが不十分かどうかを判別できます。StandByRcvBufUsed の値 (スタンバイのログ受信バッファーの使用パーセンテージを示す) が 100 に近ければ、**DB2_HADR_BUF_SIZE** を大きくする必要があります。
- **DB2_HADR_PEER_WAIT_LIMIT** レジストリー変数の設定に注意を払ってください。このレジストリー変数により、スタンバイ・データベースが低速だったりブロックされたりすることによる 1 次データベース・ロギングのブロックを防ぐことができます。

  DB2_HADR_PEER_WAIT_LIMIT レジストリー変数が設定されていると、スタンバイ・データベースへのログ・レプリケーションのために HADR 1 次データベースでのロギングが指定秒数の間ブロックされた場合、その 1 次データベースのピア状態が解除されます。この限度に達すると、1 次データベースはスタンバイ・データベースへの接続を切断します。ピア・ウィンドウが使用不可である場合、1 次データベースは切断済み状態になり、ロギングが再開します。ピア・ウィンドウが使用できる場合、1 次データベースは切断ピア状態になり、その状態では、ロギングはブロックされたままになります。1 次データベースは、再接続またはピア・ウィンドウの終了時に、切断ピア状態のままになります。1 次データベースが切断済みピア状態ではなくなると、ロギングが再開します。

  ピア状態から解除される時のピア・ウィンドウの遷移を考慮すると、すべての事例で安全にテークオーバーが行われるようにピア・ウィンドウのセマンティクスが守られます。遷移の際に 1 次データベースに障害が発生した場合でも、通常のピア・ウィンドウ保護が適用されます (それがまだ切断済みピア状態である限り、スタンバイ・データベースから安全にテークオーバーされます)。

- ほとんどのシステムでは、ロギングの能力が限界まで活用されることはありません。SYNC モードでさえ、1 次データベース上でスローダウンに気付くことはないかもしれません。例えば、HADR を使用可能にした場合のロギングの限界が 1 秒あたり 40 MB であるのに対し、HADR を使用可能にする前に 1 秒あたり 30 MB でシステムが稼働していたに過ぎなかった場合、全体的なシステム・パフォーマンスの違いに気付くことはないかもしれません。
- キャッチアップ処理の速度を速めるには、共有ログ・アーカイブ装置を使用できます。しかし、その共有装置が、磁気テープ・ドライブのようなシリアル装置である場合、読み取り操作と書き込み操作が混在するため、 1 次データベースとスタンバイ・データベースの両方で性能低下が生じる可能性があります。
- スタンバイ・データベースの読み取りフィーチャーを使用する場合、この追加処理を実行するためのリソースが、スタンバイ・データベースになければなりません。
- スタンバイ・データベースの読み取りフィーチャーを使用する場合、バッファー・プールは 1 次データベース上で構成します。そして、その情報はログを介してスタンバイ・データベースに送られます。
- スタンバイ・データベースの読み取りフィーチャーを使用する場合、スタンバイ・データベースの **pckcachesz**、**catalogcache_sz**、**applheapsz**、および **sortheap** 構成パラメーターをチューニングします。

HADR パフォーマンス・チューニングに関する最新の更新情報については、http://www.ibm.com/developerworks/wikis/display/data/HADR_tune の Web サイトを参照してください。

クラスター・マネージャーおよび高可用性災害時リカバリー (HADR)

DB2 高可用性災害時リカバリー (HADR) データベースをクラスターのノードにインプリメントし、クラスター・マネージャーを使用することによりデータベース・ソリューションの可用性を改善することができます。 1 次データベースとスタンバイ・データベースの両方を同じクラスター・マネージャーで管理することもできますし、それぞれ異なるクラスター・マネージャーで管理することもできます。

1 次データベースとスタンバイ・データベースが同じクラスター・マネージャーによって処理される HADR ペアを設定します。

この構成は、1 次データベースとスタンバイ・データベースが同じサイトに存在し、できる限り最速のフェイルオーバーが必要とされる環境に最適です。このような環境では、クラッシュ・リカバリーや別のリカバリー方式を使用するよりも、HADR を使用して DBMS 可用性を保守する方が利点があります。

クラスター・マネージャーを使用すると、問題をすぐに検出し、テークオーバー操作を開始することができます。 HADR では、DBMS に個別のストレージが必要であるため、クラスター・マネージャーは、個別のボリューム制御で構成する必要があります。この構成により、クラスター・マネージャーは、スタンバイ・システムで DBMS を使用する前に、ボリュームでフェイルオーバーが発生することを待たずに済みます。自動クライアント・リルート・フィーチャーを使用して、クライア

ント・アプリケーションを新しい 1 次データベースへリダイレクトすることができます。

1 次データベースとスタンバイ・データベースが同じクラスター・マネージャーによって処理されない HADR ペアを設定します。

この構成は、1 次データベースとスタンバイ・データベースが別々のサイトに存在し、完全なサイト障害時の災害時リカバリーに高可用性が求められる環境に最適です。この構成をインプリメントする方法はいくつかあります。 HADR 1 次データベースおよびスタンバイ・データベースがクラスターの一部であるときには、 2 つのフェイルオーバー・シナリオが考えられます。

- 部分サイト障害が発生する場合で、DBMS がフェイルオーバーできるノードがまだ使用可能な場合、クラスターのフェイルオーバーを選択できます。この場合、IP アドレスとボリュームのフェイルオーバーは、クラスター・マネージャーを使用して実行されます。 HADR は影響を受けません。
- 1 次データベースが存在するサイトで完全なサイト障害が発生する場合、 HADR を使用し、テークオーバー操作を開始することで、DBMS 可用性を維持できます。スタンバイ・データベースが存在するサイトで完全なサイト障害が発生する場合、サイトを修復するか、スタンバイ・データベースを別のサイトへ移動できます。

スタンバイ・データベースの初期化

データベース・ソリューションの可用性を高めるストラテジーの 1 つは、ユーザー・アプリケーションの要求に応答する 1 次データベース、および 1 次データベースに障害が生じた場合に 1 次データベースに代ってデータベース操作をテークオーバーできる 2 次またはスタンバイ・データベースを保守することです。スタンバイ・データベースを初期化するためには、 1 次データベースをスタンバイ・データベースにコピーすることが必要です。

手順

スタンバイ・データベースを初期化するには、いくつかの方法があります。例:

- ディスク・ミラーリングを使用して 1 次データベースをコピーしてから、 DB2 データベースの中断された I/O サポートを使用してミラーを分割することにより 2 番目のデータベースを作成します。
- 1 次データベースのバックアップ・イメージを作成して、そのイメージをスタンバイ・データベースにリカバリーします。
- SQL レプリケーションを使用して 1 次データベースからデータをキャプチャーして、そのデータをスタンバイ・データベースに適用します。

次のタスク

スタンバイ・データベースを初期化した後に、 1 次データベースとスタンバイ・データベースとが同期するようにデータベース・ソリューションを構成して、 1 次データベースに障害が生じた場合にスタンバイ・データベースが 1 次データベースをテークオーバーできるようにします。

スプリット・ミラーをスタンバイ・データベースとして使用する

DB2 pureScale 環境以外でスタンバイ・データベースとして使用するために、データベースのスプリット・ミラーを作成する場合は、以下の手順に従ってください。1 次データベースで障害が発生してアクセスできなくなった場合、スタンバイ・データベースを使用して、1 次データベースを引き継ぐことができます。

このタスクについて

1 次データベースがログをアーカイブするように構成されている場合、スタンバイ・データベースも、同じログ・アーカイブ構成を共有します。ログのアーカイブ先がスタンバイ・データベースからアクセス可能であれば、スタンバイ・データベースは、ロールフォワード操作時に、そこからログ・ファイルを自動的にリトリーブします。ただし、データベースがロールフォワード・ペンディング状態ではなくなると、スタンバイ・データベースは、1 次データベースが使用しているのと同じ場所に、ログ・ファイルをアーカイブしようとします。スタンバイ・データベースは、最初は 1 次データベースと異なるログ・チェーンを使用しますが、 1 次データベースが、最終的にスタンバイ・データベースと同じログ・チェーン値を使用するようになる場合があります。このため、スタンバイ・データベースがアーカイブしたログ・ファイルの上に、 1 次データベースがログ・ファイルをアーカイブする (またはその逆) 可能性があります。これは、両方のデータベースのリカバリー可能性に影響を与えることがあります。リカバリー可能性に関するこの問題を回避するには、スタンバイ・データベースのログのアーカイブ先を、 1 次データベースとは別の場所に変更する必要があります。

手順

スプリット・ミラーをスタンバイ・データベースとして使用するには、以下を実行します。

1. 次のコマンドを使用して、1 次データベース上で入出力書き込み操作をサスペンドします。

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

   注: データベースが中断状態のときは、他のユーティリティーやツールを実行しないようにしてください。データベースのコピーを作成するだけにしてください。**SET WRITE** SUSPEND を発行する前に、オプションで FLUSH BUFFERPOOLS ALL ステートメントを使用して、スタンバイ・データベースのリカバリー時間を最小化することができます。
2. 適切なオペレーティング・システム・レベルおよびストレージ・レベルのコマンドを使用して、 1 次データベースから 1 つまたは複数のスプリット・ミラーを作成します。

   注: ボリューム・ディレクトリーを含め、データベース・ディレクトリー全体をコピーするようにしてください。さらに、データベース・ディレクトリー外にある、ログ・ディレクトリーおよびコンテナー・ディレクトリーもコピーする必要があります。この情報を収集するには、DBPATHS 管理ビューを参照してください。このビューは、分割する必要のあるデータベースのすべてのファイルとディレクトリーを表示します。
3. 次のコマンドを使用して、1 次データベース上で入出力書き込み操作を再開します。

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

4. 2 次システムのミラー・データベースをカタログします。

   注: デフォルトでは、ミラー・データベースは、 1 次データベースと同じシステムに存在できません。これは、同じディレクトリー構造を持ち、1 次データベースと同じインスタンス名を使用する、 2 次システム上に置く必要があります。ミラー・データベースを、1 次データベースと同じシステムに置かなければならない場合、 **db2relocatedb** ユーティリティーか、 **db2inidb** コマンドの RELOCATE USING オプションを使用して、このことを実現できます。

5. 次のコマンドを使用して、2 次システムでデータベース・インスタンスを開始します。

   ```
   db2start
   ```

6. 次のコマンドを使用することにより、2 次システムでミラー・データベースを初期化して、ロールフォワード・ペンディング状態にします。

   ```
   db2inidb <database_alias> as standby
   ```

   必要であれば、**db2inidb** コマンドの RELOCATE USING オプションを指定して、スタンバイ・データベースを再配置します。

   ```
   db2inidb <database_alias> as standby relocate using relocatedbcfg.txt
   ```

   ここで、relocatedbcfg.txt ファイルには、データベースを再配置するのに必要な情報が示されています。

   注: DMS 表スペース (データベース管理スペース) または自動ストレージ表スペースがある場合は、スプリット・ミラーを使用してデータベースのフル・バックアップを取ることができます。スプリット・ミラーを使用してバックアップを取ると、稼働データベースのバックアップを取ることによるオーバーヘッドが低減されます。このようなバックアップは、オンライン・バックアップと見なされるものであり、未完了トランザクションを含んでいる可能性がありますが、このバックアップ内に、スタンバイ・データベースのログ・ファイルを含めることはできません。このようなバックアップをリストアした場合は、少なくともそのバックアップの最後までロールフォワードしてから、**ROLLFORWARD** コマンドに STOP オプションを指定して発行する必要があります。このバックアップにはログ・ファイルが含まれないため、 **SET WRITE** SUSPEND コマンドの発行時に使用されていた 1 次データベースのログ・ファイルを用意する必要があります。そうしないと、ロールフォワード操作は、このバックアップの最後に到達できません。

7. スタンバイ・データベースのログ・アーカイブ・パラメーターを構成するか、または、ログをスタンバイ・データベースにシップすることによって、 1 次データベースのアーカイブ・ログ・ファイルを、スタンバイ・データベースで使用できるようにします。
8. データベースを、to end of logs オプションで、あるいは、to point-in-time にてポイント・イン・タイム指定で、ロールフォワードします。
9. ログの最後、あるいは、指定されたポイント・イン・タイムに達するまで、ログの取得およびロールフォワードを実行し続けます。
10. スタンバイ・データベースをオンラインにするには、**ROLLFORWARD** コマンドを STOP オプションを指定して実行します。

注:

- 1 次データベースのログは、ミラー・データベースのロールフォワード・ペンディング状態解除後には、そのミラー・データベースに適用できません。
- 1 次データベースがログをアーカイブするように構成されている場合、スタンバイ・データベースも、同じログ・アーカイブ構成を共有します。ログのアーカイブ先がスタンバイ・データベースからアクセス可能であれば、スタンバイ・データベースは、ロールフォワード実行中に、そこからログ・ファイルを自動的にリトリーブします。ただし、データベースがロールフォワード・ペンディング状態ではなくなると、スタンバイ・データベースは、1 次データベースが使用しているのと同じ場所に、ログ・ファイルをアーカイブしようとします。スタンバイ・データベースは、最初は 1 次データベースと異なるログ・チェーンを使用しますが、 1 次データベースが、最終的にスタンバイ・データベースと同じログ・チェーン値を使用するようになる場合があります。このため、スタンバイ・データベースがアーカイブしたログ・ファイルの上に、 1 次データベースがログ・ファイルをアーカイブする (またはその逆) 可能性があります。これは、両方のデータベースのリカバリー可能性に影響を与えることがあります。この問題を回避するには、スタンバイ・データベースのログのアーカイブ先を、 1 次データベースとは別の場所に変更する必要があります。

DB2 pureScale環境でスプリット・ミラーをスタンバイ・データベースとして使用する

DB2 pureScale環境でスタンバイ・データベースとして使用するために、データベースのスプリット・ミラーを作成するには、以下の手順に従ってください。 1 次データベースで障害が発生してアクセスできなくなった場合、スタンバイ・データベースを使用して、1 次データベースを引き継ぐことができます。

このタスクについて

1 次データベースがログをアーカイブするように構成されている場合、スタンバイ・データベースも、同じログ・アーカイブ構成を共有します。ログのアーカイブ先がスタンバイ・データベースからアクセス可能であれば、スタンバイ・データベースは、ロールフォワード操作時に、そこからログ・ファイルを自動的にリトリーブします。ただし、データベースがロールフォワード・ペンディング状態ではなくなると、スタンバイ・データベースは、1 次データベースが使用しているのと同じ場所に、ログ・ファイルをアーカイブしようとします。スタンバイ・データベースは、最初は 1 次データベースと異なるログ・チェーンを使用しますが、 1 次データベースが、最終的にスタンバイ・データベースと同じログ・チェーン値を使用するようになる場合があります。このため、スタンバイ・データベースがアーカイブしたログ・ファイルの上に、 1 次データベースがログ・ファイルをアーカイブする (またはその逆) 可能性があります。これは、両方のデータベースのリカバリー可能性に影響を与えることがあります。リカバリー可能性に関するこの問題を回避するには、スタンバイ・データベースのログのアーカイブ先を、 1 次データベースとは別の場所に変更する必要があります。

手順

スプリット・ミラーをスタンバイ・データベースとして使用するには、以下を実行します。

1. 1 次クラスターの設定を取り出してインポートすることにより、2 次クラスター上に General Parallel File System (GPFS™) を構成します。1 次クラスター上で、以下の GPFSコマンドを実行します。

   ```
   mmfsctl <filesystem> syncFSconfig -n <remotenodefile>
   ```

   ここで、*<remotenodefile>* は、2 次クラスター内のホストのリストです。
2. 次のコマンドを使用して、クラスター・マネージャーのドメインをリストします。

   ```
   db2cluster -cm -list -domain
   ```
3. 次のコマンドを使用して、各ホスト上のクラスター・マネージャーを停止します。

   ```
   db2cluster -cm -stop -host <host> -force
   ```

   注: このコマンドを発行するホストを最後にシャットダウンする必要があります。
4. 次のコマンドを使用して、2 次システムで GPFS クラスターを停止します。

   ```
   db2cluster -cfs -stop -all
   ```
5. 次のコマンドを使用して、1 次データベース上で入出力書き込み操作をサスペンドします。

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

   注: データベースが中断状態のときは、他のユーティリティーやツールを実行しないようにしてください。データベースのコピーを作成するだけにしてください。オプションで、**SET WRITE SUSPEND** を発行する前にすべてのバッファー・プールをフラッシュして、リカバリー・ウィンドウを最小化することができます。それには FLUSH BUFFERPOOLS ALL ステートメントを使用します。
6. 次のコマンドを使用して、サスペンドしてコピーする必要のあるファイル・システムを判断します。

   ```
   db2cluster -cfs -list -filesystem
   ```
7. 次のコマンドを使用して、データまたはログ・データを含む各 GPFS ファイル・システムをサスペンドします。

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

   ここで *<filesystem>* は、データまたはログ・データを含むファイル・システムを表しています。

   注: GPFS ファイル・システムがサスペンドされている間は、読み取りおよび書き込み操作の両方がブロックされます。読み取り操作がブロックされる時間を最小限に抑えるには、この間にスプリット・ミラー操作以外は実行すべきではありません。
8. 適切なオペレーティング・システム・レベルおよびストレージ・レベルのコマンドを使用して、 1 次データベースから 1 つまたは複数のスプリット・ミラーを作成します。

   注: ボリューム・ディレクトリーを含め、データベース・ディレクトリー全体をコピーするようにしてください。さらに、データベース・ディレクトリー外

にあるログ・ディレクトリー (すべてのログ・ストリーム・サブディレクトリーも含む)、およびコンテナー・ディレクトリーもコピーする必要があります。

9. サスペンドした各 GPFS ファイル・システムに対して次のコマンドを使用して、ファイル・システムをサスペンド状態から再開します。

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

   ここで *filesystem* は、データまたはログ・データを含む、サスペンドされたファイル・システムを表しています。

10. 次のコマンドを使用して、1 次データベース上で入出力書き込み操作を再開します。

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

11. 次のコマンドを使用して、2 次システムで GPFS クラスターを開始します。

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

12. 次のコマンドを使用して、クラスター・マネージャーを開始します。

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

13. 2 次システムのミラー・データベースをカタログします。

    注: デフォルトでは、ミラー・データベースは、 1 次データベースと同じシステムに存在できません。これは、同じディレクトリー構造を持ち、1 次データベースと同じインスタンス名を使用する、 2 次システム上に置く必要があります。ミラー・データベースを、1 次データベースと同じシステムに置かなければならない場合、 **db2relocatedb** ユーティリティーか、 **db2inidb** コマンドの **RELOCATE USING** オプションを使用して、このことを実現できます。

14. 次のコマンドを使用して、2 次システムでデータベース・インスタンスを開始します。

    ```
    db2start
    ```

15. 2 次システムでデータベースを初期化して、ロールフォワード・ペンディング状態にします。

    ```
    db2inidb <database_alias> as standby
    ```

    必要であれば、**db2inidb** コマンドの RELOCATE USING オプションを指定して、データベースを再配置します。

    ```
    db2inidb database_alias as standby relocate using relocatedbcfg.txt
    ```

    ここで、relocatedbcfg.txt には、データベースを再配置するのに必要な情報が示されています。

    注: DMS 表スペース (データベース管理スペース) または自動ストレージ表スペースがある場合は、スプリット・ミラーを使用してデータベースの全バックアップを取ることができます。スプリット・ミラーを使用してバックアップを取ると、稼働データベースのバックアップを取ることによるオーバーヘッドが低減されます。このようなバックアップは、オンライン・バックアップと見なされるものであり、未完了トランザクションを含んでいる可能性がありますが、このバックアップ内に、スタンバイ・データベースのログ・ファイルを含めることはできません。このようなバックアップをリストアする場合は、少なくともそのバックアップの最後までロールフォワードしてから、 **ROLLFORWARD** STOP コマンドを実行する必要があります。このバックアップにはログ・ファイ

ルが含まれないため、 **SET WRITE** SUSPEND コマンドの発行時に使用されていた 1 次データベースのログ・ファイルを用意する必要があります。そうしないと、ロールフォワード操作は、このバックアップの最後に到達できません。

16. スタンバイ・データベースのログ・アーカイブ・パラメーターを構成するか、または、ログをスタンバイ・データベースにシップすることによって、 1 次データベースのアーカイブ・ログ・ファイルを、スタンバイ・データベースで使用できるようにします。
17. データベースを、to end of logs オプションで、あるいは、to point-in-time にてポイント・イン・タイム指定で、ロールフォワードします。

    注: ロールフォワード操作の実行中に、 SQL1273 エラーが発生することがあります。データベースのスプリット時に一部のログ・ファイルが 1 次システムからコピーされなかった場合や、または、あるメンバーが他のメンバーよりも速くログ・ファイルを生成している場合に、このエラーが予期されます。ログ・ファイルの内容が、使用できない他のメンバーのログ・ファイルの内容に依存しているために、データの整合性を保持するにはロールフォワード操作を停止しなければならないという状況で、 SQL1273 は生成されます。スタンバイ・データベースが、 1 次データベースのアーカイブ・ログ・ファイルをリトリーブするように構成されている場合は、 1 次システムによって必要なログ・ファイルがアーカイブされるまで待機することもできますが、 1 次システムで ARCHIVE LOG コマンドを使用して、ログ・ファイルを強制的にアーカイブさせることもできます。そうしない場合は、必要なログ・ファイルをスタンバイ・データベースにシップする必要があります。必要なログ・ファイルがスタンバイ・データベースで使用できるようになると、ロールフォワード操作でさらに先のログを読み取ることができます。ただし、一部のメンバーが、まだ他のメンバーより速くログ・ファイルを生成している場合は、再度 SQL1273 が発生する可能性があります。詳しくは、インフォメーション・センターのトピック『DB2 pureScale 環境でのバックアップおよびリストア操作』の『 DB2 pureScale 環境でのログ・シッピングによる災害復旧および高可用性』セクションを参照してください。
18. 必要があれば、スタンバイ・データベースに新規ログ・ファイルをシップしながら、ログの最後まで、あるいは、スタンバイ・データベースに必要なポイント・イン・タイムに達するまで、ログを使用してロールフォワード操作を実行し続けます。
19. スタンバイ・データベースをオンラインにするには、**ROLLFORWARD DATABASE** コマンドを STOP オプションを指定して実行します。

    注:

    - 1 次データベースのログは、ミラー・データベースのロールフォワード・ペンディング状態解除後には、そのミラー・データベースに適用できません。
    - 1 次データベースがログをアーカイブするように構成されている場合、スタンバイ・データベースも、同じログ・アーカイブ構成を共有します。ログのアーカイブ先がスタンバイ・データベースからアクセス可能であれば、スタンバイ・データベースは、ロールフォワード実行中に、そこからログ・ファイルを自動的にリトリーブします。ただし、データベースがロールフォワード・ペンディング状態ではなくなると、スタンバイ・データベースは、1 次データベースが使用しているのと同じ場所に、ログ・ファイルをアーカイブ

しようとします。スタンバイ・データベースは、最初は 1 次データベースと異なるログ・チェーンを使用しますが、 1 次データベースが、最終的にスタンバイ・データベースと同じログ・チェーン値を使用するようになる場合があります。このため、スタンバイ・データベースがアーカイブしたログ・ファイルの上に、 1 次データベースがログ・ファイルをアーカイブする (またはその逆) 可能性があります。これは、両方のデータベースのリカバリー可能性に影響を与えることがあります。この問題を回避するには、スタンバイ・データベースのログのアーカイブ先を、 1 次データベースとは別の場所に変更する必要があります。

高可用性災害時リカバリー (HADR) 同期モード

HADR 同期モードは、DB2 高可用性災害時リカバリー (HADR) データベース・ソリューションによるトランザクションの損失に対する保護の度合いを決定します。同期モードによって、スタンバイ・データベースへのログオン状態に基づいて、 1 次データベース・サーバーがいつトランザクションを完了したとみなすかが決まります。

同期モードの構成パラメーター値を厳密にするほど、データベース・ソリューションによるトランザクション・データの損失に対する保護が強化されますが、トランザクション処理のパフォーマンスは低くなります。トランザクション・データの損失に対する保護の必要性とパフォーマンスの必要性との間でバランスを取る必要があります。

次の 図 3 は、使用可能な DB2 HADR 同期モードを示しています。また、トランザクションがコミットされたといつ見なされるかについて、選択される同期モードに基づいて示しています。

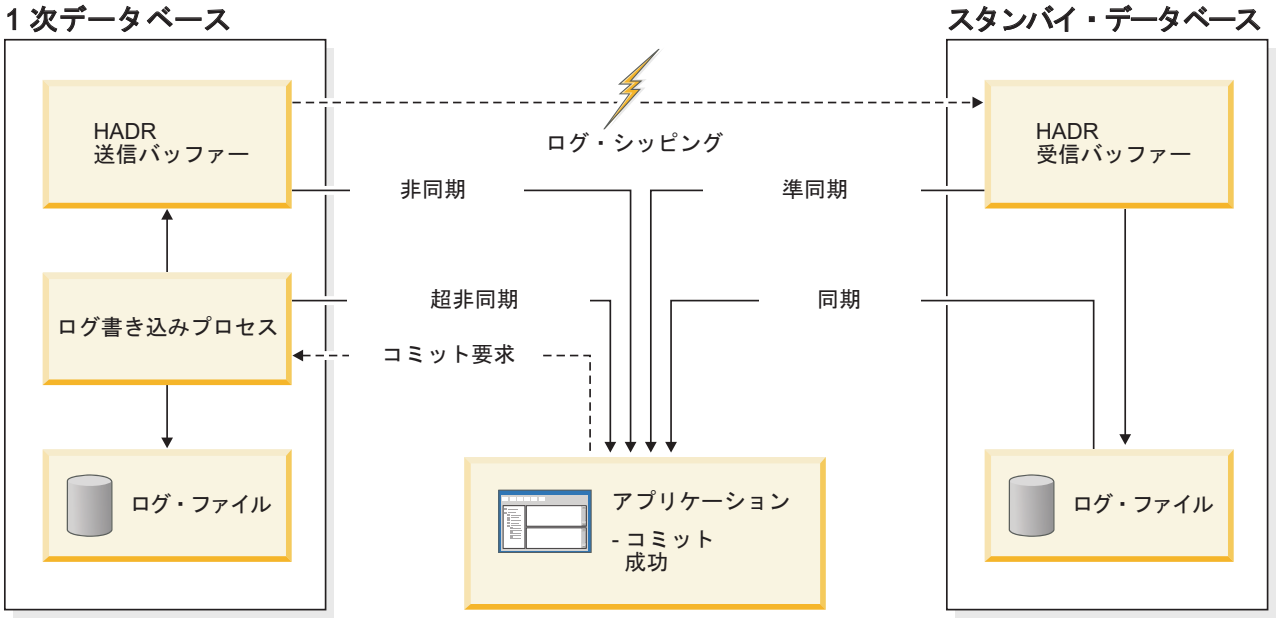


図 3. 高可用性災害時リカバリー (HADR) の同期モード

複数スタンバイ・モードでは、**hadr_syncmode** の設定が 1 次データベースとスタンバイ・データベースで同じである必要はありません。スタンバイで指定される

hadr_syncmode の設定はすべて、*その構成済み同期モード* と見なされます。この設定は、スタンバイが 1 次になる場合にのみ関係します。代わりに、スタンバイに*有効同期モード* が割り当てられます。どの補助スタンバイでも、有効同期モードは常に SUPERASYNC になります。プリンシパル・スタンバイの場合、有効同期モードは、1 次の **hadr_syncmode** の設定です。スタンバイの有効同期モードは、いずれかのモニター・インターフェースに表示される値です。

同期モードを設定するには、**hadr_syncmode** データベース構成パラメーターを使用します。以下の値が有効です。

SYNC (同期)

このモードは、トランザクション消失に対する最高度の保護を実現しますが、この使用のために、4 つのモードの中でもトランザクション応答時間は最長になります。

このモードでは、ログ書き込みは、ログが 1 次データベースのログ・ファイルに書き込まれ、ログがスタンバイ・データベース上のログ・ファイルにも書き込まれたことの確認通知をスタンバイ・データベースから 1 次データベースが受信した場合にのみ、成功したものと考えられます。ログ・データは、確実に両方のサイトに保管されます。

スタンバイ・データベースがログ・レコードを再生する前に破損する場合は、次回に開始したときに、ローカル・ログ・ファイルから取り出して再生することができます。 1 次データベースに障害が発生する場合の、スタンバイ・データベースに対するフェイルオーバーにおいて、1 次データベースでコミットされたトランザクションは、スタンバイ・データベースでもコミットされていることを保証します。フェイルオーバー操作の後で、クライアントが新しい 1 次データベースに再接続する場合、新しい 1 次データベースでコミット済みと報告されているトランザクションが、元の 1 次データベースではアプリケーションにコミット済みと報告されていない場合があります。このことが生じるのは、1 次データベースがスタンバイ・データベースからの確認通知メッセージを処理する前に障害が発生する場合です。クライアント・アプリケーションでは、データベースを照会することで、そのようなトランザクションが存在するかどうかを判別する必要があります。

1 次データベースがスタンバイ・データベースへの接続を失う場合、次の動作は **hadr_peer_window** データベース構成パラメーターの構成によって異なります。**hadr_peer_window** がゼロ以外の時間値に設定されている場合、スタンバイ・データベースとの接続を失うと、1 次データベースは切断済みピア状態に移り、トランザクションをコミットする前にスタンバイ・データベースからの確認通知を待機し続けます。 **hadr_peer_window** データベース構成パラメーターがゼロに設定されている場合、1 次データベースとスタンバイ・データベースはピア状態であるとは見なされなくなり、トランザクションはスタンバイ・データベースからの確認通知を待機して動作が止まることはありません。データベースがピア状態でないかまたは切断済みピア状態でないときに、フェイルオーバー操作が実行されると、1 次データベースでコミットされたすべてのトランザクションがスタンバイ・データベースでコミットされている保証はありません。

1 次データベースがピア状態または切断済みピア状態のときに、そのデータベースに障害が発生すると、フェイルオーバー操作の後で、HADR ペアにスタンバイ・データベースとして再度加わることができます。ログがスタン

バイ・データベース上のログ・ファイルにも書き込まれたことの確認通知をスタンバイ・データベースから 1 次データベースが受信するまで、トランザクションはコミット済みとは見なされないため、1 次データベースのログ・シーケンスは、スタンバイ・データベースのログ・シーケンスと同じになります。元の 1 次データベース (現在のスタンバイ・データベース) は、フェイルオーバー操作以降に、新しい 1 次データベースで生成される新しいログ・レコードを再生することで、最新の内容に保つ必要があります。

1 次データベースの障害時にピア状態ではない場合、そのログ・シーケンスは、スタンバイ・データベースのログ・シーケンスとは違う可能性があります。フェイルオーバー操作を実行する必要がある場合、1 次データベースとスタンバイ・データベースのログ・シーケンスは異なる場合があります。これは、スタンバイ・データベースはフェイルオーバー後に独自のログ・シーケンスを開始するためです。一部の操作 (例えば、表のドロップ) は取り消すことができないため、1 次データベースを、新しいログ・シーケンスが作成された時点まで復帰させることは不可能です。ログ・シーケンスが異なり、元の 1 次データベースに対して **AS STANDBY** パラメーターを指定した **START HADR** コマンドを発行した場合、コマンドが成功したことを示すメッセージを受け取ります。ただし、このメッセージは再統合が試行される前に発行されます。再統合が失敗する場合、1 次データベースおよびスタンバイ・データベース上の両方の管理ログおよび診断ログに対して、ペアとなる妥当性検査メッセージが発行されます。再統合されたスタンバイ・データベースはスタンバイ・データベースのままですが、1 次データベースはペアの妥当性検査時にスタンバイ・データベースを拒否するので、スタンバイ・データベースはシャットダウンします。元の 1 次データベースが HADR ペアに正常に再度加わることができた場合、 **BY FORCE** パラメーターを指定しない **TAKEOVER HADR** コマンドを出すことにより、データベースのフェイルバックを実現できます。元の 1 次データベースが HADR ペアに再度加わることができない場合、新しい 1 次データベースのバックアップ・イメージをリストアすることで、スタンバイ・データベースとして再初期設定することができます。

NEARSYNC (準同期)

このモードは同期モードよりもトランザクション応答時間が短いですが、トランザクション消失に対する保護が若干劣ります。

このモードでは、ログ書き込みは、ログ・レコードが 1 次データベースのログ・ファイルに書き込まれ、ログがスタンバイ・システム上のメイン・メモリーにも書き込まれたことの確認通知をスタンバイ・システムから 1 次データベースが受信した場合にのみ、成功したものと考えられます。データの消失は、両方のサイトに同時に障害が発生し、ターゲット・サイトが、受信したすべてのログ・データを不揮発性ストレージに転送していない場合にのみ生じます。

スタンバイ・データベースがログ・レコードをメモリーからディスクへコピーする前に破損する場合、スタンバイ・データベースのログ・レコードが失われます。通常は、スタンバイ・データベースは、再始動時に消失したログ・レコードを 1 次データベースから入手できます。しかし、1 次データベースまたはネットワークでの障害によって、検索が不可能になりフェイル

オーバーが必要になる場合、ログ・レコードはスタンバイ・データベースに送られず、そのログ・レコードに関連したトランザクションはスタンバイ・データベースに送られません。

トランザクションが失われる場合、新しい 1 次データベースは、フェイルオーバー操作後の元の 1 次データベースと同じではありません。クライアント・アプリケーションは、これらのトランザクションを再実行して、アプリケーションの状態を最新にする必要があります。

1 次データベースとスタンバイ・データベースがピア状態のときに 1 次データベースに障害が発生する場合、完全なリストア操作を使用して再初期設定しないと、元の 1 次データベースはスタンバイ・データベースとして HADR ペアに再度加わることができない可能性があります。フェイルオーバーに、(1 次データベースとスタンバイ・データベースの両方に障害が発生したために) 消失したログ・レコードが関係する場合、 1 次データベースおよびスタンバイ・データベース両方のログ・シーケンスは異なり、先にリストア操作を実行せずに、元の 1 次データベースをスタンバイ・データベースとして再始動しようとすると失敗します。元の 1 次データベースが HADR ペアに正常に再度加わることができた場合、 **BY FORCE** パラメーターを指定しない **TAKEOVER HADR** コマンドを出すことにより、データベースのフェイルバックを実現できます。元の 1 次データベースが HADR ペアに再度加わることができない場合、新しい 1 次データベースのバックアップ・イメージをリストアすることで、スタンバイ・データベースとして再初期設定することができます。

ASYNC (非同期)

SYNC および NEARSYNC モードと比較すると、ASYNC モードではトランザクション応答時間が短くなりますが、1 次データベースに障害が発生した場合にトランザクション消失が大きくなる可能性があります。

ASYNC モードでは、ログ書き込みは、ログ・レコードが 1 次データベース上のログ・ファイルに書き込まれ、 1 次システムのホスト・マシンの TCP レイヤーに送信される場合にのみ、成功したものと考えられます。 1 次システムはスタンバイ・システムからの確認通知を待たないため、トランザクションはスタンバイ・データベースへの送信途上であっても、コミット済みと見なされる可能性があります。

1 次データベース・ホスト・マシン、ネットワーク、またはスタンバイ・データベース上の障害により、転送中のログ・レコードが消失してしまうことがあります。 1 次データベースが使用可能な場合、ペアのデータベースが接続を再確立するときに、消失しているログ・レコードをスタンバイ・データベースへ再送できます。しかし、ログ・レコードが消失しているときにフェイルオーバー操作が必要な場合は、それらのログ・レコードはスタンバイ・データベースへ到達せず、関連するトランザクションはフェイルオーバーで消失します。

トランザクションが失われる場合、新しい 1 次データベースは、フェイルオーバー操作後の元の 1 次データベースとまったく同じではありません。クライアント・アプリケーションは、これらのトランザクションを再実行して、アプリケーションの状態を最新にする必要があります。

1 次データベースとスタンバイ・データベースがピア状態のときに 1 次データベースに障害が発生する場合、完全なリストア操作を使用して再初期設

定しないと、元の 1 次データベースはスタンバイ・データベースとして HADR ペアに再度加わることができない可能性があります。フェイルオーバーに、消失したログ・レコードが関係する場合、1 次データベースおよびスタンバイ・データベース両方のログ・シーケンスは異なり、元の 1 次データベースをスタンバイ・データベースとして再始動しようとすると失敗します。非同期モードでフェイルオーバーが生じる場合、ログ・レコードが消失する可能性が高くなるため、 1 次データベースが HADR ペアに再度加わることができなくなる可能性も高くなります。元の 1 次データベースが HADR ペアに正常に再度加わることができた場合、 **BY FORCE** パラメーターを指定しない **TAKEOVER HADR** コマンドを出すことにより、データベースのフェイルバックを実現できます。元の 1 次データベースが HADR ペアに再度加わることができない場合、新しい 1 次データベースのバックアップ・イメージをリストアすることで、スタンバイ・データベースとして再初期設定することができます。

SUPERASYNC (超非同期)

このモードでは、トランザクション応答時間が最も短くなりますが、1 次システムに障害が発生した場合のトランザクション消失の可能性も最も高くなります。このモードは、ネットワークの中断や輻輳のためにトランザクションがブロックされたり応答時間が長くなったりすることがないようにする場合に有用です。

このモードでは、HADR ペアがピア状態になったり切断済みピア状態になったりすることはありません。 1 次データベースのログ・ファイルにログ・レコードが書き込まれた時点で、ログ書き込みは成功と見なされます。1 次データベースはスタンバイ・データベースからの確認通知を待たないため、トランザクションはそのレプリケーションの状態に関係なく、コミットされたと見なされます。

1 次データベース・ホスト・マシン、ネットワーク、またはスタンバイ・データベース上の障害により、転送中のログ・レコードが消失してしまうことがあります。 1 次データベースが使用可能な場合、ペアのデータベースが接続を再確立するときに、消失しているログ・レコードをスタンバイ・データベースへ再送できます。しかし、ログ・レコードが消失しているときにフェイルオーバー操作が必要な場合は、それらのログ・レコードはスタンバイ・データベースへ到達せず、関連するトランザクションはフェイルオーバーで消失します。

トランザクションが失われる場合、新しい 1 次データベースは、フェイルオーバー操作後の元の 1 次データベースとまったく同じではありません。クライアント・アプリケーションは、これらのトランザクションを再実行して、アプリケーションの状態を最新にする必要があります。

1 次データベースのトランザクション・コミット操作は、HADR ネットワークやスタンバイ HADR サーバーが相対的に低速度であることの影響を受けないので、1 次データベースとスタンバイ・データベース間のログ・ギャップが大きくなり続ける可能性があります。ログ・ギャップは、1 次システムで実際に災害が発生した場合に失われる可能性のあるトランザクション数の間接的指標であるため、モニターすることが重要です。災害時リカバリー・シナリオでは、ログ・ギャップの間にコミットされたトランザクションは、スタンバイ・データベースでは使用できません。そのために、

hadr_log_gap モニター・エレメントを使用してログ・ギャップをモニターします。許容できるログ・ギャップではないことが分かったら、ネットワークの中断やスタンバイ・データベース・ノードの相対速度を調べ、ログ・ギャップを小さくするための修正措置を講じてください。

1 次データベースに障害が発生した場合、元の 1 次データベースは、完全なリストア操作を使用して再初期設定しないと、スタンバイ・データベースとして HADR ペアに再度加わることができなくなる可能性があります。フェイルオーバーに、消失したログ・レコードが関係する場合、1 次データベースおよびスタンバイ・データベース両方のログ・シーケンスは異なり、元の 1 次データベースをスタンバイ・データベースとして再始動しようとすると失敗します。超非同期モードでフェイルオーバーが行われる場合、ログ・レコードが消失する可能性が高くなるため、1 次データベースが HADR ペアに再度加わることができなくなる可能性も高くなります。元の 1 次データベースが HADR ペアに正常に再度加わることができた場合、**BY FORCE** パラメーターを指定しない **TAKEOVER HADR** コマンドを出すことにより、データベースのフェイルバックを実現できます。元の 1 次データベースが HADR ペアに再度加わることができない場合、新しい 1 次データベースのバックアップ・イメージをリストアすることで、スタンバイ・データベースとして再初期設定することができます。

高可用性災害時リカバリー (HADR) のサポート

DB2 データベースの高可用性災害時リカバリー (HADR) フィーチャーで最大の効果を得るには、高可用性データベース・ソリューションを設計するときにシステム要件およびフィーチャー制限を検討してください。

高可用性災害時リカバリー (HADR) のシステム要件

高可用性災害時リカバリー (HADR) での最良のパフォーマンスを実現するには、システムが、ハードウェア、オペレーティング・システム、および DB2 データベース・システムについての以下の要件を満たすようにします。

推奨: パフォーマンスを良くするため、1 次データベースが存在するシステムと、スタンバイ・データベースが存在するシステムとで、同じハードウェアおよびソフトウェアを使用してください。スタンバイ・データベースが存在するシステムのリソースが、1 次データベースが存在するシステムのリソースよりも少ない場合、スタンバイ・データベースは、1 次データベースによって生成されたトランザクションを処理できない可能性があります。これにより、スタンバイ・データベースが遅れるか、1 次データベースのパフォーマンスが低下する場合があります。フェイルオーバー状態では、新規 1 次データベースには、クライアント・アプリケーションに適切に対応できるだけのリソースが必要です。

スタンバイ・データベースの読み取りを使用可能にして読み取り専用ワークロードの一部をスタンバイ・データベースで実行する場合は、スタンバイ・データベースに十分なリソースがあることを確認してください。トランザクション、セッション、新規スレッド、および、ソートや結合操作を含む照会をサポートするために、アクティブ・スタンバイ・データベースのメモリーおよび TEMPORARY 表スペースに追加の容量が必要です。

ハードウェアおよびオペレーティング・システム要件

推奨: HADR 1 次データベースとスタンバイ・データベースとで、同じホスト・コンピューターを使用してください。つまり、同じベンダーの製品で、同じアーキテクチャーでなければならないということです。

1 次データベースとスタンバイ・データベースのオペレーティング・システムは、パッチも含めて、同じバージョンでなければなりません。アップグレード時には、短期間だけこのルールに違反できますが、細心の注意が必要です。

HADR ホスト・マシン間では、TCP/IP インターフェースが使用できる必要があり、高速大容量のネットワークが推奨されます。

DB2 データベース要件

1 次データベースおよびスタンバイ・データベースのデータベース・システムのバージョンは同一でなければなりません。例えば、両方ともバージョン 8 かバージョン 9 でなければなりません。ローリング更新中には、スタンバイ・データベースのデータベース・システムの修正レベル (例えば、フィックスパック・レベル) は、新規レベルをテストするために、短期間だけ、1 次データベースよりも新しくても構いません。ただし、この構成は長期間維持しないようにしてください。 1 次データベースのデータベース・システムの修正レベルがスタンバイ・データベースよりも新しい場合、1 次データベースおよびスタンバイ・データベースは、互いに接続しません。スタンバイ・データベースの読み取りフィーチャーを使用するには、 1 次データベースおよびスタンバイ・データベースの両方がバージョン 9.7 フィックスパック 1 である必要があります。

1 次データベースとスタンバイ・データベースの両方の DB2 データベース・ソフトウェアは、同じビット・サイズ (32 または 64 ビット) にする必要があります。表スペースとそのコンテナーは、1 次データベースおよびスタンバイ・データベース上で同一でなければなりません。同一でなければならないプロパティーには、表スペース・タイプ (DMS または SMS)、表スペース・サイズ、コンテナー・パス、コンテナー・サイズ、およびコンテナー・ファイル・タイプ (ロー・デバイスまたはファイル・システム) があります。ログ・ファイルに割り振られるスペース量も、1 次データベースとスタンバイ・データベースの両方で同じでなければなりません。

1 次データベースに対して CREATE TABLESPACE、ALTER TABLESPACE、または DROP TABLESPACE などの表スペース・ステートメントを発行すると、そのステートメントはスタンバイ・データベースで再生されます。 1 次データベースに対して表スペース・ステートメントを発行する前に、関係するデバイスを両方のデータベースでセットアップしておく必要があります。

1 次データベース上に表スペースを作成したものの、コンテナーが使用不可であったためにスタンバイ・データベースでのログの適用が失敗した場合、1 次データベースはログ適用の失敗を通知するエラー・メッセージを受け取りません。

ログ適用のエラーを確認するには、新しい表スペースの作成時に、スタンバイ・データベース上の **db2diag** ログ・ファイルと管理通知ログ・ファイルをモニターすることが必要です。

テークオーバー操作が発生する場合、作成した新しい表スペースは新しい 1 次データベース上で使用不可になります。この状態から回復するには、新しい 1 次データベース上の表スペースをバックアップ・イメージからリストアします。

以下の例では、データベース MY_DATABASE を新しい 1 次データベースとして使用する前に、このデータベースに表スペース MY_TABLESPACE をリストアします。

1. `db2 connect to my_database`
2. `db2 list tablespaces show detail`

   注: db2 list tablespaces show detail コマンドを実行して、すべての表スペースの状況を表示し、ステップ 5 で必要な表スペース ID 番号を入手してください。
3. `db2 stop hadr on database my_database`
4. `db2 "restore database my_database tablespace (my_tablespace) online redirect"`
5. `db2 "set tablespace containers for my_tablespace_ID_# ignore rollforward container operations using (path '/my_new_container_path/')"`
6. `db2 "restore database my_database continue"`
7. `db2 rollforward database my_database to end of logs and stop tablespace "(my_tablespace)"`
8. `db2 start hadr on database my_database as primary`

1 次データベースとスタンバイ・データベースは同じデータベース・パスを必要としません。相対コンテナー・パスが使用される場合、同じ相対パスであっても、1 次データベースとスタンバイ・データベースとで、異なる絶対コンテナー・パスにマッピングすることが可能です。

HADR はストレージ・グループを完全にサポートしています。これには、CREATE STOGROUP、ALTER STOGROUP、および DROP STOGROUP の各ステートメントのレプリケーションも含まれます。表スペース・コンテナーと同様、1 次およびスタンバイの両方にストレージ・パスが存在しなければなりません。

1 次データベースとスタンバイ・データベースのデータベース名は、同じでなければなりません。これは、両者が異なるインスタンスに入っていなければならないことを意味します。

リダイレクトされるリストアはサポートされません。つまり、HADR は表スペース・コンテナーのリダイレクトをサポートしません。ただし、データベース・ディレクトリーおよびログ・ディレクトリーの変更はサポートされています。相対パスによって作成される表スペース・コンテナーは、新規のデータベース・ディレクトリーから見た相対パスにリストアされます。

バッファー・プール要件

バッファー・プール操作もスタンバイ・データベースで再生されるため、 1 次データベースとスタンバイ・データベースのメモリー量を同じにしておくことは重要です。スタンバイ・データベースで読み取りを行う場合は、アクティブ・スタンバイ・データベースでログの適用および読み取りアプリケーションを実行できるよう

に、 1 次データベースのバッファー・プールを構成することが必要です。

高可用性災害時リカバリー (HADR) のインストールおよびストレージ要件

高可用性災害時リカバリー (HADR) での最良のパフォーマンスを実現するには、システムが以下のインストール要件とストレージ要件を満たすようにします。

インストール要件

HADR の場合、インスタンス・パスは、1 次データベースとスタンバイ・データベースで同じである必要があります。異なるインスタンス・パスを使用すると、一部の状況では問題が発生する可能性があります。例えば、SQL ストアード・プロシージャーがユーザー定義関数 (UDF) を呼び出し、UDF オブジェクト・コードへのパスが、1 次サーバーとスタンバイ・サーバーに対して同じディレクトリーが期待されるような場合です。

ストレージ要件

HADR はストレージ・グループを完全にサポートしています。これには、CREATE STOGROUP、ALTER STOGROUP、および DROP STOGROUP の各ステートメントのレプリケーションも含まれます。表スペース・コンテナーと同様、1 次およびスタンバイの両方にストレージ・パスが存在しなければなりません。同一のパスを作成するためにシンボリック・リンクを使用できます。1 次データベースとスタンバイ・データベースは、同じコンピューター上に存在していてもかまいません。データベース・ストレージが同じパスで始まっていても、使用される実際のディレクトリーの名前の中にはインスタンス名が組み込まれているため、両者が対立することはありません (1 次データベースとスタンバイ・データベースのデータベース名は同じでなければならないので、インスタンスは異なっている必要がある)。ストレージ・パスの形式は、`storage_path_name/inst_name/dbpart_name/db_name/tbsp_name/container_name` となります。

表スペースとそのコンテナーは、1 次データベースおよびスタンバイ・データベース上で同一でなければなりません。表スペース・タイプ (DMS または SMS)、表スペース・サイズ、コンテナー・パス、コンテナー・サイズ、およびコンテナー・ファイル・タイプ (ロー・デバイスまたはファイル・システム) などのプロパティーも同一でなければなりません。ストレージ・グループ、およびそのストレージ・パスはそれぞれ同一でなければなりません。これには、パス名とそれぞれのストレージ・グループにあてられるスペースの量も含まれます。ログ・ファイルに割り振られるスペース量も、1 次データベースとスタンバイ・データベースの両方で同じでなければなりません。

1 次データベースに対して CREATE TABLESPACE、ALTER TABLESPACE、または DROP TABLESPACE などの表スペース・ステートメントを発行すると、そのステートメントはスタンバイ・データベースで再生されます。 1 次データベースに対して表スペース・ステートメントを発行する前に、関係するデバイスを両方のデータベースでセットアップしておく必要があります。

表スペースのセットアップが 1 次データベースとスタンバイ・データベースで同一でない場合、スタンバイ・データベース上でのログの再生で OUT OF SPACE または TABLE SPACE CONTAINER NOT FOUND などのエラーが発生する場合があり

ます。同様に、ストレージ・グループのストレージ・パス・セットアップが 1 次データベースとスタンバイ・データベースで同一でない場合、CREATE STOGROUP または ALTER STOGROUP に関連付けられているログ・レコードは適用されません。その結果、既存のストレージ・パスはスタンバイ・システム上のスペースを早い時点で使い尽くし、自動ストレージ表スペースはサイズを大きくすることができなくなります。 これらの状態のいずれかが発生する場合、影響を受ける表スペースはロールフォワード・ペンディング状態に置かれ、後続のログの再生で無視されます。テークオーバー操作が発生する場合、アプリケーションで表スペースを使用できなくなります。

テークオーバーの前にスタンバイ・システム上の問題に気付いた場合の解決策は、ストレージの問題に取り組みつつ、スタンバイ・データベースを再確立することです。このステップには、次の作業が含まれます。

- スタンバイ・データベースを非活動状態にする。
- スタンバイ・データベースをドロップする。
- 必須のファイル・システムが存在し、後続のリストアおよびロールフォワードのための十分なフリー・スペースがあることを確認する。
- 1 次データベースの最新のバックアップを使用してスタンバイ・システムでデータベースをリストアする (または db2inidb コマンドによるスプリット・ミラーまたはフラッシュ・コピーを使用して再初期化する)。ストレージ・グループのストレージ・パスを、リストア中に再定義してはなりません。また、リストアの際に表スペース・コンテナーをリダイレクト・オプションによって変更しないでください。
- スタンバイ・システム上で HADR を再始動する。

ただし、テークオーバーが発生した後でスタンバイ・データベースの問題に気付いた場合 (またはこの時点までストレージの問題に取り組まないという選択が行われた場合)、解決策は発生した問題の種類によって異なります。

データベースで自動ストレージが使用可能で、スタンバイ・データベースに関連したストレージ・パスでスペースを使用できない場合、以下のステップに従ってください。

1. ファイル・システムを拡張するか、またはそこで不要な DB2 以外のファイルを除去することによってストレージ・パス上のスペースを使用可能にする。
2. ログの最後まで表スペースのロールフォワードを実行する。

ログの再生の一部としてコンテナーを追加または拡張できない場合、必須のバックアップ・イメージおよびログ・ファイルのアーカイブが使用可能であれば、まず IGNORE ROLLFORWARD CONTAINER OPERATIONS オプションを付けて SET TABLESPACE CONTAINERS ステートメントを発行し、次に ROLLFORWARD コマンドを発行すると、表スペースをリカバリーできる可能性があります。

1 次データベースとスタンバイ・データベースは同じデータベース・パスを必要としません。相対コンテナー・パスが使用される場合、同じ相対パスであっても、1 次データベースとスタンバイ・データベースとで、異なる絶対コンテナー・パスにマッピングすることが可能です。したがって、1 次データベースとスタンバイ・データベースが同じコンピューター上に位置する場合、すべての表スペース・コンテナーは、1 次データベースとスタンバイ・データベースとで異なるパスにマップさ

れるように、相対パスを使って定義することが必要です。

HADR およびネットワーク・アドレス変換 (NAT) のサポート

HADR 環境でサポートされる NAT は、サーバーの実アドレスを非表示にするため、通常はファイアウォールとセキュリティー用に使用されます。

HADR セットアップでは、1 次ノードとスタンバイ・ノードのローカル・ホストおよびリモート・ホストの構成が正しいことを確認するためにクロス・チェックが行われます。NAT 環境では、ホストはそれ自体に対しては特定の IP アドレスで識別されますが、他のホストに対しては別の IP アドレスで識別されます。この性質が原因で、HADR ホストのクロスチェックは失敗します。ただし、**DB2_HADR_NO_IP_CHECK** レジストリー変数を ON に設定している場合を除きます。この設定を使用すると、ホストのクロス・チェックはバイパスされ、1 次とスタンバイが NAT 環境で接続できるようになります。

NAT 環境で実行しない場合は、**DB2_HADR_NO_IP_CHECK** レジストリー変数にデフォルト設定の OFF を使用します。クロス・チェックを無効にすると、HADR による構成の妥当性検査機能が弱くなります。

HADR 複数スタンバイ・モードに関する考慮事項

複数スタンバイ・セットアップを使用する NAT 環境では、**hadr_local_host** および **hadr_local_svc** の各スタンバイの設定を 1 次の **hadr_target_list** で引き続きリストする必要があります。そうしないと、1 次はスタンバイからの接続を受け入れません。

通常、複数スタンバイ・モードでは、スタンバイは始動時に **hadr_remote_host** と **hadr_remote_svc** の設定がその **hadr_target_list** 内にあることをチェックし、役割の切り替え時に古い 1 次が新しいスタンバイになれることを確認します。NAT シナリオでは、**DB2_HADR_NO_IP_CHECK** レジストリー変数を ON に設定した場合を除き、このチェックは失敗します。このチェックをバイパスすると、スタンバイは、1 次の **hadr_local_host** と **hadr_local_svc** がスタンバイの **hadr_target_list** にあることをチェックするために、1 次に接続されるまで待機します。その場合でも、このペアで正常に役割が切り替えられることを確認するチェックが行われます。

注: **DB2_HADR_NO_IP_CHECK** レジストリー変数が ON に設定されている場合、**hadr_remote_host** と **hadr_remote_svc** は自動的に更新されません。

複数スタンバイ・セットアップでは、NAT 境界を超えて他のデータベースに接続している可能性のあるすべてのデータベースに **DB2_HADR_NO_IP_CHECK** を設定する必要があります。データベースが別のデータベースに接続するために NAT 境界を超えることがない場合 (つまり、そのようなリンクが構成されていない場合) は、このレジストリー変数をそのデータベースに設定する必要はありません。**DB2_HADR_NO_IP_CHECK** を設定すると、テークオーバー発生後にスタンバイが新しい 1 次を自動的に検出できなくなるため、ユーザーはスタンバイが新しい 1 次に接続するように手動で再構成する必要があります。

高可用性災害時リカバリー (HADR) の制約事項

高可用性災害時リカバリー (HADR) での最良のパフォーマンスを実現するには、高可用性 DB2 データベース・ソリューションを設計するときに、 HADR の制限を検討してください。

次のリストは、高可用性災害時リカバリー (HADR) の制約事項をサマリーしています。

- HADR は、パーティション・データベース環境ではサポートされません。
- DB2 pureScale環境では HADR はサポートされません。
- 1 次データベースとスタンバイ・データベースでは、同じオペレーティング・システム・バージョンであることと、同じバージョンの DB2 データベース・システムを使用することが求められます。ただし、ローリング・アップグレード時の短期間は除きます。
- 1 次データベース上とスタンバイ・データベース上の DB2 データベース・システム・ソフトウェアは、同じビット・サイズ (32 または 64 ビット) でなければなりません。
- スタンバイ・データベースの読み取りを使用可能にしない限り、クライアントは、スタンバイ・データベースに接続できません。スタンバイ・データベースの読み取りを使用可能にすると、クライアントはアクティブ・スタンバイ・データベースに接続し、読み取り専用照会を実行できます。
- スタンバイ・データベースの読み取りを使用可能にしても、ログ・レコードを書き込む操作は、スタンバイ・データベース上で実行できません。アクティブ・スタンバイ・データベースに接続できるのは、読み取り専用クライアントのみです。
- スタンバイ・データベースの読み取りを使用可能にしても、データベースの内容を変更する書き込み操作は、スタンバイ・データベース上では許可されません。リアルタイム統計収集や自動索引再作成のような非同期スレッド、およびデータベース・オブジェクトを変更しようとするユーティリティーは、サポートされません。リアルタイム統計収集および自動索引再作成は、スタンバイ・データベース上で実行しないでください。
- ログ・ファイルのアーカイブは、1 次データベースによってのみ行われます。
- Self Tuning Memory Manager (STMM) は、現在の 1 次データベースでのみ実行可能です。1 次データベースの開始後、またはスタンバイ・データベースがテークオーバーによって 1 次データベースに変換された後でも、最初のクライアント接続が確立されるまで STMM EDU が開始しない場合もあります。
- スタンバイ・データベースでのバックアップ操作はサポートされていません。
- データベース構成パラメーターやリカバリー履歴ファイルへの変更など、ログに記録されない操作は、スタンバイ・データベースに複製されません。
- **COPY** NO オプションを指定したロード操作はサポートされません。
- HADR は、データベース・ログ・ファイルでのロー I/O (直接ディスク・アクセス) の使用をサポートしていません。HADR が **START HADR** コマンドを介して開始された場合、または HADR を構成した状態でデータベースがアクティブにされた (再始動された) 場合、ロー・ログが検出されると、関連するコマンドは失敗します。

- 1 フェーズ・コミットの場合、HADR データベースはフェデレーテッド・サーバー (トランザクション・マネージャー) あるいはデータ・ソース (リソース・マネージャー) のどちらの役割も果たすことができます。2 フェーズ・コミットの場合は、HADR データベースはデータ・ソースとしてのみ動作します。
- HADR は無限ロギングをサポートしません。
- HADR 1 次データベースのシステム・クロックは、HADR スタンバイ・データベースのシステム・クロックと同期していなければなりません。

高可用性のための保守のスケジュール

DB2 データベース・ソリューションは、定期的な保守を必要とします。ソフトウェアまたはハードウェアのアップグレード、データベースのパフォーマンス・チューニング、データベースのバックアップ、ビジネス上の目的のための統計収集およびモニターなどの、保守を実行する必要があります。これらの保守活動がデータベース・ソリューションの可用性に与える影響を最小限にする必要があります。

始める前に

保守活動をスケジュールする前に、データベース・ソリューション上で実行する必要のある保守活動を識別しなければなりません。

手順

保守をスケジュールするには、以下のステップを実行します。

1. データベース活動が少なくなる期間を識別します。

   活動が少ない時間 (データベース・システムに要求を出すユーザー・アプリケーションが最も少なくなる期間) に保守活動をスケジュールすることが最善です。作成するビジネス・アプリケーションのタイプに応じて、データベース・システムにアクセスするユーザー・アプリケーションがない期間が存在することもあります。
2. 以下に従って、実行の必要な保守活動をカテゴリー化してください。
   - 自動化できる保守
   - 保守の実行中にデータベース・ソリューションをオフラインにする必要があるもの
   - データベース・ソリューションがオンラインのときに保守を実行できるもの
3. 自動化できる保守活動については、以下のいずれかの方法で自動保守を構成します。
   - **auto_maint** 構成パラメーターを使用する
   - AUTOMAINT_SET_POLICY および AUTOMAINT_SET_POLICYFILE と呼ばれるシステム・ストアード・プロシージャーのいずれか 1 つを使用する
4. 実行する必要のあるいずれかの保守活動でデータベース・サーバーをオフラインにすることが必要な場合、活動の少ない時間にそれらのオフライン保守活動をスケジュールします。
5. データベース・サーバーがオンラインのときに実行できる保守活動については、以下のようにします。

- それらのオンライン保守活動を実行することによる可用性への影響を識別します。
- それら保守活動を実行することによるデータベース・システムの可用性への影響が最小となるように、オンライン保守活動をスケジュールします。

例えば、活動の少ない時間にオンライン保守活動をスケジュールして、スロットル・メカニズムの使用により、保守活動が使用するシステム・リソースの量のバランスを取ります。

SYSPROC.AUTOMAINT_SET_POLICY または SYSPROC.AUTOMAINT_SET_POLICYFILE を使用した自動保守ポリシーの構成

システム・ストアード・プロシージャー AUTOMAINT_SET_POLICY および AUTOMAINT_SET_POLICYFILE を使用して、データベース用の自動保守ポリシーを構成できます。

手順

データベースの自動保守ポリシーを構成するには、以下のステップを実行します。

1. データベースに接続します
2. AUTOMAINT_SET_POLICY または AUTOMAINT_SET_POLICYFILE を呼び出します
   - AUTOMAINT_SET_POLICY に必要なパラメーターは、次のとおりです。
     a. 構成する自動保守活動のタイプを指定する、保守タイプ。
     b. 自動保守ポリシーを XML 形式で指定する BLOB へのポインター。
   - AUTOMAINT_SET_POLICYFILE に必要なパラメーターは、次のとおりです。
     a. 構成する自動保守活動のタイプを指定する、保守タイプ。
     b. 自動保守ポリシーを指定する XML ファイルの名前。

   有効な保守タイプは、以下のとおりです。
   - AUTO_BACKUP - 自動バックアップ
   - AUTO_REORG - 表および索引の自動再編成
   - AUTO_RUNSTATS - 自動の表 **RUNSTATS** 操作
   - MAINTENANCE_WINDOW - 保守期間

次のタスク

システム・ストアード・プロシージャー AUTOMAINT_GET_POLICY および AUTOMAINT_GET_POLICYFILE を使用して、データベース用に構成された自動保守ポリシーを検索できます。

AUTOMAINT_SET_POLICY または AUTOMAINT_SET_POLICYFILE の自動保守ポリシーを指定する XML のサンプル

自動保守ポリシーを指定するために AUTOMAINT_SET_POLICY または AUTOMAINT_SET_POLICYFILE のどちらを使用していても、XML を使用してポリシーを指定する必要があります。自動保守ポリシーを XML で指定する方法を例示するサンプル・ファイルが、SQLLIB/samples/automaintcfg にあります。

システム・ストアード・プロシージャー AUTOMAINT_SET_POLICY に渡す 2 番目のパラメーターは、目的の自動保守ポリシーを指定する XML を含む BLOB です。システム・ストアード・プロシージャー AUTOMAINT_SET_POLICYFILE に渡す 2 番目のパラメーターは、目的の自動保守ポリシーを指定する XML ファイルの名前です。 AUTOMAINT_SET_POLICY に渡す BLOB で有効な XML エレメントは、 AUTOMAINT_SET_POLICYFILE に渡す XML ファイルで有効なエレメントと同じです。

サンプル・ディレクトリー SQLLIB/samples/automaintcfg には、自動保守ポリシー指定の例を含む 4 つの XML ファイルがあります。

DB2MaintenanceWindowPolicySample.xml

データベース・マネージャーが自動保守をスケジュールする必要のある保守期間の指定方法を例示します。

DB2AutoBackupPolicySample.xml

データベース・マネージャーが自動バックアップをどのように実行するかについての指定方法を例示します。

DB2AutoReorgPolicySample.xml

データベース・マネージャーが表および索引の自動再編成をどのように実行するかについての指定方法を例示します。

DB2DefaultAutoRunstatsPolicySample.xml

データベース・マネージャーが表の自動的な **runstats** 操作をどのように実行するかについての指定方法を例示します。

これらのファイルから XML をコピーして、システムの要件に応じてその XML を変更することにより、独自の自動保守ポリシー指定 XML を作成できます。

データベース・ロギング・オプションの構成

データベース・ロギング構成パラメーターを使用して、使用するロギングのタイプ、ログ・ファイルのサイズ、ログ・ファイルが格納されるロケーションなどの、データベースのデータ・ロギング・オプションを指定します。

始める前に

データベース・ロギング・オプションを構成するには、SYSADM、SYSCTRL、または SYSMAINT 権限が必要です。

このタスクについて

データベース・ロギング・オプションは、コマンド行プロセッサー (CLP) で **UPDATE DATABASE CONFIGURATION** コマンドを使用する、または db2CfgSet API を呼び出すことにより構成できます。

手順

- コマンド行プロセッサーで **UPDATE DATABASE CONFIGURATION** コマンドを使用してデータベース・ロギング・オプションを構成するには、以下のようにします。
  1. 循環ロギングまたはアーカイブ・ロギングのどちらを使用するか指定します。循環ロギングを使用する場合、**logarchmeth1** および **logarchmeth2** データベース構成パラメーターは OFF に設定する必要があります。この設定はデフォルトです。アーカイブ・ロギングを使用するには、これらのデータベース構成パラメーターのうちの少なくとも 1 つを OFF 以外の値に設定する必要があります。例えば、アーカイブ・ロギングを使用し、アーカイブ・ログをディスクに保管する場合、以下のコマンドを発行します。

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

     アーカイブ・ログは /u/dbuser/archived_logs というディレクトリーに置かれます。
  2. 必要に応じて、他のデータベース・ロギング構成パラメーターの値を指定します。以下の追加の構成パラメーターが、データベース・ロギングに適用されます。
     - **archretrydelay**
     - **blk_log_dsk_ful**
     - **failarchpath**
     - **logarchcompr1**
     - **logarchcompr2**
     - **logarchmeth1**
     - **logarchmeth2**
     - **logarchopt1**
     - **logarchopt2**
     - **logbufsz**
     - **logfilsiz**
     - **logprimary**
     - **logsecond**
     - **max_log**
     - **mirrorlogpath**
     - **newlogpath**
     - **mincommit**
     - **numarchretry**
     - **num_log_span**
     - **overflowlogpath**

これらのデータベース・ロギング構成パラメーターの詳細は、『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。

- IBM Data Studio を使用してデータベース・ロギング・オプションを構成するには、**UPDATE DATABASE CONFIGURATION** コマンドのタスク・アシストを使用します。

関連情報:

データベース・ロギングの構成パラメーター

高可用性ストラテジーには、データベース・ロギングという重要なエレメントがあります。データベース・ログを使用することにより、トランザクション情報の記録、1 次データベースと 2 次 (スタンバイ) データベースの同期、および障害が発生した 1 次データベースをテークオーバーした 2 次データベースのロールフォワードを行うことができます。これらのデータベース・ロギング・アクティビティーを構成するには、各種データベース構成パラメーターを設定する必要があります。

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

ログ・ディスク・フルによるアプリケーション中断 (blk_log_dsk_ful)

このデータベース構成パラメーターを設定して、 DB2 データベース・マネージャーで新しいログ・ファイルをアクティブ・ログ・パス内に作成できない場合に、ディスク満杯エラーが発生し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 データベース・マネージャーは 5 分おきにログ・ファイルの作成を試行します。 試行するたびに、DB2 データベース・マネージャーはメッセージを管理通知ログに書き込みます。ログ・ディスクが満杯になったためにアプリケーションがハングしているかを確認するには、管理通知ログをモニターするしかありません。ログ・ファイルが正常に作成されるまでは、表データの更新を試行するユーザー・アプリケーションにより、トランザクションをコミットすることができません。読み取り専用照会が直接影響を受ける可能性はありませんが、照会が更新要求によってロックされているデータ、または更新アプリケーションによってバッファー・プール内に固定されているデータにアクセスする必要がある場合は、読み取り専用照会もブロックされます。

blk_log_dsk_ful を YES に設定すると、 DB2 データベース・マネージャーにログ・ディスク満杯エラーが発生する時に、アプリケーションがハングする原因になります。その時エラーを解決し、アプリケーションを継続することができます。ディスク満杯の状態は、古いログ・ファイルを別のファイル・システムに移動したり、ハングを起こしているアプリケーションが完了できるようにファイル・システムのサイズを増やしたり、または、ログのアーカイブの失敗について調査して対処したりすることによって解決できます。

blk_log_dsk_ful が NO に設定されている場合には、ログ・ディスク満杯エラーを受け取るトランザクションは失敗し、ロールバックが行われます。

フェイルオーバー・アーカイブ・パス (failarchpath)
通常のアーカイブ・パスに問題が生じた場合 (アクセスできない、満杯であるなど) のために、アーカイブ・ログ・ファイルの代替ディレクトリーを指

定します。このディレクトリーは、失敗したログ・アーカイブ・メソッドがもう一度使用可能になるまでの、一時的なログ・ファイル用ストレージ域です。メソッドがもう一度使用可能になると、ログ・ファイルは、このディレクトリーから元のログ・アーカイブの指定パスに移されます。ログ・ファイルをこの一時ロケーションに移動することにより、ログ・ディレクトリーが満杯になる状態を避けられます。このパラメーターは、完全に修飾された既存のディレクトリーでなければなりません。

1 次ログ・アーカイブの圧縮 (logarchcompr1)、2 次ログ・アーカイブの圧縮 (logarchcompr2)

特定の環境では、これらのパラメーターによって、データベース・マネージャーがアーカイブ・ログ・ファイルを圧縮するかどうかが制御されます。ログ・アーカイブ・ファイルに圧縮を使用する場合、そのファイルの保管に関するコストを削減できます。

これらのパラメーターの有効値は以下のとおりです。

OFF この値は、ログ・アーカイブ・ファイルを圧縮しないことを指定します。デフォルト値は OFF です。

ON この値は、ログ・アーカイブ・ファイルを圧縮することを指定します。動的に設定されている場合、既にアーカイブされているログ・ファイルは圧縮されません。

注:

1. **logarchmeth1** 構成パラメーターを DISK、TSM、VENDOR 以外の値に設定すると、**logarchcompr1** 構成パラメーターの設定に関係なく、ログ・アーカイブ圧縮の効果はありません。
2. **logarchmeth2** 構成パラメーターを DISK、TSM、VENDOR 以外の値に設定すると、**logarchcompr2** 構成パラメーターの設定に関係なく、ログ・アーカイブ圧縮の効果はありません。

ログ・アーカイブ・メソッド 1 (logarchmeth1)、ログ・アーカイブ・メソッド 2 (logarchmeth2)

これらのパラメーターを指定すると、データベース・マネージャーは、ログ・ファイルを、アクティブ・ログ・パスではないロケーションにアーカイブできます。これらのパラメーターの両方を指定すると、**logpath** 構成パラメーターによって設定されているアクティブ・ログ・パスの各ログ・ファイルは 2 回アーカイブされます。つまり、2 つの別個の宛先に、ログ・パスからのアーカイブ・ログ・ファイルの同一コピーが 2 つできることになります。 **mirrorlogpath** 構成パラメーターを使用してミラー・ロギングを指定すると、**logarchmeth2** 構成パラメーターは、アクティブ・ログ・パスにあるログ・ファイルの追加のコピーをアーカイブする代わりに、ミラー・ログ・パスからログ・ファイルをアーカイブします。つまり、2 つの別個の宛先に、ログ・ファイルの別個のコピーが 2 つ (ログ・パスからのコピーが 1 つと、ミラー・ログ・パスからのコピーが 1 つ) アーカイブされることになります。

これらのパラメーターの有効値は以下のとおりです。

OFF この値は、ログ・アーカイブ方式が使用されないことを指定します。 **logarchmeth1** と **logarchmeth2** の両方の構成パラメーターを

OFF に設定すると、データベースは循環ロギングを使用していると見なされ、ロールフォワード・リカバリー可能ではなくなります。デフォルト値は OFF です。

LOGRETAIN
この値は、**logarchmeth1** 構成パラメーターについてのみ使用できます。この値を設定することは、**logretain** 構成パラメーターを RECOVERY に設定することと同じです。LOGRETAIN 値を指定する場合、**logretain** 構成パラメーターが自動的に更新されます。

USEREXIT
この値は、**logarchmeth1** 構成パラメーターにのみ使用できます。この値を設定することは、**userexit** 構成パラメーターを ON に設定することと同じです。USEREXIT 値を指定する場合、**userexit** 構成パラメーターが自動的に更新されます。

DISK この値の後にコロン (:) を付け、その後に、ログ・ファイルがアーカイブされる完全修飾された既存のパス名を続ける必要があります。例えば、**logarchmeth1** 構成パラメーターを DISK:/u/dbuser/archived_logs に設定した場合、アーカイブ・ログ・ファイルは /u/dbuser/archived_logs/*INSTANCE_NAME*/*DBNAME*/NODE*xxxx*/LOGSTREAM*xxxx*/C*xxxxxxx* ディレクトリー以下に配置されます。

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

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

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

注:

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

ログ・アーカイブ・オプション 1 (logarchopt1)、ログ・アーカイブ・オプション 2 (logarchopt2)
TSM API またはベンダー API に渡されるストリングを指定します。

TSM 環境の場合、このパラメーターを使用すれば、別の TSM ノード上で、別の TSM ユーザーによって、または DB2 pureScale環境のようにプロキシー・ノードを使用する TSM 環境で生成されたログを、データベースがリトリーブできるようになります。このストリングは、次のいずれかの形式で指定する必要があります。

- TSM サーバーがプロキシー・ノード・クライアントをサポートするように構成されていないときに、別の TSM ノード上で生成されたログを検索する場合:

  "-fromnode=*nodename*"

- TSM サーバーがプロキシー・ノード・クライアントをサポートするように構成されていないときに、別の TSM ユーザーによって生成されたログを検索する場合:

  "-fromowner=*ownername*"

- TSM サーバーがプロキシー・ノード・クライアントをサポートするように構成されていないときに、別の TSM ノード上で別の TSM ユーザーによって生成されたログを検索する場合:

  "-fromnode=*nodename* -fromowner=*ownername*"

- 複数のメンバーが同じデータを処理する DB2 pureScale環境のような、クライアント・プロキシー・ノード構成で生成されたログを検索する場合:

  "-asnodename=*proxynode*"

  ここで、*nodename* はログ・ファイルを最初にアーカイブした TSM ノードの名前、*ownername* はログ・ファイルを最初にアーカイブした TSM ユーザーの名前、*proxynode* は共有 TSM ターゲット・プロキシー・ノードの名前です。それぞれのログ・アーカイブ・オプション・フィールドには、対応する 1 つのログ・アーカイブ・メソッドがあります。*logarchopt1* は *logarchmeth1* で使用され、*logarchopt2* は *logarchmeth2* で使用されます。

注:

- -asnodename TSM オプションを使用する場合は、データの保管に、各メンバーのノード名 (nodename) が使用されません。代わりに、DB2 pureScaleインスタンス内のすべてのメンバーで使用される共有 TSM ターゲット・ノードの名前を使用して、データは保管されます。
- -fromnode オプションと -fromowner オプションは -asnodename オプションと両立しないため、併用できません。 -asnodename オプションはプロキシー・ノードを使用する TSM 構成用に、他の 2 つのオプションはその他の種類の TSM 構成に使用してください。詳しくは、 459 ページの『Tivoli Storage Manager クライアントの構成』を参照してください。

ログ・バッファー (logbufsz)

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

- トランザクションのコミット
- ログ・バッファーが満杯

- 他の何らかの内部データベース・マネージャー・イベントの結果として

ログ・バッファーのサイズを大きくすると、ログ・レコードがディスクに書き込まれる頻度が少なくなり、一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) アクティビティーをさらに効率的にすることができます。しかし、ログ・バッファー・サイズの値が大きければ大きいほど、リカバリーは長くかかります。また、*logbufsz* の設定を大きくすると、ログ・ディスクからの読み取り回数を削減できる可能性があります (これがシステムに有益かどうかを判断するには、log_reads モニター・エレメントを使用して、ログ・ディスクからの読み取りが多いかどうかを確認してください)。

ログ・ファイル・サイズ (logfilsiz)

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

ログ・ストリームあたりの合計アクティブ・ログ・スペースとして構成できる値には、1024 GB という論理的な限界があります。この限界は、各ログ・ファイルの上限 (4 GB) と、1 次および 2 次ログ・ファイルを組み合わせた最大数 (256) の計算結果です。

ログ・ファイルのサイズは、パフォーマンスに直接影響します。ログの切り替えにはパフォーマンス・コストがあります。従って、純粋にパフォーマンスだけを考えると、ログ・ファイルのサイズが大きければ大きいほどよいということになります。一方、このパラメーターは、アーカイブ用のログ・ファイル・サイズも表しています。この点では、ログ・ファイルのサイズが大きければ必ずしもよいというわけではありません。ログ・ファイルのサイズが大きくなると、障害の発生が増加したり、ログ・シッピング・シナリオに遅延が発生する可能性があるからです。アクティブ・ログ・スペースを考慮する時には、小さいログ・ファイルを多く持つ方がよいかもしれません。例えば、ログ・スペースが非常に大きな 2 つのログ・ファイルで占められていた場合、 1 つのログ・ファイルの終わりに近いところでトランザクションが開始した時には、アクティブ・ログとしては、全ログ・スペースの半分しか使用できないことになります。

データベースが非活動になる (データベースへのすべての接続が終了する) たびに、現在書き込みが行われているログ・ファイルは、切り捨てられます。それで、データベースが頻繁に非活動になる場合には、DB2 データベース・マネージャーが大きなファイルを作成しても切り捨てられてしまうため、大きなログ・ファイル・サイズを選択しないほうが賢明です。ACTIVATE DATABASE コマンドを使用すると、このような無駄を避けることができます。このコマンドは、最後のクライアントがデータベースから切断した時に、データベースが自動的に非アクティブ化されることを防ぐからです。

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

ログ・ファイルの損失を最小限にすることも、ログ・サイズを設定する際の重要な考慮事項の 1 つです。アーカイブ処理は、一度に 1 つのログ・ファイル全体を処理します。大きなログ・ファイルを構成している場合は、アー

カイブが間隔を長くなります。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・ファイル・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、どの時点においても、まだアーカイブされていないログ・データが平均的に少なくなるため、メディア障害が発生しても失う情報量は少なくなります。

ログ保持 (logretain)

この構成パラメーターは、*logarchmeth1* に置き換えられました。現在は、以前のバージョンの DB2 データベース・ソフトウェアとの互換性のためにサポートされています。

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

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

トランザクションごとの最大ログ (max_log)

このパラメーターは、1 つのトランザクションで消費できる 1 次ログ・スペースのパーセンテージを示します。値は、*logprimary* 構成パラメーターに指定された値のパーセントを示します。

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

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

このパラメーター、および *num_log_span* 構成パラメーターは、無限のアクティブ・ログ・スペースが使用可能なときに役立つことがあります。無限ロギングがオン (つまり *logsecond* が -1) の場合、トランザクションはログ・ファイル数の上限 (*logprimary* + *logsecond*) によって制限されません。*logprimary* の値に到達した場合、DB2 データベース・マネージャーはトランザクションを失敗させる代わりに、アクティブ・ログのアーカイブを開始します。これは、例えば、コミットされずに残っている長期実行トランザクション (ロジックにエラーがあるアプリケーションによって生じる可能性があります) が存在する場合に、問題となることがあります。この状況では、アクティブ・ログ・スペースは増大を続けて、クラッシュ・リカバリーのパフォーマンスが低下することがあります。これを防止するには、*max_log* および *num_log_span* 構成パラメーターのどちらか一方または両方の値を指定できます。

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

ログ・パスのミラー (mirrorlogpath)

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

mirrorlogpath パラメーターはまた、ログ・アーカイブの振る舞いに影響します。これを利用して、ロールフォワード・リカバリー中の回復力を向上させることができます。**mirrorlogpath** と **logarchmeth2** の両方が設定されていると、**logarchmeth2** は、アクティブ・ログ・パスにあるログ・ファイルの追加のコピーをアーカイブする代わりに、ミラー・ログ・パスからログ・ファイルをアーカイブします。このログ・アーカイブの振る舞いを利用して、回復力を向上させることができます。なぜなら、1 次ログ・ファイルが、アーカイブの前にディスク障害により破損した場合でも、ミラー・ログ・パスにある、使用可能なアーカイブ・ログ・ファイルを使用してデータベースのリカバリー操作を続行できる可能性があるからです。

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

そのログの位置を記録しておく必要があります。

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

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

注:

1. この構成パラメーターは、1 次ログ・パスがロー・デバイスである場合には、サポートされません。
2. このパラメーターのために指定される値は、ロー・デバイスにはなり得ません。
3. DB2 pureScale環境では、最初にデータベースに接続した、またはデータベースをアクティブ化したメンバーが、このログ・パス・パラメーターに対する構成変更を処理します。 DB2 データベース・マネージャーは、パスが存在することと、そのパスに対する読み取りおよび書き込み権限の両方があることを確認します。また、メンバー固有のログ・ファイル用サブディレクトリーを作成します。これらの操作のいずれかに失敗すると、DB2 データベース・マネージャーは、指定されたパスを拒否し、古いパスを使用してデータベースをオンラインにします。指定されたパスが受け入れられた場合は、その新しい値が各メンバーに伝搬され

ます。新しいパスへの切り替えの試行中に、いずれかのメンバーに障害が起こった場合、それ以降、そのデータベースに対してアクティブ化または接続を試行しても失敗します (SQL5099N)。すべてのメンバーが同じログ・パスを使用する必要があります。

新規ログ・パス (newlogpath)

データベース・ログは、最初に次のディレクトリー内に作成されます: *db_path*/*instance_name*/*dbname*/NODE0000/LOGSTREAM0000。アクティブ・ログ・ファイル (および今後のログ・ファイル) の保管場所を変更するには、別のディレクトリーまたは装置を指すように、この構成パラメーターの値を変更します。データベースの構成が現在ロールフォワード・リカバリー用になっている場合、データベース・ログ・パス・ディレクトリーに保管されるアクティブ・ログは新規の保管位置に移動されません。

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

そのログの位置を記録しておく必要があります。

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

注: DB2 pureScale環境では、最初にデータベースに接続した、またはデータベースをアクティブ化したメンバーが、このログ・パス・パラメーターに対する構成変更を処理します。 DB2 データベース・マネージャーは、パスが存在することと、そのパスに対する読み取りおよび書き込み権限の両方があることを確認します。また、メンバー固有のログ・ファイル用サブディレクトリーを作成します。これらの操作のいずれかに失敗すると、DB2 データベース・マネージャーは、指定されたパスを拒否し、古いパスを使用してデータベースをオンラインにします。指定されたパスが受け入れられた場合は、その新しい値が各メンバーに伝搬されます。新しいパスへの切り替えの試行中に、いずれかのメンバーに障害が起こった場合、それ以降、そのデータベースに対してアクティブ化または接続を試行しても失敗します (SQL5099N)。すべてのメンバーが同じログ・パスを使用する必要があります。

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、複数のアプリケーションがトランザクションのコミットをほぼ同時に試行する場合だけです。コミットのグループ化が有効な場合、アプリケーションのコミット要求は、1 秒経過するか、またはコミット要求回数がこのパラメーター値に達するまで保留されます。

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

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

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

このパラメーター、および *max_log* 構成パラメーターは、無限のアクティブ・ログ・スペースが使用可能なときに役立つことがあります。無限ロギングがオン (つまり *logsecond* が -1) の場合、トランザクションはログ・ファイル数の上限 (*logprimary* + *logsecond*) によって制限されません。*logprimary* の値に到達した場合、DB2 データベース・マネージャーはトランザクションを失敗させる代わりに、アクティブ・ログのアーカイブを開始します。これは、例えば、コミットされずに残っている長期実行トランザクション (ロジックにエラーがあるアプリケーションによって生じる可能性があります) が存在する場合に、問題となることがあります。この状況では、アクティブ・ログ・スペースは増大を続けて、クラッシュ・リカバリーのパフォーマンスが低下することがあります。これを防止するには、*max_log* および *num_log_span* 構成パラメーターのどちらか一方または両方の値を指定できます。

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

オーバーフロー・ログ・パス (overflowlogpath)
このパラメーターは、ロギング要件に応じていくつかの関数で使用できます。 DB2 データベース・マネージャーがロールフォワード操作に必要なログ・ファイルを検索するために、場所を指定できます。それは、ROLLFORWARD コマンドの OVERFLOW LOG PATH オプションに似ていますが、発行されるすべての ROLLFORWARD コマンドに OVERFLOW LOG PATH オプションを指定する代わりに、この構成パラメーターを一度だけ設定します。もし両方とも使用される場合には、そのロールフォワード操作で指定した OVERFLOW LOG PATH オプションが *overflowlogpath* 構成パラメーターを上書きして実行します。

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

overflowlogpath が指定されていない場合には、 DB2 データベース・マネージャーはログ・ファイルを検索してアクティブ・ログ・パスに入れます。このパラメーターを指定することによって、リトリーブしたログ・ファイルを保管するために DB2 データベース・マネージャーが使用できるストレージ・リソースを追加できます。利点としては、入出力コストが異なるディスクに拡散することや、アクティブ・ログ・パスに保管できるログ・ファイルが増えることなどがあります。

例えば、複製のために **db2ReadLog** API を使用している場合、*overflowlogpath* を使用して、DB2 データベース・マネージャーがこの API に必要なログ・ファイルを検索するための場所を指定できます。ログ・ファイルが (アクティブ・ログ・パスまたはオーバーフロー・ログ・パスのいずれにも) 見つからない場合、データベースがログをアーカイブするように構成されていると、DB2 データベース・マネージャーはそのログ・ファイルをリトリーブします。さらに、このパラメーターを使用して、検索したログ・ファイルを DB2 データベース・マネージャーが保管するためにディレクトリーを指定できます。利点としては、アクティブ・ログ・パスの入出力コストの削減、およびアクティブ・ログ・パスに保管できるログ・ファイル数の増加などがあります。

無限ロギングが構成されている場合 (つまり、*logsecond* が -1 に設定されている場合)、*overflowlogpath* を設定することは有用です。 DB2 データベース・マネージャーは、アーカイブからリトリーブしたアクティブ・ログ・ファイルを、このパスに保管することができます。(無限ロギングでは、ロールバックまたはクラッシュ・リカバリーの操作時に、アクティブ・ログ・ファイルがアクティブ・ログ・パスに存在しない場合、それらのファイルをアーカイブからリトリーブしなければならないことがあります。)

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

overflowlogpath を設定するには、最大 242 バイトのストリングを指定します。ストリングがパス名を示す必要があり、これは相対パス名ではなく、絶対パス名でなければなりません。パス名はディレクトリーでなければならず、ロー・デバイスではありません。

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

1 次ログ・ファイル (logprimary)
このパラメーターには、作成するサイズ *logfilsiz* の 1 次ログの個数を指定します。

空の場合でも満杯の場合でも、1 次ログ・ファイルには同じ容量のディスク・スペースが必要です。従って、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・フル状態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・フル状態を処理できるかどうかを考慮する必要があります。アクティブ・ログ・スペースのログ・ファイルの合計サイズの限界は 256 GB です。

既存データベースでロールフォワード・リカバリーを使用可能にしている場合は、1 次ログの数を、 1 次ログと 2 次ログの合計数に 1 を足した数に変更する必要があります。

2 次ログ (logsecond)

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

1 次ログ・ファイルが満杯になると、 *logfilsiz* で指定されたサイズの 2 次ログ・ファイルが必要に応じて一度に 1 つずつ割り振られます。このパラメーターは、その最大数を指定します。このパラメーターが -1 に設定されている場合には、データベースは無限のアクティブ・ログ・スペースで構成されます。データベース上で実行中の未完了トランザクションのサイズまたは数には、制限がありません。通常 1 次ログに割り振るログ・スペースよりも多くのログ・スペースを必要とする大規模ジョブに対応しなければならない環境では、無限のアクティブ・ロギングが役立ちます。

注:

1. *logsecond* を -1 に設定するためには、ログ・アーカイブが使用可能になっている必要があります。
2. このパラメーターが -1 に設定されている場合には、 DB2 データベース・マネージャーがアーカイブ・ログ・ファイルを検索する必要があるかもしれないので、クラッシュ・リカバリー時間が増える可能性があります。

ユーザー出口 (userexit)

この構成パラメーターは、*logarchmeth1* に置き換えられました。現在は、以前のバージョンの DB2 データベース・ソフトウェアとの互換性のためにサポートされています。

このパラメーターを使うと、データベース・マネージャーでログのアーカイブと検索のためのユーザー出口プログラムを呼び出せるようになります。ログ・ファイルはアクティブ・ログ・パスでない場所にアーカイブされます。*userexit* を ON に設定すると、ロールフォワード・リカバリーを使用できるようになります。

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

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

テープ装置には、このほかにも考慮事項があります。アーカイブ要求の頻度も重要です。例えば、コピー操作の完了に要する時間が 5 分間であるとす

ると、ログは、ワークロードがピークの時の 5 分間分のログ・データを保持できるだけの十分な大きさになっている必要があります。テープ装置には、1 日あたりの操作回数に設計上の限界がある場合があります。ログ・サイズを決定するときは、これらの要因を考慮するようにしてください。

注:

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

NOT LOGGED INITIALLY パラメーターによるロギングの低減

アプリケーションでマスター表から作業表を作成してその中にデータを入れる場合に、マスター表から簡単に再作成できるという理由でこれら作業表のリカバリー可能性について心配する必要がない場合、作業表は CREATE TABLE ステートメントに NOT LOGGED INITIALLY パラメーターを指定して作成することができます。これにより、ロギングが低減され、パフォーマンスが改善されます。

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させることができます。 NOT LOGGED INITIALLY パラメーターを指定した ALTER TABLE ステートメントを使用して、既存の表について同じ結果を確保できます。

注:

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

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

- 表に対するすべての 変更項目は、コミット時にディスクに書き出されます。つまり、コミットには時間がかかることがあります。
- NOT LOGGED INITIALLY 属性がアクティブになっており、ログに記録されないアクティビティーが発生した場合、ステートメントが失敗したり、ROLLBACK TO SAVEPOINT が実行された場合には、作業単位全体はロールバックされます (SQL1476N)。
- 高可用性災害時リカバリー (HADR) を使用している場合、NOT LOGGED INITIALLY 表属性を使用すべきではありません。 NOT LOGGED INITIALLY オプションを指定して 1 次データベースに作成された表は、スタンバイ・データベースに複製されません。アクティブ・スタンバイ・データベース上、またはテークオーバー操作の結果スタンバイ・データベースから 1 次データベースになったデータベース上にある、このような表にアクセスしようとすると、エラー (SQL1477N) が発生します。

- ロールフォワード時には、これらの表をリカバリーできません。ロールフォワード操作実行時に NOT LOGGED INITIALLY オプションにより作成または変更された表が検出されると、この表には使用不可のマークが付けられます。データベースのリカバリー後にこの表にアクセスしようとすると、 SQL1477N が戻されます。

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

宣言済み一時表によるロギングの低減

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

- 宣言済み一時表はカタログには記録されません。したがって、ロックは保持されません。
- 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 の後に表の行を維持するため、 ON COMMIT PRESERVE オプションを使用してください。これを行わないと、すべての行が削除されます。
- 宣言済み一時表を作成したアプリケーションだけが、その表のインスタンスにアクセスできます。
- データベースのアプリケーション接続がドロップされるときに、宣言済み一時表は暗黙的にドロップされます。
- アクティブ・スタンバイ・データベースでは、作成済み一時表 (CGTT) および宣言済み一時表 (DGTT) を作成またはアクセスすることはできません。
- 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 (またはセーブポイント) のロールバックでは、その作業単位 (またはセーブポイント) 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。

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

新規ファイルのための十分なスペースがないために DB2 データベース・マネージャーがアクティブのログ・パスにログ・ファイルを作成できない場合、ディスクが満杯であることを示すエラーが出されます。 **blk_log_dsk_ful** データベース構成パラメーターを設定すると、DB2 データベース・マネージャーは「ディスク満杯」エラーを戻す代わりに、ファイルが正常に作成されるまでログ・ファイルの作成を繰り返し試みます。

blk_log_dsk_ful データベース構成パラメーターを設定すると、 DB2 データベース・マネージャーは成功するまで 5 分おきにログ・ファイルの作成を試みます。ログ・アーカイブ・メソッドが指定される場合、DB2 データベース・マネージャーによりログ・ファイルのアーカイブが完了したかチェックされます。アーカイブ・ロ

グ・ファイルが正常にアーカイブされたら、 DB2 データベース・マネージャーにより非アクティブのログ・ファイルの名前を新しいログ・ファイル名に変更して処理を継続できます。試行するたびに、DB2 データベース・マネージャーはメッセージを管理通知ログに書き込みます。ログ・ディスクが満杯になったためにアプリケーションがハングしているかを確認するには、管理通知ログをモニターするしかありません。

ログ・ファイルが正常に作成されるまでは、表データの更新を試行するユーザー・アプリケーションは、トランザクションをコミットすることができません。読み取り専用照会が直接影響を受ける可能性はありませんが、照会が更新要求によってロックされているデータ、または更新アプリケーションによってバッファー・プール内に固定されているデータにアクセスする必要がある場合は、読み取り専用照会もハングするように見えます。

ログ・アーカイブを使用したログ・ファイルの管理

DB2 データ・サーバー・ログ・ファイル・アーカイブは、多様なオペレーティング・システムのファイル処理やスケジューリングの問題によって複雑化しています。例えば、DB2 データベース・マネージャーがログ・ファイルのキューをアーカイブしているときにディスクで障害が発生する場合、そのログ・ファイルおよびそれに含まれるトランザクション・データが失われてしまうこともあり得ます。データベース・ロギングを正しく構成することで、そうした問題によって可用性やリカバリー・ストラテジーが損なわれることを防ぐことができます。

以下の一般的な考慮事項は、すべてのログ・アーカイブのメソッドに当てはまります。

- **logarchcompr1** データベース構成パラメーターでは、データベース・マネージャーが **logarchmeth1** に設定されているロケーションに含まれるログ・ファイルを圧縮するかどうかを指定します。 **logarchmeth1** 構成パラメーターを DISK、TSM、VENDOR 以外の値に設定すると、**logarchcompr1** 構成パラメーターの設定に関係なく、ログ・アーカイブの圧縮の効果はありません。
- **logarchcompr2** データベース構成パラメーターでは、データベース・マネージャーが **logarchmeth2** に設定されているロケーションに含まれるログ・ファイルを圧縮するかどうかを指定します。 **logarchmeth2** 構成パラメーターを DISK、TSM、VENDOR 以外の値に設定すると、**logarchcompr2** 構成パラメーターの設定に関係なく、ログ・アーカイブの圧縮の効果はありません。
- **logarchmeth1** データベース構成パラメーターを指定すると、データベースのロールフォワード・リカバリー中に、指定したメソッドを使用してデータベース・マネージャーによってログ・ファイルがアーカイブされるか、ログ・ファイルがリトリーブされます。ログ・ファイルのリトリーブ要求は、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。ログ・ファイルは、**logpath** 構成パラメーターで指定されているパスからアーカイブされます。

  logarchmeth2 データベース構成パラメーターを指定すると、データベース・マネージャーによってログ・ファイルの追加のコピーがアーカイブされます。ミラー・ロギングを構成すると、**logarchmeth2** パラメーターで指定されているパスにアーカイブされているログ・ファイルが、ミラー・ログのパスから取られます。

ミラー・ロギングを構成しない場合、**logarchmeth2** パラメーターで指定されているパスにアーカイブされているログ・ファイルは、現行のログ・パスから取られます。

- 以下のいずれかの機能を使用している場合には、ログ・ファイルの保管で、ローカル接続された磁気テープ・ドライブを使用するべきではありません。
  - 無限ロギング
  - 表スペース・レベルでのオンライン・リカバリー
  - レプリケーション
  - ログの非同期読み取り API (db2ReadLog)
  - 高可用性災害時リカバリー (HADR)

  上記のどの機能もログ・ファイルのリトリーブ操作を呼び出す場合があり、その際にログ・アーカイブ操作と競合する可能性があります。また、DB2 pureScale環境では、ローカル接続された磁気テープ・ドライブは使用できません。これは、ログのマージ操作を実行するメンバーが、他のメンバーのログをリトリーブする必要があるためです。
- ログ・アーカイブを使用している場合、ログ・マネージャーはアクティブ・ログが満杯になるとそのアーカイブを試みます。場合によっては、ログ・マネージャーがアーカイブの成功を記録する前にデータベースが非活動状態になっていると、ログ・マネージャーはデータベースの活動状態のときにも再びログをアーカイブしようとするかもしれません。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。
- 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、ログ・マネージャーに渡されます。このプロセスにより、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。ログ・マネージャーに渡されたログ・ファイルは、通常の処理に必要なくなるまで、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。
- ログ・ファイルがアーカイブされ、そこにオープン・トランザクションが含まれない場合、DB2 データベース・マネージャー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。このプロセスにより、パフォーマンスが向上します。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場合は、必要なディスク・スペースまたは他のストレージ・スペースが使用可能であることを保証するために、すべてのページを書き出すことになるためです。
- **logsecond** データベース構成パラメーターを -1 (無限ロギングを有効) に設定していない限り、クラッシュ・リカバリー、メンバー・クラッシュ・リカバリー (DB2 pureScale 環境の場合)、またはランタイム・ロールバックの実行中は、DB2 データベース・マネージャーは、ログ・ファイルをリトリーブしません。 DB2 pureScale 環境では、無限ロギングを有効にしていない場合でも、グループ・クラッシュ・リカバリー中に、データベース・マネージャーによるアーカイブ・ログのリトリーブが必要になる場合があります。
- ログ・アーカイブを構成しても、障害の時点までのロールフォワード・リカバリーは保証されませんが、障害期間を短くすることが試行されます。ログ・ファイルが満杯になると、ログ・マネージャーはログを非同期的にアーカイブします。ログ・ファイルが満杯になる前にログのあるディスクで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。

ログ・パスがあるディスクまたは装置の障害によってログ・ファイルが完全に失われてしまうことを防ぐために、*mirrorlogpath* データベース構成パラメーターを使用して、ログを 2 次パスに書き込むことができます。 1 次ディスクまたは装置と一緒に 2 次パスに障害が起きることがない場合、このログ・ファイルを使用してリカバリーできます。

mirrorlogpath と **logarchmeth2** の両方の構成パラメーターを設定すると、**logarchmeth2** 構成パラメーターは、現行のログ・パスにあるログ・ファイルの追加のコピーをアーカイブする代わりに、ミラー・ログ・パスからログ・ファイルをアーカイブします。このログ・アーカイブの振る舞いを利用して、ロールフォワード・リカバリーの際の回復力を向上させることができます。なぜなら、現行のログ・パスにある 1 次ログ・ファイルが、アーカイブの前にディスク障害により破損した場合でも、ミラー・ログ・パスにある使用可能なアーカイブ・ログ・ファイルを使用してデータベースのリカバリー操作を続行できる可能性があるからです。

- ログ・ファイルの構成サイズは、ログ・アーカイブに直接影響します。個々のログ・ファイルが非常に大きい場合、ディスクに障害が起こると、大量のデータが失われることがあります。小さなログ・ファイルを使用するようにデータベースを構成すると、ログ・マネージャーがログをアーカイブする頻度が高くなります。

  しかし、テープなど比較的低速の装置にデータを移動している場合は、比較的大きいサイズのログ・ファイルを使用して、キューが作成されないようにすることもできます。各ファイルのアーカイブで、テープ装置の巻き戻しやアーカイブ・メディアへの接続の確立など、実際のオーバーヘッドが必要になる場合にも、大きいログ・ファイルを使用することをお勧めします。
- ログ・アーカイブを使用している場合、ログ・マネージャーは 1 次ログが満杯になるとそのアーカイブを試みます。満杯になる前にログ・マネージャーがログをアーカイブする場合もあります。これが行われるのは、データベースの非アクティブ化、**ARCHIVE LOG** コマンドの発行、オンライン・バックアップの最後への到達、または **SET WRITE SUSPEND** コマンドの発行によって、ログ・ファイルが切り捨てられる場合です。

  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。
- 磁気テープ・ドライブにログおよびバックアップ・イメージをアーカイブする場合には、バックアップ・イメージとアーカイブ・ログの保存先が、同じ磁気テープ・ドライブにならないようにする必要があります。いくつかのログのアーカイブは、バックアップ操作の進行中に行われる可能性があるため、2 つのプロセスが同時に同じ磁気テープ・ドライブに書き込もうとすると、エラーが起きるかもしれません。

以下の考慮事項が、ログ・ファイルのアーカイブとリトリーブのためのユーザー出口プログラムまたはベンダー・プログラムの呼び出しに適用されます。

- DB2 データベース・マネージャーは、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、オペレーティング・システムによっては、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 オペレーティング・システムなど、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オペレーティング・システムもあります。ログ・ファイルはアーカイブ後も依然としてアクティブで、クラッシュ・リカバリーに必要なので、このようなファイルを決してユーザー出口プログラムで削除してはなりません。ログ・ファイルがアーカイブされると、DB2 データベース・マネージャーによりディスク・スペースの再使用が管理されます。
- 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に、存在しなくなったファイルのアーカイブ要求をユーザー出口プログラムまたはベンダー・プログラムが受け取ることがあります。また、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に、存在しなくなったファイルのリトリーブ要求をユーザー出口プログラムまたはベンダー・プログラムが受け取ることもあります。どちらの場合も、ユーザー出口プログラムまたはベンダー・プログラムはこの要求を無視し、正常な戻りコードを渡す必要があります。
- Windows オペレーティング・システムの場合、REXX ユーザー出口を使用してログをアーカイブすることはできません。
- ユーザー出口プログラムまたはベンダー・プログラムは、ポイント・イン・タイム・リカバリーの後で、同じ名前の別のログ・ファイルが存在できるようにする必要があります。その両方のログ・ファイルを保存し、正しいリカバリー・パスと関連付けるようにユーザー出口プログラムまたはベンダー・プログラムを作成する必要があります。
- ログ・ファイルをアーカイブするために同一の磁気テープ装置を使用している複数のデータベースに対して、ユーザー出口プログラムまたはベンダー・プログラムを使用可能にしており、そのデータベースの 1 つでロールフォワード操作が行われている場合、それ以外のデータベースをアクティブにしてはなりません。ロールフォワード操作が進行中に別のデータベースがログ・ファイルをアーカイブしようとすると、次のいずれかの状況が生じる可能性があります。
  - ロールフォワード操作に必要なログが見つからない可能性があります。
  - 磁気テープ装置にアーカイブされた新しいログ・ファイルが、その磁気テープ装置に以前に保管したログ・ファイルを上書きしてしまう可能性があります。

  上記のどちらの状態も生じないようにするために、以下のいずれかのステップを行うことができます。
  - ユーザー出口プログラムを呼び出すデータベース・パーティション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ことができます。
  - 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。

高可用性のためのクラスター環境の構成

複数のマシンのクラスターを作成して、クラスター管理ソフトウェアを使用してそれらのマシンのワークロードのバランスを取ることは、高可用性ソリューションを設計するための 1 つのストラテジーです。 IBM Data Server をクラスター内の 1 つまたは複数のマシンにインストールした場合、クラスター・マネージャーがデータベース (1 つまたは複数) に影響を与える障害に適切に反応するように構成する必要があります。さらに、データベース・マネージャー・インスタンスがクラスター環境で正しく機能するように構成する必要があります。

このタスクについて

データベース・インスタンスおよびクラスター・マネージャーの手動での構成と管理は、複雑で時間がかかり、誤りを起こしがちです。DB2 高可用性 (HA) フィーチャーは、データベース・マネージャー・インスタンスの停止などのインスタンス構成の変更によってクラスター変更が必要な場合に、データベース・マネージャーがクラスター・マネージャーと通信できるようにするためのインフラストラクチャーを提供します。

手順

1. クラスター管理ソフトウェアをインストールします。

   AIX、Linux、および Solaris SPARC オペレーティング・システムでは、SA MP は DB2 Enterprise Server Edition、DB2 Advanced Enterprise Server Edition、DB2 Workgroup Server Edition、 DB2 Connect Enterprise Edition、および DB2 Connect Application Server Edition と統合されています。Linux オペレーティング・システムでは、DB2 Express-C Fixed Term License (FTL) および IBM DB2 High Availability Feature for Express® Edition とも統合されます。Windows オペレーティング・システムでは、SA MP が上記のすべての DB2 データベース製品およびフィーチャーにバンドルされていますが、DB2 インストーラーとは統合されていません。

2. クラスター・マネージャー用に IBM Data Server データベース・マネージャー・インスタンスを構成し、IBM Data Server 用にクラスター・マネージャーを構成します。

   DB2 高可用性インスタンス構成ユーティリティー (db2haicu) は、クラスター環境の高可用性データベースを構成および管理するために使用できるテキスト・ベースのユーティリティーです。 **db2haicu** は、データベース・インスタンス、クラスター環境、およびクラスター・マネージャーに関する情報を、システムを照会して収集します。ユーザーは、**db2haicu** 呼び出しへのパラメーターおよび入力ファイルを介して、または実行時に **db2haicu** プロンプトで、詳細情報を提供します。

3. 時間が経過するにつれて、データベースが変更を必要とし、クラスター環境内でデータベースの構成を変更する必要があるので、データベース・マネージャー・インスタンスの構成とクラスター・マネージャーの構成を同期し続けます。

   DB2 高可用性 (HA) フィーチャーは、データベース・マネージャー・インスタンスの停止などのインスタンス構成の変更によってクラスター変更が必要な場合

に、データベース・マネージャーがクラスター・マネージャーと通信できるようにするためのインフラストラクチャーを提供します。

SA MP とともに **db2haicu** を使用するか、または DB2 クラスター・マネージャー API をサポートする他のクラスター・マネージャーを使用するかにかかわらず、DB2 HA フィーチャーを持つクラスター環境の管理は、データベース・マネージャー構成およびクラスター構成を個別に保守するより容易です。

DB2 高可用性 (HA) フィーチャーとクラスター・マネージャーの統合

DB2 高可用性 (HA) フィーチャーによって、IBM Data Server とクラスター管理ソフトウェアの統合が可能になります。

クラスター環境でデータベース・マネージャー・インスタンスを停止する場合、当該インスタンスの停止をクラスター・マネージャーに知らせる必要があります。クラスター・マネージャーは、当該インスタンスの停止を知らない場合、停止したインスタンスにフェイルオーバーなどの操作を試みる可能性があります。 DB2 高可用性 (HA) フィーチャーは、データベース・マネージャー・インスタンスの停止などのインスタンス構成の変更によってクラスター変更が必要な場合に、データベース・マネージャーがクラスター・マネージャーと通信できるようにするためのインフラストラクチャーを提供します。

DB2 HA フィーチャー は、以下のエレメントで構成されています。

- IBM Tivoli System Automation for Multiplatforms (SA MP) は、AIX および Linux 上の IBM Data Server に DB2 高可用性 (HA) フィーチャーの一部としてバンドルされており、DB2 インストーラーに統合されています。DB2 インストーラーまたは IBM Data Server のインストール・メディアに組み込まれている **installSAM** および **uninstallSAM** スクリプトを使用して、SA MP をインストール、アップグレード、またはアンインストールすることができます。
- クラスター化された環境では、一部のデータベース・マネージャーのインスタンス構成および管理操作で、関連するクラスター構成の変更が必要となります。DB2 High Availability Feature (HA) フィーチャーは、特定のデータベース・マネージャーのインスタンス構成および管理操作を実行する度に、データベース・マネージャーがクラスター・マネージャーの構成変更を自動的に要求できるようにします。 99 ページの『DB2 高可用性 (HA) フィーチャーを使用したクラスターの自動構成』 を参照。
- DB2 高可用性インスタンス構成ユーティリティー (db2haicu) は、クラスター環境の高可用性データベースを構成および管理するために使用できるテキスト・ベースのユーティリティーです。 **db2haicu** は、データベース・インスタンス、クラスター環境、およびクラスター・マネージャーに関する情報を、システムを照会して収集します。ユーザーは、**db2haicu** 呼び出しへのパラメーターおよび入力ファイルを介して、または実行時に **db2haicu** プロンプトで、詳細情報を提供します。 110 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu)』 を参照。
- DB2 クラスター・マネージャー API は、データベース・マネージャーが構成変更をクラスター・マネージャーに通信するために必要な関数のセットを定義します。 150 ページの『DB2 クラスター・マネージャー API』 を参照。

IBM Tivoli System Automation for Multiplatforms (SA MP) Base Component

IBM Tivoli System Automation for Multiplatforms (SA MP) Base Component は、AIX、Linux、 Solaris SPARC、および Windows における高可用性機能および災害時リカバリー機能を提供します。

SA MP は、AIX、Linux、および Solaris SPARC オペレーティング・システム上で DB2 Enterprise Server Edition、DB2 Advanced Enterprise Server Edition、DB2 Workgroup Server Edition、DB2 Connect Enterprise Edition および DB2 Connect Application Server Edition と統合されます。 DB2 Express-C Fixed Term License (FTL) および DB2 High Availability Feature と一緒に使用するために、Express Edition とも統合されています。

Windows オペレーティング・システムでは、SA MP がそれらのすべての DB2 データベース製品およびフィーチャーとバンドルされていますが、DB2 データベース製品インストーラーとは統合されていません。

このコピーの SA MP を使用して、DB2 データベース・システムの高可用性を管理することができます。SA MP ライセンスのアップグレードを購入しないと、このコピーを使用して DB2 データベース・システム以外のデータベース・システムを管理することはできません。

SA MP は、AIX、Linux、および Solaris SPARC オペレーティング・システム上の IBM データ・サーバー・クラスター環境のデフォルトのクラスター・マネージャーです。

SA MP の詳細については、IBM Tivoli System Automation for Multiplatforms (SA MP)publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms3.1.html を参照してください。サポートされているオペレーティング・システムのリストは、Web サイト www.ibm.com/software/tivoli/products/sys-auto-linux/platforms.html でも確認できます。

DB2 高可用性 (HA) フィーチャーを使用したクラスターの自動構成

クラスター化された環境では、一部のデータベース・マネージャーのインスタンス構成および管理操作で、関連するクラスター構成の変更が必要となります。DB2 High Availability Feature (HA) フィーチャーは、特定のデータベース・マネージャーのインスタンス構成および管理操作を実行する度に、データベース・マネージャーがクラスター・マネージャーの構成変更を自動的に要求できるようにします。

始める前に

データベース・マネージャーがデータベース管理タスクに必要なクラスター構成を実行できるようにするには、**db2haicu** を使用して高可用性のインスタンスを構成し、インスタンスのクラスター・ドメインを作成する必要があります。詳しくは、101 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用したクラスター化環境の構成』を参照してください。

手順

以下のデータベース・マネージャーのインスタンス構成および管理操作を実行するとき、データベース・マネージャーは関連するクラスター・マネージャーの構成を自動的に実行します。

- **START DATABASE** または **db2start** を使用したデータベースの開始。
- **STOP DATABASE** または **db2stop** を使用したデータベースの停止。
- **CREATE DATABASE** を使用したデータベースの作成。
- **CREATE TABLESPACE** を使用したストレージの追加。
- **ALTER TABLESPACE DROP** または **DROP TABLESPACE** を使用したストレージの除去。
- ALTER DATABASE を使用したストレージ・パスの追加または除去。
- **DROP TABLESPACE** を使用したデータベースのドロップ。
- **RESTORE DATABASE** または db2Restore を使用したデータベースのリストア。
- **SET TABLESPACE CONTAINERS** を使用したリダイレクト・リストアの表スペース・コンテナーの指定。
- **ROLLFORWARD DATABASE** または db2Rollforward を使用したデータベースのロールフォワード。
- **RECOVER DATABASE** または db2Recover を使用したデータベースのリカバリー。
- **CREATE EVENT MONITOR** を使用したイベント・モニターの作成。
- **DROP EVENT MONITOR** を使用したイベント・モニターのドロップ。
- 以下を使用した外部ルーチンの作成および変更
  - CREATE PROCEDURE
  - CREATE FUNCTION
  - CREATE FUNCTION
  - CREATE METHOD
  - ALTER PROCEDURE
  - ALTER FUNCTION
  - ALTER METHOD
- 以下を使用した外部ルーチンのドロップ
  - DROP PROCEDURE
  - DROP FUNCTION
  - DROP METHOD
- **START HADR** を使用したデータベースの DB2 高可用性災害時リカバリー (HADR) 操作の開始。
- **STOP HADR** を使用したデータベースの HADR 操作の停止。
- **TAKEOVER HADR** を使用した HADR スタンバイ・データベースの HADR 1 次データベースとしてのテークオーバー。
- データベース・マネージャー構成パラメーター **diagpath** または **spm_log_path** の設定。
- データベース構成パラメーター **newlogpath**、**overflowlogpath**、**mirrorlogpath**、または **failarchpath** の設定。
- **db2idrop** を使用したデータベース・マネージャー・インスタンスのドロップ。

タスクの結果

データベース・マネージャーが、リストされているデータベース管理タスクのクラスター構成の変更を調整するとき、別個のクラスター・マネージャー操作を実行する必要はありません。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用したクラスター化環境の構成

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用して、クラスター化された環境でデータベースを構成および管理することができます。データベース・マネージャーのインスタンス構成の詳細を **db2haicu** に指定するとき、**db2haicu** は必要なクラスター構成の詳細をクラスター管理ソフトウェアに通知します。

始める前に

- DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用する前に実行しなければない一連のタスクがあります。詳しくは、144 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の前提条件』を参照してください。

このタスクについて

以下に示すように、**db2haicu** を対話式に、または XML 入力ファイルを使用して実行することができます。

対話モード

-f パラメーターで XML 入力ファイルを指定せずに、**db2haicu** コマンドを実行して DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を呼び出すとき、ユーティリティーは対話モードで実行されます。対話モードでは、**db2haicu** は情報をテキスト・ベース形式で表示し、情報についてユーザーに照会します。 詳しくは、112 ページの『対話モードでの DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の実行』を参照してください。

XML 入力ファイルを使用したバッチ・モード

-f *input-file-name* パラメーターを **db2haicu** コマンドで使用して、構成の詳細を指定する XML 入力ファイルを使って DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を実行することができます。複数のデータベース・パーティションを高可用性として構成するときなど、構成タスクを複数回実行する必要があるときに、XML 入力ファイルを使用した **db2haicu** の実行は便利です。 詳しくは、113 ページの『XML 入力ファイルを使用した DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の実行』を参照してください。

制約事項

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用する場合、いくつかの制約事項があります。詳しくは、148 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の制約事項』を参照してください。

手順

各データベース・マネージャー・インスタンスごとに以下のステップを実行します。

1. 新規クラスター・ドメインの作成

   DB2 高可用性インスタンス構成ユーティリティー (db2haicu) をデータベース・マネージャー・インスタンスに対して初めて実行するとき、**db2haicu** はクラスター・ドメインと呼ばれる、クラスターのモデルを作成します。 詳しくは、146 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用したクラスター・ドメインの作成』を参照してください。

2. クラスター・ドメインの構成の改善の続行、およびクラスター・ドメインの管理と保守

   db2haicu を使用して、クラスター化した環境のクラスター・ドメイン・モデルを変更する場合、データベース・マネージャーは関連する変更を、データベース・マネージャー・インスタンスおよびクラスター構成に伝搬させます。 詳しくは、146 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用したクラスター・ドメインの保守』を参照してください。

次のタスク

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) には、別個の診断ログがありません。データベース・マネージャーの診断ログ **db2diag** ログ・ファイル、および **db2pd** ツールを使用して、**db2haicu** のエラーを調査、および診断することができます。 詳しくは、148 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) のトラブルシューティング』を参照してください。

クラスター・ドメイン

クラスター・ドメインは、データベース、マウント・ポイント、およびフェイルオーバー・ポリシーなどのクラスター・エレメントに関する情報が入っているモデルです。クラスター・ドメインは DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用して作成します。**db2haicu** はクラスター・ドメイン内の情報を使用して、構成および保守のクラスター管理タスクを使用可能にします。データベース・マネージャーも、DB2 高可用性 (HA) フィーチャーの一部として、クラスター・ドメイン内の情報を使用して自動クラスター管理タスクを実行します。

クラスター・ドメインにクラスター・エレメントを追加した場合、当該エレメントは、DB2 HA フィーチャーの一部としてデータベース・マネージャーによって実行される以後のすべての **db2haicu** 構成操作、またはすべての自動クラスター管理操作に組み込まれます。クラスター・ドメインからクラスター・エレメントを除去した場合、当該エレメントは、**db2haicu** 操作またはデータベース・マネージャーの自動クラスター管理操作に組み込まれなくなります。 **db2haicu** およびデータベース・マネージャーは、**db2haicu** を使用して作成されるクラスター・ドメイン内のクラスター・エレメントに関してのみ、クラスター・マネージャーと調整できます。

db2haicu を使用して、以下のクラスター・ドメイン・エレメントを作成および構成できます。

- コンピューターまたはマシン (クラスター・ドメイン・コンテキストではクラスター・*ドメイン・ノード* と呼ばれる)
- ネットワーク・インターフェース・カード (NIC) (**db2haicu** ではネットワーク・インターフェース、インターフェース、ネットワーク・アダプター、またはアダプター と呼ばれる)
- IP アドレス
- データベース (高可用性災害時リカバリー (HADR) のための 1 次データベースとスタンバイ・データベースのペアを含む)
- データベース・パーティション
- マウント・ポイントおよびパス (障害発生時のフェイルオーバーにとって不可欠でないパスを含む)
- フェイルオーバー・ポリシー
- クォーラム装置

クラスター管理ソフトウェア:

クラスター管理ソフトウェアは、コンピューター・クラスターが実行できる作業を最大限に引き出します。クラスター・マネージャーは、ワークロードのバランスを取ってボトルネックを減らし、クラスターのエレメントの正常性をモニターし、エレメントに障害が発生したときはフェイルオーバーを管理します。クラスター・マネージャーは、システム管理者がクラスター内のエレメントに対して管理タスクを実行するときにも役立ちます (例えば、保守の必要なコンピューターからワークロードを転送します)。

クラスターのエレメント

クラスター・マネージャーが適切に機能するためには、クラスターのエレメントに関連したさまざまな詳細情報と、クラスターのエレメント間の関係を、クラスター・マネージャーが認識しておかなければなりません。

クラスター・マネージャーが認識しておく必要があるクラスター・エレメントの例をいくつか示します。

- クラスター内の物理または仮想コンピューター、マシン、デバイス (クラスター・コンテキストではクラスター・*ノード* と呼ばれる)
- クラスター・ノードを接続するネットワーク
- クラスター・ノードをネットワークに接続するネットワーク・インターフェース・カード
- クラスター・ノードの IP アドレス
- 仮想またはサービス IP アドレス

クラスター・マネージャーが認識しておく必要がある関係の例をいくつか示します。

- 同じソフトウェアがインストールされていて相互にフェイルオーバーできるクラスター・ノードのペア
- プロパティーが同じで相互のフェイルオーバーに使用できるネットワーク
- 仮想 IP アドレスが現在関連付けられているクラスター・ノード

クラスター・エレメントの追加または変更

クラスターのエレメントとエレメント間の関係をクラスター・マネージャーに認識させるには、システム管理者がエレメントをクラスター・マネージャーに登録する必要があります。システム管理者は、クラスターのエレメントに変更を加えた場合は、その変更をクラスター・マネージャーに伝える必要があります。クラスター・マネージャーには、こうしたタスクのためのインターフェースがあります。

クラスター・エレメントとして実にさまざまなものが存在しうるので、クラスター管理は難しい作業です。管理者は、クラスター・ノードのハードウェアとオペレーティング・システム、ネットワーキングのプロトコルと構成、クラスター・ノードにインストールされているデータベース・ソフトウェアなどのソフトウェアについて、熟知しておく必要があります。クラスターのエレメントをクラスター管理ソフトウェアに登録したり、システム変更後にクラスター・マネージャーを更新したりする作業は、手間と時間がかかる場合があります。

db2haicu を使用してクラスター・エレメントを追加または変更する

DB2 データベース・ソリューションでは、DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用することにより、クラスターのエレメントをクラスター・マネージャーに登録でき、クラスターに管理上の変更を加えた後にクラスター・マネージャーを更新できます。**db2haicu** を使用すると、こうしたタスクが簡単になります。**db2haicu** が使用するモデルを理解してクラスターのエレメントとエレメント間の関係をカプセル化してしまえば、タスクを実行するうえでは、ハードウェア、オペレーティング・システム、クラスター・マネージャー・インターフェースの特異性を熟知しておく必要がないからです。

リソースおよびリソース・グループ:

リソース とは、クラスター・ノード、データベース、マウント・ポイント、ネットワーク・インターフェース・カードなど、クラスター・マネージャーに登録されている何らかのクラスター・エレメントのことです。クラスター・マネージャーに登録されていないエレメントは、クラスター・マネージャーに認識されないので、クラスター管理操作に含まれません。*リソース・グループ* は、リソースの論理コレクションです。リソース・グループは非常に強力な構成体です。リソース・グループの関係と制約を定義できるので、そうしたグループ内のリソースに対して複雑な管理タスクを簡単に実行できるからです。

リソースをグループに収集したクラスター・マネージャーは、それらのすべてのリソースに対してまとめて操作できるようになります。例えば、`resource-group-A` というリソース・グループに `database-1` と `database-2` の 2 つのデータベースが属している場合、クラスター・マネージャーが `resource-group-A` に対して開始操作を実行すると、`database-1` と `database-2` の両方がクラスター・マネージャーのこの 1 つの操作で開始されます。

制約事項

- リソース・グループに*等価リソース* を含めることはできません。また、等価リソースにリソース・グループを含めることもできません。(*等価リソース* とは、それぞれが同じ機能を提供し、相互にフェイルオーバーできるリソースのセットのことです。)

- リソースは 1 つのリソース・グループにのみ含めることができます。
- リソースをリソース・グループと等価リソースの両方に含めることはできません。
- リソース・グループに他のリソース・グループを含めることができますが、最大ネスト・レベルは 50 です。
- リソース・グループに集めることができるリソースの最大数は 100 です。

クォーラム装置:

クォーラム装置 は、クラスター・マネージャーがクラスター管理上の決定を行うときに、通常の決定プロセスでは明確に選択できない場合に役立ちます。クラスター・マネージャーが、可能性のある複数のアクションから選択する必要があるときは、可能な各アクションをサポートするクラスター・ドメイン・ノードの数をカウントし、サポートしているクラスター・ドメイン・ノードが最も多いアクションを選択します。クラスター・ドメイン・ノードの数がまったく同じであるために複数の選択肢がサポートされる場合、クラスター・マネージャーはクォーラム装置を参照して選択を行います。

db2haicu は、次の表に示すクォーラム装置をサポートします。

表 2. **db2haicu** *がサポートするクォーラム装置のタイプ*

クォーラム装置	説明
network	network クォーラム装置とは、どのクラスター・ドメイン・ノードもいつでも接続できる IP アドレスのことです。

クラスター・ドメインにおけるネットワーク:

クラスター・ドメインのネットワーク関連エレメントを構成するには、DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用してクラスター・ドメインに*物理ネットワーク* を追加します。物理ネットワークは、ネットワーク・インターフェース・カード、IP アドレス、サブネットワーク・マスクで構成されます。

ネットワーク・インターフェース・カード

ネットワーク・インターフェース・カード (NIC) は、コンピューター (*クラスター・ノード* とも呼ばれる) をネットワークに接続するためのハードウェアです。NIC は、*インターフェース* やネットワーク・アダプター、あるいは*アダプター* と呼ばれることもあります。**db2haicu** を使用してクラスター・ドメインに物理ネットワークを追加するときに、NIC を少なくとも 1 つ以上指定します。この場合、NIC が属するコンピューターのホスト名、そのホスト・コンピューターでの NIC の名前、NIC の IP アドレスを指定します。

IP アドレス

インターネット・プロトコル・アドレス (IP アドレス) は、ネットワーク上の固有のアドレスです。IP バージョン 4 では、IP アドレスは 32 ビットの長さを持ち、通常は `129.30.180.16` のようにドット 10 進表記で表されます。IP アドレスは、ネットワーク部分とホスト・コンピューター部分で構成されます。

db2haicu は IP バージョン 6 をサポートしていません。

サブネットワーク・マスク

サブネットワーク・マスク を使用することにより、ネットワークを複数の論理サブネットワークに分割できます。サブネットワーク・マスクは、IP アドレスのホスト部分のいくつかのビットを IP アドレスのネットワーク部分に移すための手段です。**db2haicu** を使用してクラスター・ドメインに IP アドレスを追加するときに、IP アドレスのサブネットワーク・マスクを指定しなければならない場合もあります。例えば、**db2haicu** を使用して NIC を追加するときに、NIC の IP アドレスのサブネットワーク・マスクを指定する必要があります。

ネットワーク等価リソース

等価リソース とは、それぞれが同じ機能を提供し、相互にフェイルオーバーできるリソースのセットのことです。**db2haicu** を使用してネットワークを作成すると、そのネットワーク内の NIC は相互にフェイルオーバーできます。このようなネットワークは、*ネットワーク等価リソース* とも呼ばれます。

ネットワーク・プロトコル

db2haicu を使用してクラスター・ドメインにネットワークを追加するときに、使用されるネットワーク・プロトコルのタイプを指定する必要があります。現在は、TCP/IP ネットワーク・プロトコルのみがサポートされています。

使用上の注意

db2haicu を使用して構成されたネットワークが必要なのは、仮想 IP (VIP) フェイルオーバーの場合のみです。別々のサブネット (つまり別々の仮想ローカル・エリア・ネットワーク) にあるネットワーク・アダプターを、同じネットワークに追加することはできません。 VIP フェイルオーバーには共通仮想ローカル・エリア・ネットワークが必要だからです。

クラスター・ドメインにおけるフェイルオーバー・ポリシー:

フェイルオーバー・ポリシー は、ネットワーク・インターフェース・カードやデータベース・サーバーなどのクラスター・エレメントに障害が発生したときにクラスター・マネージャーがどのように対処するかを指定したものです。一般に、クラスター・マネージャーは障害の起きたエレメントから代替エレメントにワークロードを転送します。この代替エレメントは、障害の起きたエレメントに適合する代替としてあらかじめクラスター・マネージャーに示されています。障害の起きたエレメントから 2 次エレメントへのこうしたワークロード転送を、*フェイルオーバー* といいます。

ラウンドロビン・フェイルオーバー・ポリシー

ラウンドロビン・フェイルオーバー・ポリシー を使用すると、クラスター・ドメイン内のあるコンピューター (*クラスター・ドメイン・ノード* または単に*ノード* とも呼ばれる) に関連した障害が発生すると、データベース・マネージャーは障害の起きたクラスター・ドメイン・ノードでの作業をクラスター・ドメイン内の他のノードで再開します。

相互フェイルオーバー・ポリシー

相互フェイルオーバー・ポリシー を構成するには、クラスター・ドメイン内のコンピューター (*クラスター・ドメイン・ノード* または単に*ノード* とも呼ばれる) のペアをシステム・ペアとして関連付けます。このペアの一方のノードに障害が発生すると、障害の起きたノード上のデータベース・パーティションは、ペアのもう一方のノードにフェイルオーバーします。相互フェイルオーバーは、複数のデータベース・パーティションがある場合のみ使用できます。

N プラス M フェイルオーバー・ポリシー

N プラス M フェイルオーバー・ポリシー を使用すると、クラスター・ドメイン内のあるコンピューター (*クラスター・ドメイン・ノード* または単に*ノード* とも呼ばれる) に関連した障害が発生すると、障害の起きたノード上のデータベース・パーティションは、クラスター・ドメイン内の他のいずれかのノードにフェイルオーバーします。ロービング HA フェイルオーバーを有効にした場合、最後に障害の起きたノードが再びオンラインになると、そのノードがスタンバイ・ノードになります。 M=1 の場合は、N プラス M フェイルオーバー・ポリシーのロービング HA フェイルオーバーのみがサポートされます。 N プラス M フェイルオーバーは、複数のデータベース・パーティションがある場合のみ使用できます。

ローカル再開フェイルオーバー・ポリシー

ローカル再開フェイルオーバー・ポリシー を使用すると、クラスター・ドメイン内のコンピューター (*クラスター・ドメイン・ノード* または単に*ノード* とも呼ばれる) の一つで障害が発生すると、データベース・マネージャーは障害の起きたそのノード上で (つまりローカルに) データベースを再開します。

HADR フェイルオーバー・ポリシー

HADR フェイルオーバー・ポリシー を構成する場合は、DB2 高可用性災害時リカバリー (HADR) フィーチャーでフェイルオーバーを管理できるようにすることになります。HADR 1 次データベースに障害が起こると、データベース・マネージャーは障害の起きたデータベースから HADR スタンバイ・データベースにワークロードを移動します。

カスタム・フェイルオーバー・ポリシー

カスタム・フェイルオーバー・ポリシー を構成する場合は、クラスター・ドメイン内のコンピューター (*クラスター・ドメイン・ノード* または単に*ノード* とも呼ばれる) のうち、データベース・マネージャーがフェイルオーバー先として使用できるコンピューターのリストを作成します。クラスター・ドメイン内のノードに障害が起こると、データベース・マネージャーは障害の起きたノードから、指定されたリスト内のノードの一つにワークロードを移動します。

パーティション・データベース環境でのロービング高可用性 (HA) フェイルオーバーの使用:

アクティブ・ノードが「N」個でスタンバイ・ノードが 1 個の環境で N プラス M フェイルオーバー・ポリシーを使用する場合は、ロービング HA フェイルオーバーを使用可能にすることができます。

始める前に

クラスター内の各ノードは、ロービング HA フェイルオーバー・サポートが使用可能または使用不可になっていなければなりません。

ロービング HA フェイルオーバーが使用可能になっていないパーティション・データベース環境では、すべてのディスクおよびボリューム・グループ (これらのボリューム・グループ上のファイル・システムを含む) にアクセスできるノードは通常、指定されたスタンバイ・ノードだけです。このような環境では、クラスター内の外部ストレージ LUN マッピングと SAN ゾーンからデータベース・インスタンス内のすべてのディスクが可視になるようにしておきます。さらに、クラスターによって制御されるすべてのボリューム・グループが、すべてのクラスター・ノードにインポートされていることを確認します。ボリューム・グループのインポート後、すべてのアクティブ・クラスター・ノードのボリューム・グループの auto-varyon 属性とファイル・システムの auto-mount 属性を使用不可にします。

ロービング HA フェイルオーバーを使用する場合は、新しいフィックスパックを適用した後に、ここに記載したステップを実行して、ロービング HA フェイルオーバーを再度使用可能にする必要があります。

このタスクについて

アクティブ・ノードが「N」個でスタンバイ・ノードが 1 個の環境で *N* プラス *M* フェイルオーバー・ポリシーを使用する場合は、いずれかのアクティブ・ノードに障害が発生するとフェイルオーバー操作が行われます。次にスタンバイ・ノードが、障害の起きたノードのリソースのホスティングを開始します。障害の起きたノードがオンラインに戻ると、もともとスタンバイ・ノードとして選択されていたノードが再びスタンバイ・ノードになるように、通常はクラスター環境を再度オフラインにする必要があります。追加のフェイルバック操作の必要なしにクラスター内の最後に障害が発生したノードがスタンバイ・ノードになるように、ロービング HA フェイルオーバーを構成できるようになりました。

手順

ロービング HA フェイルオーバーを使用可能にするには、以下のようにします。

1. 進行中のフェイルオーバー操作がないことを確認します。
2. sqllib¥samples¥tsa ディレクトリーにある db2V10_start.ksh スクリプトのバックアップ・コピーを作成します。
3. db2V10_start.ksh スクリプトを編集します。 次の行を見つけます。

   ```
   ROVING_STANDBY_ENABLED=false
   ```

   次のように変更します。

   ```
   ROVING_STANDBY_ENABLED=true
   ```
4. 変更内容を保存します。

タスクの結果

この変更は、次回のフェイルオーバー操作時に有効になります。

次のタスク

ロービング HA フェイルオーバー・サポートを使用不可にする場合は、各ノードで以下のステップを実行します。

1. 進行中のフェイルオーバー操作がないことを確認します。
2. db2V10_start.ksh スクリプトを編集します。次の行を見つけます。

   ```
   ROVING_STANDBY_ENABLED=true
   ```

   次のように変更します。

   ```
   ROVING_STANDBY_ENABLED=false
   ```
3. 変更内容を保存します。この変更は、次回のフェイルオーバー操作時に有効になります。

クラスター・ドメインにおけるマウント・ポイント:

ファイル・システムをマウントした後、DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用してそのマウント・ポイントをクラスター・ドメインに追加できます。

マウント・ポイント

UNIX、Linux、および AIX オペレーティング・システムでは、ファイル・システムをマウント することは、そのファイル・システムをオペレーティング・システムが使用できるようにすることを意味します。オペレーティング・システムは、マウント操作時に索引やナビゲーション・データ構造の読み取りなどのタスクを実行し、マウントされるそのファイル・システムにディレクトリー・パスを関連付けます。関連付けられたこのディレクトリー・パスをマウント・ポイント といい、マウントされたファイル・システムにアクセスするときに使用できます。

不可欠でないマウント・ポイントまたはパス

障害が発生してもフェイルオーバーされる必要のないマウント・ポイントまたはパスがクラスターに含まれている場合があります。**db2haicu** を使用して、こうした不可欠でないマウント・ポイントまたはパスのリストをクラスター・ドメインに追加できます。クラスター・マネージャーは、この非不可欠リストに含まれるマウント・ポイントまたはパスを、フェイルオーバー操作に組み込みません。

例えば、クラスター内の node1 というコンピューターの /mnt/driveA にハード・ディスクがマウントされているとします。/mnt/driveA が使用可能であることが不可欠であるとした場合、node1 に障害が発生すると、クラスター・マネージャーは /mnt/driveA の使用可能状態を維持するフェイルオーバーを行います。一方、node1 に障害が発生したときは /mnt/driveA を使用できなくてもよいとする場合は、不可欠でないパスのリストに /mnt/driveA を追加することによって、フェイルオーバーにとって /mnt/driveA は不可欠でないことをクラスター・マネージャーに示すことができます。/mnt/driveA はフェイルオーバーにとって不可欠でないことを指定すると、node1 に障害が発生した場合は、このドライブを使用できなくなる可能性があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu)

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) は、クラスター環境の高可用性データベースを構成および管理するために使用できるテキスト・ベースのユーティリティーです。 **db2haicu** は、データベース・インスタンス、クラスター環境、およびクラスター・マネージャーに関する情報を、システムを照会して収集します。ユーザーは、**db2haicu** 呼び出しへのパラメーターおよび入力ファイルを介して、または実行時に **db2haicu** プロンプトで、詳細情報を提供します。

構文

```
db2haicu [ -f XML-input-file-name ]
         [ -disable ]
         [ -delete [ dbpartitionnum db-partition-list |
                     hadrdb database-name ] ]
```

パラメーター

db2haicu コマンドに渡すパラメーターには大文字と小文字の区別があり、小文字でなければなりません。

-f *XML-input-file-name*

-f パラメーターを使用すれば、XML 入力ファイル *XML-input-file-name* にクラスター・ドメインの詳細を指定できます。詳しくは、 113 ページの『XML 入力ファイルを使用した DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の実行』を参照してください。

-disable

データベース・マネージャー・インスタンスは、**db2haicu** を使用してインスタンスのクラスター・ドメインを作成すると、*高可用性として構成されている*と見なされます。データベース・マネージャー・インスタンスが高可用性として構成された場合、関連するクラスター構成変更が必要な特定のデータベース・マネージャー管理操作を実行するたびに、データベース・マネージャーはこれらのクラスター構成変更をクラスター・マネージャーに通信します。データベース・マネージャーがこれらのクラスター管理タスクについてクラスター・マネージャーと調整するので、ユーザーがこのような管理タスクのための別個のクラスター管理操作を実行する必要はありません。データベース・マネージャーとクラスター・マネージャーのこのような統合が、DB2 高可用性 (HA) フィーチャーの機能の 1 つです。

-disable パラメーターを使用すれば、データベース・マネージャー・インスタンスが高可用性として構成されるのを中止できます。データベース・マネージャー・インスタンスの高可用性としての構成が中止された場合、関連するクラスター構成変更が必要なデータベース・マネージャー管理操作を実行しても、データベース・マネージャーはクラスター・マネージャーと調整しません。

データベース・マネージャー・インスタンスを高可用性について再構成するには、**db2haicu** を再び実行します。

-delete

-delete パラメーターを使用すれば、現行データベース・マネージャー・インスタンスのリソース・グループを削除できます。

dbpartitionnum パラメーターも **hadrdb** パラメーターも使用しない場合、**db2haicu** は、現行データベース・マネージャー・インスタンスに関連するすべてのリソース・グループを削除します。

dbpartitionnum *db-partition-list*
dbpartitionnum パラメーターを使用すれば、*db-partition-list* にリストされているデータベース・パーティションに関連するリソース・グループを削除できます。 *db-partition-list* は、データベース・パーティションを識別する数値のコンマ区切りリストです。

hadrdb *database-name*
hadrdb パラメーターを使用すれば、*database-name* という名前の DB2 高可用性災害時リカバリー (HADR) データベースに関連するリソース・グループを削除できます。

db2haicu がリソース・グループを削除した後、クラスター・ドメインにリソース・グループが残っていない場合、**db2haicu** はクラスター・ドメインも削除します。

-delete パラメーターを指定して **db2haicu** を実行した場合、現行データベース・マネージャー・インスタンスの高可用性としての構成が中止されます。データベース・マネージャー・インスタンスの高可用性としての構成が中止された場合、関連するクラスター構成変更が必要なデータベース・マネージャー管理操作を実行しても、データベース・マネージャーはクラスター・マネージャーと調整しません。

データベース・マネージャー・インスタンスを高可用性について再構成するには、**db2haicu** を再び実行します。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の始動モード:

特定のデータベース・マネージャー・インスタンスに対する DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の最初の実行時には、**db2haicu** は始動モードで作動します。

db2haicu を実行すると、**db2haicu** は、データベース・マネージャー・インスタンスおよびシステム構成を調べ、既存のクラスター・ドメインを検索します。クラスター・ドメインは、データベース、マウント・ポイント、およびフェイルオーバー・ポリシーなどのクラスター・エレメントに関する情報が入っているモデルです。クラスター・ドメインは DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用して作成します。**db2haicu** はクラスター・ドメイン内の情報を使用して、構成および保守のクラスター管理タスクを使用可能にします。データベース・マネージャーも、DB2 高可用性 (HA) フィーチャーの一部として、クラスター・ドメイン内の情報を使用して自動クラスター管理タスクを実行します。

特定のデータベース・マネージャー・インスタンスに関して **db2haicu** を実行するときに、当該インスタンス用に作成および構成されたクラスター・ドメインがまだない場合には、**db2haicu** は新規のクラスター・ドメインの作成および構成プロセスを直ちに開始します。 **db2haicu** は、新規クラスター・ドメインの名前および現行マシンのホスト名などの情報の入力をユーザーに指示して、新規クラスター・ドメインを作成します。

クラスター・ドメインを作成したが、クラスター・ドメインを構成するタスクを完了しなかった場合、次回 **db2haicu** を実行するときに、**db2haicu** はクラスター・ドメインを構成するタスクを再開します。

データベース・マネージャー・インスタンス用のクラスター・ドメインの作成および構成後、**db2haicu** は保守モードで実行します。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の保守モード:

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を実行中で、現行データベース・マネージャー・インスタンス用に作成されたクラスター・ドメインがすでに存在する場合、**db2haicu** は保守モードで作動します。

保守モードで実行中の **db2haicu** は、実行可能な構成タスクおよび管理タスクのリストを提供します。

db2haicu 保守タスクには、データベースやクラスター・ノードなどのクラスター・エレメントをクラスター・ドメインに追加することや、クラスター・ドメインからエレメントを除去することが含まれています。**db2haicu** 保守タスクには、データベース・マネージャー・インスタンスのフェイルオーバー・ポリシーなどのクラスター・ドメインのエレメントの詳細を変更することも含まれています。

保守モードで **db2haicu** を実行するとき、以下のような **db2haicu** はクラスター・ドメインで実行できる操作のリストを示します。

- クラスター・ノード (ホスト名によって識別されるマシン) の追加または除去
- ネットワーク・インターフェース (ネットワーク・インターフェース・カード) の追加または除去
- データベース・パーティション (パーティション・データベース環境のみ) の追加または除去
- DB2 高可用性災害時リカバリー (HADR) データベースの追加または除去
- 高可用性データベースの追加または除去
- マウント・ポイントの追加または除去
- IP アドレスの追加または除去
- クリティカルでないパスの追加または除去
- 定期保守用データベース・パーティションおよび HADR データベースの移動
- 現行のインスタンスのフェイルオーバー・ポリシーの変更
- クラスター・ドメイン用の新しいクォーラム装置の作成
- クラスター・ドメインの破棄

対話モードでの DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の実行:

-f パラメーターで XML 入力ファイルを指定せずに、**db2haicu** コマンドを実行して DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を呼び出すとき、ユーティリティーは対話モードで実行されます。対話モードでは、**db2haicu** は情報をテキスト・ベース形式で表示し、情報についてユーザーに照会します。

始める前に

- DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用する前に実行しなければない一連のタスクがあります。詳しくは、 144 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の前提条件』を参照してください。

このタスクについて

db2haicu を対話モードで実行するときには、画面上にテキスト形式で情報および質問が表示されます。**db2haicu** によって要求される情報は、画面下のプロンプトに入力できます。

手順

db2haicu を対話モードで実行するには、**-f** *input-file-name* を指定せずに **db2haicu** コマンドを呼び出します。

次のタスク

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) には、別個の診断ログがありません。データベース・マネージャーの診断ログ **db2diag** ログ・ファイル、および **db2pd** ツールを使用して、**db2haicu** のエラーを調査、および診断することができます。 詳しくは、 148 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) のトラブルシューティング』を参照してください。

XML 入力ファイルを使用した DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の実行:

-f *input-file-name* パラメーターを **db2haicu** コマンドで使用して、構成の詳細を指定する XML 入力ファイルを使って DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を実行することができます。複数のデータベース・パーティションを高可用性として構成するときなど、構成タスクを複数回実行する必要があるときに、XML 入力ファイルを使用した **db2haicu** の実行は便利です。

始める前に

- DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用する前に実行しなければない一連のタスクがあります。詳しくは、 144 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の前提条件』を参照してください。

このタスクについて

XML 入力ファイルのサンプルのセットが、**sqllib** ディレクトリーの **samples** サブディレクトリーにあるため、それらを適宜変更し、**db2haicu** で使用して、クラスター環境を構成できます。 詳しくは、 135 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の XML 入力ファイルのサンプル』を参照してください。

手順

1. XML 入力ファイルを作成します。
2. **-f** *input-file-name* を指定して、**db2haicu** を呼び出します。

db2haicu および作成する db2haicu-input.xml という入力ファイルを使用して、現行のデータベース・マネージャー・インスタンスのクラスター化された環境を構成するには、以下のコマンドを使用します。

```
db2haicu -f db2haicu-input.xml
```

次のタスク

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) には、別個の診断ログがありません。データベース・マネージャーの診断ログ **db2diag** ログ・ファイル、および **db2pd** ツールを使用して、**db2haicu** のエラーを調査、および診断することができます。 詳しくは、 148 ページの『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) のトラブルシューティング』を参照してください。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイル XML スキーマ定義:

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイル XML スキーマ定義 (XSD) は、**db2haicu** XML 入力ファイルで指定できるクラスター・ドメイン・オブジェクトを定義します。この **db2haicu** XSD は、sqllib/samples/ha/xml ディレクトリーの db2ha.xsd ファイル内にあります。

DB2ClusterType

db2haicu XML スキーマ定義 (XSD) のルート・エレメントは DB2Cluster で、そのタイプは DB2ClusterType です。**db2haicu** XML 入力ファイルは DB2Cluster エレメントから始まる必要があります。

『XML スキーマ定義』
115 ページの『サブエレメント』
116 ページの『属性』
116 ページの『使用上の注意』

XML スキーマ定義

```
<xs:complexType name='DB2ClusterType'>
  <xs:sequence>
    <xs:element name='DB2ClusterTemplate'
                type='DB2ClusterTemplateType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='ClusterDomain'
                type='ClusterDomainType'
                maxOccurs='unbounded'/>
    <xs:element name='FailoverPolicy'
                type='FailoverPolicyType'
                minOccurs='0'/>
    <xs:element name='DB2PartitionSet'
                type='DB2PartitionSetType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='HADRDBSet'
                type='HADRDBType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='HADBSet'
                type='HADBType'
                minOccurs='0'
```

```
                maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='clusterManagerName' type='xs:string' use='optional'/>
</xs:complexType>
```

サブエレメント

DB2ClusterTemplate

タイプ:

DB2ClusterTemplateType

使用上の注意:

db2haicu XML 入力ファイルに DB2ClusterTemplateType エレメントを含めないでください。DB2ClusterTemplateType エレメントは現在、将来の使用のために予約されています。

ClusterDomain

タイプ:

ClusterDomainType

ClusterDomainType エレメントには、クラスター・ドメイン内のマシンまたはコンピューター (*クラスター・ドメイン・ノード* とも呼ばれる)、*ネットワーク等価リソース* (相互にフェイルオーバーできるネットワークのグループ)、*クォーラム装置* (競合時の決定手段) についての指定を記述します。

出現規則:

DB2ClusterType エレメントに ClusterDomain エレメントを 1 つ以上含める必要があります。

FailoverPolicy

タイプ:

FailoverPolicyType

FailoverPolicyType エレメントは、クラスター・マネージャーがクラスター・ドメインで使用する *フェイルオーバー・ポリシー* を指定します。

出現規則:

DB2ClusterType エレメントに FailoverPolicy エレメントをゼロ個または 1 つ含めることができます。

DB2PartitionSet

タイプ:

DB2PartitionSetType

DB2PartitionSetType エレメントには、データベース・パーティションに関する情報を記述します。DB2PartitionSetType エレメントは、パーティション・データベース環境でのみ適用できます。

出現規則:

db2haicu XML スキーマ定義に従って、DB2ClusterType エレメントに DB2PartitionSet エレメントをゼロ個以上含めることができます。

HADRDBSet

タイプ:

HADRDBType

HADRDBType エレメントには、高可用性災害時リカバリー (HADR) のための 1 次データベースとスタンバイ・データベースのペアのリストを記述します。

出現規則:

db2haicu XML スキーマ定義に従って、DB2ClusterType エレメントに HADRDBSet エレメントをゼロ個以上含めることができます。

使用上の注意:

- パーティション・データベース環境では、HADRDBSet を含めてはなりません。
- HADRDBSet を含める場合は、FailoverPolicy エレメント内で HADRFailover のフェイルオーバー・ポリシーを指定する必要があります。

HADBSet

タイプ:

HADBType

HADBType エレメントには、クラスター・ドメインに組み込んで高可用性にするデータベースのリストを記述します。

出現規則:

db2haicu XML スキーマ定義に従って、DB2ClusterType エレメントに HADBSet エレメントをゼロ個以上含めることができます。

属性

clusterManagerName (オプション)

clusterManagerName 属性は、クラスター・マネージャーを指定します。

この属性の有効な値を次の表に示します。

表 3. clusterManager 属性の有効な値

clusterManagerName の値	クラスター・マネージャー製品
TSA	IBM Tivoli System Automation for Multiplatforms (SA MP)

使用上の注意

単一パーティション・データベース環境では通常、データベース・マネージャー・インスタンスごとに単一クラスター・ドメインを作成するだけで済みます。

複数パーティション・データベース環境の場合の可能な構成の 1 つを次に示します。

- FailoverPolicy エレメントを Mutual に設定します。
- DB2PartitionSet の DB2Partition サブエレメント内で、MutualPair エレメントを使用して単一クラスター・ドメイン内のクラスター・ドメイン・ノードを 2 つ指定します。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの ClusterDomainType XML スキーマ定義:

ClusterDomainType エレメントには、クラスター・ドメイン内のマシンまたはコンピューター (クラスター・ドメイン・ノード とも呼ばれる)、ネットワーク等価リソース (相互にフェイルオーバーできるネットワークのグループ)、クォーラム装置 (競合時の決定手段) についての指定を記述します。

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
118 ページの『属性』

スーパーエレメント

以下のタイプのエレメントに ClusterDomainType サブエレメントが含まれます。

- DB2ClusterType

XML スキーマ定義

```
<xs:complexType name='ClusterDomainType'>
  <xs:sequence>
    <xs:element name='Quorum'
                type='QuorumType'
                minOccurs='0'/>
    <xs:element name='PhysicalNetwork'
                type='PhysicalNetworkType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='ClusterNode'
                type='ClusterNodeType'
                maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='domainName'      type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

Quorum

タイプ:

QuorumType

QuorumType エレメントは、クラスター・ドメインのクォーラム装置を指定します。

出現規則:

ClusterDomainType エレメントに Quorum エレメントをゼロ個または 1 つ含めることができます。

PhysicalNetwork

タイプ:

PhysicalNetworkType

PhysicalNetworkType エレメントには、相互にフェイルオーバーできるネットワーク・インターフェース・カードを記述します。この種のネットワークは、ネットワーク等価リソース とも呼ばれます。

出現規則:

ClusterDomainType エレメントに PhysicalNetwork エレメントをゼロ個以上含めることができます。

ClusterNode

タイプ:

ClusterNodeType

ClusterNodeType エレメントには、クラスター内の特定のコンピューターまたはマシン (クラスター・ドメイン・ノード とも呼ばれる) に関する情報を記述します。

出現規則:

ClusterDomainType エレメント内に ClusterNode エレメントを少なくとも 1 つ以上指定する必要があります。

使用上の注意

IBM Tivoli System Automation for Multiplatforms (SA MP) は、最大 32 のクラスター・ドメイン・ノードをサポートします。クラスター・マネージャーとして SA MP を使用する場合は、最大 32 の ClusterNode エレメントを ClusterDomainType エレメントに含めることができます。

属性

domainName (必須)

ClusterDomainType エレメントの固有の名前を指定する必要があります。

Reliable Scalable Cluster Technology (RSCT) を使用してクラスターを管理する場合は、domainName について以下の制約事項があります。

- domainName に組み込めるのは、文字 A から Z、a から z、数字 0 から 9、ピリオド (.)、下線 (_) に限られます。
- domainName を「IW」にすることはできません。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの QuorumType XML スキーマ定義:

QuorumType エレメントは、クラスター・ドメインのクォーラム装置 を指定します。

『スーパーエレメント』
119 ページの『XML スキーマ定義』
119 ページの『サブエレメント』
119 ページの『属性』

スーパーエレメント

以下のタイプのエレメントに QuorumType サブエレメントが含まれます。

- ClusterDomainType

XML スキーマ定義

```
<xs:complexType name='QuorumType'>
  <xs:attribute name='quorumDeviceProtocol'
                type='QuorumDeviceProtocolType'
                use='required'/>
  <xs:attribute name='quorumDeviceName'
                type='xs:string'
                use='required'/>
</xs:complexType>
```

サブエレメント

なし。

属性

quorumDeviceProtocol (必須)

quorumDeviceProtocol は、使用するクォーラムのタイプを指定します。

クォーラム装置 は、クラスター・マネージャーがクラスター管理上の決定を行うときに、通常の決定プロセスでは明確に選択できない場合に役立ちます。クラスター・マネージャーが、可能性のある複数のアクションから選択する必要があるときは、可能な各アクションをサポートするクラスター・ドメイン・ノードの数をカウントし、サポートしているクラスター・ドメイン・ノードが最も多いアクションを選択します。クラスター・ドメイン・ノードの数がまったく同じであるために複数の選択肢がサポートされる場合、クラスター・マネージャーはクォーラム装置を参照して選択を行います。

quorumDeviceProtocol 属性のタイプは QuorumDeviceProtocolType です。

QuorumDeviceProtocolType の XML スキーマ定義を以下に示します。

```
<xs:simpleType name='QuorumDeviceProtocolType'>
  <xs:restriction base='xs:string'>
    <xs:enumeration value='disk'/>
    <xs:enumeration value='scsi'/>
    <xs:enumeration value='network'/>
    <xs:enumeration value='eckd'/>
    <xs:enumeration value='mns'/>
  </xs:restriction>
</xs:simpleType>
```

この属性の現在サポートされている値を、次の表に示します。

表 4. quorumDeviceProtocol 属性の有効な値

quorumDeviceProtocol の値	意味
network	network クォーラム装置とは、どのクラスター・ドメイン・ノードもいつでも接続できる IP アドレスのことです。

quorumDeviceName (必須)

quorumDeviceName の値は、quorumDeviceProtocol で指定したクォーラム装置のタイプによります。

この属性の有効な値を次の表に示します。

表 5. quorumDeviceName 属性の有効な値

quorumDeviceProtocol の値	quorumDeviceName の有効な値
network	正しいフォーマットの IP アドレスのストリング。例えば、以下のようにします。 12.126.4.5 network クォーラム装置として有効であると指定する IP アドレスであるからには、どのクラスター・ドメイン・ノードもこの IP アドレスにアクセスできなければなりません (例えば ping ユーティリティーを使用)。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの PhysicalNetworkType XML スキーマ定義:

PhysicalNetworkType エレメントには、相互にフェイルオーバーできるネットワーク・インターフェース・カードを記述します。この種のネットワークは、ネットワーク等価リソース とも呼ばれます。

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
121 ページの『属性』

スーパーエレメント

以下のタイプのエレメントに PhysicalNetworkType サブエレメントが含まれます。

- ClusterDomainType

XML スキーマ定義

```
<xs:complexType name='PhysicalNetworkType'>
  <xs:sequence>
    <xs:element name='Interface'
                type='InterfaceType'
                minOccurs='1'
                maxOccurs='unbounded'/>
    <xs:element name='LogicalSubnet'
                type='IPAddressType'
                minOccurs='0'
                maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='physicalNetworkName'
                type='xs:string'
                use='required'/>
  <xs:attribute name='physicalNetworkProtocol'
                type='PhysicalNetworkProtocolType'
                use='required'/>
</xs:complexType>
```

サブエレメント

インターフェース

タイプ:

InterfaceType

InterfaceType エレメントは、IP アドレス、ネットワーク内のコンピューターまたはマシン (クラスター・ドメイン・ノード とも呼ばれる) の名前、そのクラスター・ドメイン・ノード上のネットワーク・インターフェース・カード (NIC) の名前で構成されます。

出現規則:

PhysicalNetworkType エレメント内に Interface エレメントを 1 つ以上指定する必要があります。

LogicalSubnet

タイプ:

IPAddressType

IPAddressType エレメントには、IP アドレスのすべての詳細を記述します。これには、基底アドレス、サブネット・マスク、IP アドレスが属するネットワークの名前が含まれます。

出現規則:

PhysicalNetworkType エレメントに LogicalSubnet エレメントをゼロ個以上含めることができます。

属性

physicalNetworkName (必須)

PhysicalNetworkType エレメントごとに固有の physicalNetworkName を指定する必要があります。

physicalNetworkProtocol (必須)

physicalNetworkProtocol 属性のタイプは PhysicalNetworkProtocolType です。

PhysicalNetworkProtocolType エレメントの XML スキーマ定義を以下に示します。

```
<xs:simpleType name='PhysicalNetworkProtocolType'>
  <xs:restriction base='xs:string'>
    <xs:enumeration value='ip'/>
    <xs:enumeration value='rs232'/>
    <xs:enumeration value='scsi'/>
    <xs:enumeration value='ssa'/>
    <xs:enumeration value='disk'/>
  </xs:restriction>
</xs:simpleType>
```

この属性の現在サポートされている値を、次の表に示します。

表 6. physicalNetworkProtocol 属性の有効な値

physicalNetworkProtocol の値	意味
ip	TCP/IP プロトコル

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの InterfaceType XML スキーマ定義:

InterfaceType エレメントは、IP アドレス、ネットワーク内のコンピューターまたはマシン (クラスター・ドメイン・ノード とも呼ばれる) の名前、そのクラスター・ドメイン・ノード上のネットワーク・インターフェース・カード (NIC) の名前で構成されます。

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
『属性』

スーパーエレメント

以下のタイプのエレメントに InterfaceType サブエレメントが含まれます。

- PhysicalNetworkType

XML スキーマ定義

```
<xs:complexType name='InterfaceType'>
  <xs:sequence>
    <xs:element name='IPAddress'       type='IPAddressType'/>
  </xs:sequence>
  <xs:attribute name='interfaceName'   type='xs:string' use='required'/>
  <xs:attribute name='clusterNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

IPAddress

タイプ:

IPAddressType

IPAddressType エレメントには、IP アドレスのすべての詳細を記述します。これには、*基底アドレス*、サブネット・マスク、IP アドレスが属するネットワークの名前が含まれます。

出現規則:

InterfaceType エレメント内に IPAddress を 1 つだけ指定する必要があります。

属性

interfaceName (必須)

interfaceName 属性で NIC の名前を指定する必要があります。
interfaceName で指定する NIC は、clusterNodeName 属性で指定するクラスター・ドメイン・ノードに存在しなければなりません。

clusterNodeName (必須)

IPAddress エレメントで指定する IP アドレスにあるクラスター・ドメイン・ノードの名前を指定する必要があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの IPAddressType XML スキーマ・エレメント:

IPAddressType エレメントには、IP アドレスのすべての詳細を記述します。これには、*基底アドレス*、サブネット・マスク、IP アドレスが属するネットワークの名前が含まれます。

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
『属性』

スーパーエレメント

以下のタイプのエレメントに IPAddressType サブエレメントが含まれます。

- PhysicalNetworkType
- InterfaceType
- DB2PartitionType

XML スキーマ定義

```
<xs:complexType name='IPAddressType'>
  <xs:attribute name='baseAddress' type='xs:string' use='required'/>
  <xs:attribute name='subnetMask'  type='xs:string' use='required'/>
  <xs:attribute name='networkName' type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

なし。

属性

baseAddress (必須)
IP アドレスを有効なフォーマット (0 から 255 までの数値の 4 セットをピリオドで区切る) で記述したストリングで、基底 IP アドレスを指定する必要があります。例えば、以下のようにします。

```
162.148.31.101
```

subnetMask (必須)
IP アドレスを有効なフォーマットで記述したストリングで、基底 IP アドレスを指定する必要があります。

networkName (必須)
この IPAddress エレメントが含まれる PhysicalNetworkType エレメントの physicalNetworkName 属性に指定した値と同じ値を、ここで networkName に指定する必要があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの ClusterNodeType XML スキーマ定義:

ClusterNodeType エレメントには、クラスター内の特定のコンピューターまたはマシン (クラスター・ドメイン・ノード とも呼ばれる) に関する情報を記述します。

124 ページの『スーパーエレメント』
124 ページの『XML スキーマ定義』
124 ページの『サブエレメント』
124 ページの『属性』

スーパーエレメント

以下のタイプのエレメントに ClusterNodeType エレメントが含まれます。

- ClusterDomainType

XML スキーマ定義

```
<xs:complexType name='ClusterNodeType'>
  <xs:attribute name='clusterNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

なし。

属性

clusterNodeName (必須)
クラスター・ドメイン・ノードの名前を指定する必要があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの FailoverPolicyType XML スキーマ定義:

FailoverPolicyType エレメントは、クラスター・マネージャーがクラスター・ドメインで使用するフェイルオーバー・ポリシー を指定します。

『スーパーエレメント』
『XML スキーマ定義』
125 ページの『サブエレメント』
125 ページの『可能な値』

スーパーエレメント

以下のタイプのエレメントに InterfaceType サブエレメントが含まれます。

- DB2ClusterType

XML スキーマ定義

```
<xs:complexType name='FailoverPolicyType'>
  <xs:choice>
    <xs:element name='RoundRobin'
                type='xs:string'
                minOccurs='0' />
    <xs:element name='Mutual'
                type='xs:string'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='NPlusM'
                type='xs:string'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='LocalRestart'
                type='xs:string'
                fixed=''/>
    <xs:element name='HADRFailover'
                type='xs:string'
                fixed=''/>
    <xs:element name='Custom'
```

```
                type='xs:string'
                minOccurs='0' />
  </xs:choice>
</xs:complexType>
```

サブエレメント

なし。

可能な値

以下の選択肢から 1 つ選択します。この選択を行うことで、クラスター・ドメイン内のどこかで障害が発生した場合にクラスター・マネージャーが使用するフェイルオーバー・ポリシーのタイプを指定します。

フェイルオーバー・ポリシー は、ネットワーク・インターフェース・カードやデータベース・サーバーなどのクラスター・エレメントに障害が発生したときにクラスター・マネージャーがどのように対処するかを指定したものです。一般に、クラスター・マネージャーは障害の起きたエレメントから代替エレメントにワークロードを転送します。この代替エレメントは、障害の起きたエレメントに適合する代替としてあらかじめクラスター・マネージャーに示されています。障害の起きたエレメントから 2 次エレメントへのこうしたワークロード転送を、*フェイルオーバー* といいます。

RoundRobin
ラウンドロビン・フェイルオーバー・ポリシー を使用すると、クラスター・ドメイン内のあるコンピューター (*クラスター・ドメイン・ノード* または単に*ノード* とも呼ばれる) に関連した障害が発生すると、データベース・マネージャーは障害の起きたクラスター・ドメイン・ノードでの作業をクラスター・ドメイン内の他のノードで再開します。

Mutual
相互フェイルオーバー・ポリシー を構成するには、クラスター・ドメイン内のコンピューター (*クラスター・ドメイン・ノード* または単に*ノード* とも呼ばれる) のペアをシステム・ペアとして関連付けます。このペアの一方のノードに障害が発生すると、障害の起きたノード上のデータベース・パーティションは、ペアのもう一方のノードにフェイルオーバーします。相互フェイルオーバーは、複数のデータベース・パーティションがある場合のみ使用できます。

NPlusM
N プラス M フェイルオーバー・ポリシー を使用すると、クラスター・ドメイン内のあるコンピューター (*クラスター・ドメイン・ノード* または単に*ノード* とも呼ばれる) に関連した障害が発生すると、障害の起きたノード上のデータベース・パーティションは、クラスター・ドメイン内の他のいずれかのノードにフェイルオーバーします。ロービング HA フェイルオーバーを有効にした場合、最後に障害の起きたノードが再びオンラインになると、そのノードがスタンバイ・ノードになります。 M=1 の場合は、N プラス M フェイルオーバー・ポリシーのロービング HA フェイルオーバーのみがサポートされます。 N プラス M フェイルオーバーは、複数のデータベース・パーティションがある場合のみ使用できます。

LocalRestart

ローカル再開フェイルオーバー・ポリシー を使用すると、クラスター・ドメイン内のコンピューター (クラスター・ドメイン・ノード または単にノード とも呼ばれる) の一つで障害が発生すると、データベース・マネージャーは障害の起きたそのノード上で (つまりローカルに) データベースを再開します。

HADRFailover

HADR フェイルオーバー・ポリシー を構成する場合は、DB2 高可用性災害時リカバリー (HADR) フィーチャーでフェイルオーバーを管理できるようにすることになります。HADR 1 次データベースに障害が起こると、データベース・マネージャーは障害の起きたデータベースから HADR スタンバイ・データベースにワークロードを移動します。

Custom

カスタム・フェイルオーバー・ポリシー を構成する場合は、クラスター・ドメイン内のコンピューター (クラスター・ドメイン・ノード または単にノード とも呼ばれる) のうち、データベース・マネージャーがフェイルオーバー先として使用できるコンピューターのリストを作成します。クラスター・ドメイン内のノードに障害が起こると、データベース・マネージャーは障害の起きたノードから、指定されたリスト内のノードの一つにワークロードを移動します。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの DB2PartitionSetType XML スキーマ定義:

DB2PartitionSetType エレメントには、データベース・パーティションに関する情報を記述します。DB2PartitionSetType エレメントは、パーティション・データベース環境でのみ適用できます。

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
127 ページの『属性』

スーパーエレメント

InterfaceType は、以下のエレメントのサブエレメントです。

- PhysicalNetworkType

XML スキーマ定義

```
<xs:complexType name='DB2PartitionSetType'>
  <xs:sequence>
    <xs:element name='DB2Partition'
                type='DB2PartitionType'
                maxOccurs='unbounded'/>
  </xs:sequence>
</xs:complexType>
```

サブエレメント

DB2Partition

タイプ:

DB2PartitionType

DB2PartitionType エレメントは、データベース・パーティションを指定するとともに、データベース・パーティションが属する DB2 データベース・マネージャー・インスタンスと、データベース・パーティション番号を指定します。

出現規則:

DB2PartitionSetType エレメント内に DB2Partition エレメントを 1 つ以上指定する必要があります。

属性

なし。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの DB2PartitionType XML スキーマ・エレメント:

DB2PartitionType エレメントは、データベース・パーティションを指定するとともに、データベース・パーティションが属する DB2 データベース・マネージャー・インスタンスと、データベース・パーティション番号を指定します。

『スーパーエレメント』
『XML スキーマ定義』
128 ページの『サブエレメント』
129 ページの『属性』

スーパーエレメント

InterfaceType は、以下のエレメントのサブエレメントです。

- DB2PartitionSetType

XML スキーマ定義

```
<xs:complexType name='DB2PartitionType'>
 <xs:sequence>
    <xs:element name='VirtualIPAddress'
                type='IPAddressType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='Mount'
                type='MountType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='HADRDB'
                type='HADRDBType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='MutualPair'
                type='MutualPolicyType'
                minOccurs='0'
                maxOccurs='1'/>
    <xs:element name='NPlusMNode'
                type='NPlusMPolicyType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='CustomNode'
                type='CustomPolicyType'
```

```
                minOccurs='0'
                maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='instanceName'     type='xs:string'  use='required'/>
  <xs:attribute name='dbpartitionnum'   type='xs:integer' use='required'/>
</xs:complexType>
```

サブエレメント

VirtualIPAddress

タイプ: IPAddressType

IPAddressType エレメントには、IP アドレスのすべての詳細を記述します。これには、基底アドレス、サブネット・マスク、IP アドレスが属するネットワークの名前が含まれます。

VirtualIPAddress の組み込みは省略できます。また、DB2PartitionType エレメントに組み込むことができる VirtualIPAddress エレメントの数は無制限です。

Mount タイプ: MountType

MountType エレメントには、マウント・ポイント に関する情報として、マウントされたファイルのロケーションを示すファイル・パスを記述します。

Mount の組み込みは省略できます。また、DB2PartitionType エレメントに組み込むことができる Mount エレメントの数は無制限です。

HADRDB

タイプ: HADRDBType

HADRDBType エレメントには、高可用性災害時リカバリー (HADR) のための 1 次データベースとスタンバイ・データベースのペアのリストを記述します。

HADRDB の組み込みは省略できます。また、DB2PartitionType エレメントに組み込むことができる HADRDB エレメントの数は無制限です。

MutualPair

タイプ: MutualPolicyType

MutualPolicyType エレメントには、相互にフェイルオーバーできるクラスター・ドメイン・ノードのペアに関する情報を記述します。

MutualPair の組み込みは省略できます。また、DB2PartitionType エレメントに組み込むことができる MutualPair エレメントは 1 つだけです。

NPlusMNode

タイプ: NPlusMPolicyType

NPlusMNode の組み込みは省略できます。また、DB2PartitionType エレメントに組み込むことができる NPlusMNode エレメントの数は無制限です。

CustomNode

タイプ: CustomPolicyType

CustomNode の組み込みは省略できます。また、DB2PartitionType エレメントに組み込むことのできる CustomNode エレメントの数は無制限です。

属性

instanceName (必須)
instanceName 属性で、この DB2PartitionType エレメントを関連付ける DB2 データベース・マネージャー・インスタンスを指定する必要があります。

dbpartitionnum (必須)
dbpartitionnum 属性で、データベース・パーティションを一意に識別できるデータベース・パーティション番号 (例えば db2nodes.cfg ファイルで指定された dbpartitionnum の番号) を指定する必要があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの MountType XML スキーマ定義:

MountType エレメントには、マウント・ポイント に関する情報として、マウントされたファイルのロケーションを示すファイル・パスを記述します。

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
『属性』

スーパーエレメント

以下のタイプのエレメントに MountType サブエレメントが含まれます。

- DB2PartitionType

XML スキーマ定義

```
<xs:complexType name='MountType'>
  <xs:attribute name='filesystemPath' type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

なし。

属性

filesystemPath (必須)
ファイル・システムをマウントしたときにマウント・ポイントに関連付けられたパスを指定します。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの MutualPolicyType XML スキーマ定義:

MutualPolicyType エレメントには、相互にフェイルオーバーできるクラスター・ドメイン・ノードのペアに関する情報を記述します。

130 ページの『スーパーエレメント』
130 ページの『XML スキーマ定義』
130 ページの『サブエレメント』
130 ページの『属性』

スーパーエレメント

以下のタイプのエレメントに MutualPolicyType サブエレメントが含まれます。

- DB2PartitionType

XML スキーマ定義

```
<xs:complexType name='MutualPolicyType'>
  <xs:attribute name='systemPairNode1' type='xs:string' use='required'/>
  <xs:attribute name='systemPairNode2' type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

なし。

属性

systemPairNode1 (必須)
systemPairNode1 では、systemPairNode2 で指定するクラスター・ドメイン・ノードのフェイルオーバーが可能なクラスター・ドメイン・ノードの名前を指定する必要があります。

systemPairNode2 (必須)
systemPairNode2 では、systemPairNode1 で指定するクラスター・ドメイン・ノードのフェイルオーバーが可能なクラスター・ドメイン・ノードの名前を指定する必要があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの NPlusMPolicyType XML スキーマ定義:

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
『属性』

スーパーエレメント

以下のタイプのエレメントに NPlusMPolicyType サブエレメントが含まれます。

- DB2PartitionType

XML スキーマ定義

```
<xs:complexType name='NPlusMPolicyType'>
  <xs:attribute name='standbyNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

なし。

属性

standbyNodeName (必須)
standbyNodeName エレメントで、この NPlusMPolicyType エレメントが含ま

れるパーティションのフェイルオーバー先として使用できるクラスター・ドメイン・ノードの名前を指定する必要があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイル用の CustomPolicyType XML スキーマ定義:

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
『属性』

スーパーエレメント

以下のタイプのエレメントに、CustomPolicyType サブエレメントが含まれます。

- DB2PartitionType

XML スキーマ定義

```
<xs:complexType name='NPlusMPolicyType'>
  <xs:attribute name='standbyNodeName' type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

なし。

属性

customNodeName (必須)
customNodeName エレメントの中で、この CustomPolicyType エレメントが含まれるパーティションのフェイルオーバー先として使用できるクラスター・ドメイン・ノードの名前を指定する必要があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの HADRDBType XML スキーマ定義:

HADRDBType エレメントには、高可用性災害時リカバリー (HADR) のための 1 次データベースとスタンバイ・データベースのペアのリストを記述します。

『スーパーエレメント』
132 ページの『XML スキーマ定義』
132 ページの『サブエレメント』
132 ページの『属性』
132 ページの『使用上の注意』
132 ページの『制約事項』

スーパーエレメント

以下のタイプのエレメントに HADRDBType サブエレメントが含まれます。

- DB2ClusterType
- DB2PartitionType

XML スキーマ定義

```
<xs:complexType name='HADRDBType'>
  <xs:sequence>
    <xs:element name='VirtualIPAddress'
                type='IPAddressType'
                minOccurs='0'
                maxOccurs='unbounded'/>
    <xs:element name='HADRDB'
                type='HADRDBDefn'
                maxOccurs='unbounded'/>
  </xs:sequence>
</xs:complexType>
```

サブエレメント

VirtualIPAddress

タイプ:

IPAddressType

IPAddressType エレメントには、IP アドレスのすべての詳細を記述します。これには、*基底アドレス*、サブネット・マスク、IP アドレスが属するネットワークの名前が含まれます。

出現規則:

HADRDBType エレメントに VirtualIPAddress エレメントをゼロ個以上含めることができます。

HADRDB

タイプ:

HADRDBDefn

HADRDBDefn エレメントには、高可用性災害時リカバリー (HADR) のための 1 次データベースとスタンバイ・データベースのペアに関する情報を記述します。

出現規則:

HADRDBType エレメントに VirtualIPAddress エレメントを 1 つ以上含めることができます。

属性

なし。

使用上の注意

あるクラスター・ドメインの仕様に HADRDBType エレメントを含める場合は、HADRFailover を指定した FailoverPolicy エレメントを、同じクラスター・ドメインの仕様に含める必要があります。

制約事項

パーティション・データベース環境では、HADRDBType エレメントは使用できません。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの HADRDBDefn XML スキーマ定義:

HADRDBDefn エレメントには、高可用性災害時リカバリー (HADR) のための 1 次データベースとスタンバイ・データベースのペアに関する情報を記述します。

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
『属性』

スーパーエレメント

以下のタイプのエレメントに HADRDBDefn サブエレメントが含まれます。

- HADRDBType

XML スキーマ定義

```
<xs:complexType name='HADRDBDefn'>
  <xs:attribute name='databaseName'      type='xs:string' use='required'/>
  <xs:attribute name='localInstance'     type='xs:string' use='required'/>
  <xs:attribute name='remoteInstance'    type='xs:string' use='required'/>
  <xs:attribute name='localHost'         type='xs:string' use='required'/>
  <xs:attribute name='remoteHost'        type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

なし。

属性

databaseName (必須)
HADR データベースの名前を入力します。

localInstance (必須)
localInstance は、HADR 1 次データベースのデータベース・マネージャー・インスタンスです。

remoteInstance (必須)
remoteInstance は、HADR スタンバイ・データベースのデータベース・マネージャー・インスタンスです。

localHost (必須)
localHost は、HADR 1 次データベースがあるクラスター・ドメイン・ノードのホスト名です。

remoteHost (必須)
remoteHost は、HADR スタンバイ・データベースがあるクラスター・ドメイン・ノードのホスト名です。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの HADBType XML スキーマ定義:

HADBType エレメントには、クラスター・ドメインに組み込んで高可用性にするデータベースのリストを記述します。

『スーパーエレメント』
『XML スキーマ定義』
『サブエレメント』
『属性』

スーパーエレメント

以下のタイプのエレメントに HADBType サブエレメントが含まれます。

- DB2ClusterType

XML スキーマ定義

```
<xs:complexType name='HADBType'>
  <xs:sequence>
    <xs:element name='HADB'         type='HADBDefn' maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='instanceName' type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

HADB

タイプ:

HADBDefn

HADBDefn エレメントには、クラスター・ドメインに組み込んで高可用性にするデータベースを記述します。

出現規則:

HADBType エレメントに HADB エレメントを 1 つ以上含める必要があります。

属性

instanceName (必須)

instanceName 属性で、HADB エレメントで指定したデータベースが属する DB2 データベース・マネージャー・インスタンスを指定する必要があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) 入力ファイルの HADBDefn XML スキーマ・エレメント:

HADBDefn エレメントには、クラスター・ドメインに組み込んで高可用性にするデータベースを記述します。

『スーパーエレメント』
135 ページの『XML スキーマ定義』
135 ページの『サブエレメント』
135 ページの『属性』

スーパーエレメント

HADBDefn は、以下のエレメントのサブエレメントです。

- HADRDBType

XML スキーマ定義

```
<xs:complexType name='HADBDefn'>
  <xs:attribute name='databaseName' type='xs:string' use='required'/>
</xs:complexType>
```

サブエレメント

なし。

属性

databaseName (必須)
databaseName 属性にデータベース名を 1 つだけ指定する必要があります。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の XML 入力ファイルのサンプル:

XML 入力ファイルのサンプルのセットが、sqllib ディレクトリーの samples サブディレクトリーにあるため、それらを適宜変更し、**db2haicu** で使用して、クラスター環境を構成できます。

db2ha_sample_sharedstorage_mutual.xml:

サンプル・ファイル db2ha_sample_sharedstorage_mutual.xml は、新しいクラスター・ドメイン を指定するために DB2 高可用性インスタンス構成ユーティリティー (db2haicu) に渡す XML 入力ファイルの一例です。
db2ha_sample_sharedstorage_mutual.xml は sqllib/samples/ha/xml ディレクトリーにあります。

フィーチャー

db2ha_sample_sharedstorage_mutual.xml サンプルは、**db2haicu** と XML 入力ファイルを使用して以下の詳細仕様のクラスター・ドメインを定義する方法を示しています。

- クォーラム装置: ネットワーク
- クラスター内コンピューター (クラスター・ドメイン・ノード) 数: 2
- フェイルオーバー・ポリシー: 相互
- データベース・パーティション数: 1
- 仮想 (サービス) IP アドレス数: 1
- フェイルオーバー対象共有マウント・ポイント数: 1

XML ソース

```
<!-- ================================================================= -->
<!-- = Use the DB2 High Availability Instance Configuration Utility  = -->
<!-- = (db2haicu) XML schema definition, db2ha.xsd, and specify      = -->
<!-- = IBM Tivoli System Automation for Multiplatforms (SA MP)       = -->
<!-- = Base Component as the cluster manager.                        = -->
<!-- ================================================================= -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
            clusterManagerName="TSA"
            version="1.0">

  <!-- ================================================================= -->
```

```
<!-- = Create a cluster domain named db2HAdomain.                    = -->
<!-- ================================================================ -->
<ClusterDomain domainName="db2HAdomain">

  <!-- ================================================================ -->
  <!-- = Specify a network quorum device (IP address: 19.126.4.5).   = -->
  <!-- = The IP must be pingable at all times by each of the cluster = -->
  <!-- = domain nodes.                                                = -->
  <!-- ================================================================ -->
  <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

  <!-- ================================================================ -->
  <!-- = Create a network named db2_public_network_0 with an IP      = -->
  <!-- = network protocol.                                            = -->
  <!-- = This network contains two computers: hasys01 and hasys02.   = -->
  <!-- = Each computer has one network interface card (NIC) called   = -->
  <!-- = eth0.                                                        = -->
  <!-- = The IP address of the NIC on hasys01 is 19.126.52.139       = -->
  <!-- = The IP address of the NIC on hasys02 is 19.126.52.140       = -->
  <!-- ================================================================ -->
  <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                   physicalNetworkProtocol="ip">

    <Interface interfaceName="eth0" clusterNodeName="hasys01">
      <IPAddress baseAddress="19.126.52.139"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

    <Interface interfaceName="eth0" clusterNodeName="hasys02">
      <IPAddress baseAddress="19.126.52.140"
                 subnetMask="255.255.255.0"
                 networkName="db2_public_network_0"/>
    </Interface>

  </PhysicalNetwork>

  <!-- ================================================================ -->
  <!-- = List the computers (cluster nodes) in the cluster domain.   = -->
  <!-- ================================================================ -->
  <ClusterNode clusterNodeName="hasys01"/>
  <ClusterNode clusterNodeName="hasys02"/>

</ClusterDomain>


<!-- ================================================================ -->
<!-- = The failover policy specifies the order in which the cluster  = -->
<!-- = domain nodes should fail over.                               = -->
<!-- ================================================================ -->
<FailoverPolicy>
   <Mutual />
</FailoverPolicy>


<!-- ================================================================ -->
<!-- = Specify all the details of the database partition            = -->
<!-- ================================================================ -->
<DB2PartitionSet>

  <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
     <VirtualIPAddress baseAddress="19.126.52.222"
                       subnetMask="255.255.255.0"
                       networkName="db2_public_network_0"/>
     <Mount filesystemPath="/home/db2inst1"/>
     <MutualPair systemPairNode1="hasys01" systemPairNode2="hasys02" />
  </DB2Partition>
```

```
  </DB2PartitionSet>

</DB2Cluster>
```

db2ha_sample_DPF_mutual.xml:

サンプル・ファイル db2ha_sample_DPF_mutual.xml は、新しいクラスター・ドメイン を指定するために DB2 高可用性インスタンス構成ユーティリティー (db2haicu) に渡す XML 入力ファイルの一例です。db2ha_sample_DPF_mutual.xml は sqllib/samples/ha/xml ディレクトリーにあります。

フィーチャー

db2ha_sample_DPF_mutual.xml サンプルは、**db2haicu** と XML 入力ファイルを使用して以下の詳細仕様のクラスター・ドメインを定義する方法を示しています。

- クォーラム装置: ネットワーク
- クラスター内コンピューター (クラスター・ドメイン・ノード) 数: 4
- フェイルオーバー・ポリシー: 相互
- データベース・パーティション数: 2
- 仮想 (サービス) IP アドレス数: 1
- フェイルオーバー対象共有マウント・ポイント数: 2
- 高可用性構成データベース数: 2

XML ソース

```
<!-- ================================================================== -->
<!-- = Use the DB2 High Availability Instance Configuration Utility  = -->
<!-- = (db2haicu) XML schema definition, db2ha.xsd, and specify      = -->
<!-- = IBM Tivoli System Automation for Multiplatforms (SA MP)       = -->
<!-- = Base Component as the cluster manager.                        = -->
<!-- ================================================================== -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
            clusterManagerName="TSA"
            version="1.0">

  <!-- ================================================================== -->
  <!-- = Create a cluster domain named db2HAdomain.                    = -->
  <!-- ================================================================== -->
  <ClusterDomain domainName="db2HAdomain">

    <!-- ================================================================== -->
    <!-- = Specify a network quorum device (IP address: 19.126.4.5).   = -->
    <!-- = The IP must be pingable at all times by each of the cluster = -->
    <!-- = domain nodes.                                                = -->
    <!-- ================================================================== -->
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

    <!-- ================================================================== -->
    <!-- = Create a network named db2_public_network_0 with an IP      = -->
    <!-- = network protocol.                                            = -->
    <!-- = This network contains four computers: hasys01, hasys02,     = -->
    <!-- = hasys03, and hasys04.                                        = -->
    <!-- = Each computer has a network interface card called eth0.     = -->
    <!-- = The IP address of eth0 on hasys01 is 19.126.124.30          = -->
    <!-- = The IP address of eth0 on hasys02 is 19.126.124.31          = -->
    <!-- = The IP address of eth0 on hasys03 is 19.126.124.32          = -->
    <!-- = The IP address of eth0 on hasys04 is 19.126.124.33          = -->
```

```
<!-- ============================================================= -->
<PhysicalNetwork physicalNetworkName="db2_public_network_0"
                 physicalNetworkProtocol="ip">

  <Interface interfaceName="eth0" clusterNodeName="hasys01">
    <IPAddress baseAddress="19.126.124.30"
               subnetMask="255.255.255.0"
               networkName="db2_public_network_0"/>
  </Interface>

  <Interface interfaceName="eth0" clusterNodeName="hasys02">
    <IPAddress baseAddress="19.126.124.31"
               subnetMask="255.255.255.0"
               networkName="db2_public_network_0"/>
  </Interface>

  <Interface interfaceName="eth0" clusterNodeName="hasys03">
    <IPAddress baseAddress="19.126.124.32"
               subnetMask="255.255.255.0"
               networkName="db2_public_network_0"/>
  </Interface>

  <Interface interfaceName="eth0" clusterNodeName="hasys04">
    <IPAddress baseAddress="19.126.124.33"
               subnetMask="255.255.255.0"
               networkName="db2_public_network_0"/>
  </Interface>

</PhysicalNetwork>

<!-- ============================================================= -->
<!-- = Create a network named db2_private_network_0 with an IP     = -->
<!-- = network protocol.                                           = -->
<!-- = This network contains four computers: hasys01, hasys02,     = -->
<!-- = hasys03, and hasys04 (same as db2_public_network_0.)        = -->
<!-- = In addition to eth0, each computer has a network interface  = -->
<!-- = card called eth1.                                           = -->
<!-- = The IP address of eth1 on hasys01 is 192.168.23.101         = -->
<!-- = The IP address of eth1 on hasys02 is 192.168.23.102         = -->
<!-- = The IP address of eth1 on hasys03 is 192.168.23.103         = -->
<!-- = The IP address of eth1 on hasys04 is 192.168.23.104         = -->
<!-- ============================================================= -->
<PhysicalNetwork physicalNetworkName="db2_private_network_0"
                 physicalNetworkProtocol="ip">

  <Interface interfaceName="eth1" clusterNodeName="hasys01">
    <IPAddress baseAddress="192.168.23.101"
               subnetMask="255.255.255.0"
               networkName="db2_private_network_0"/>
  </Interface>

  <Interface interfaceName="eth1" clusterNodeName="hasys02">
    <IPAddress baseAddress="192.168.23.102"
               subnetMask="255.255.255.0"
               networkName="db2_private_network_0"/>
  </Interface>

  <Interface interfaceName="eth1" clusterNodeName="hasys03">
    <IPAddress baseAddress="192.168.23.103"
               subnetMask="255.255.255.0"
               networkName="db2_private_network_0"/>
  </Interface>

  <Interface interfaceName="eth1" clusterNodeName="hasys04">
    <IPAddress baseAddress="192.168.23.104"
               subnetMask="255.255.255.0"
               networkName="db2_private_network_0"/>
```

```
      </Interface>

    </PhysicalNetwork>

    <!-- ================================================================= -->
    <!-- = List the computers (cluster nodes) in the cluster domain.   = -->
    <!-- ================================================================= -->
    <ClusterNode clusterNodeName="hasys01"/>
    <ClusterNode clusterNodeName="hasys02"/>
    <ClusterNode clusterNodeName="hasys03"/>
    <ClusterNode clusterNodeName="hasys04"/>

  </ClusterDomain>


  <!-- ================================================================= -->
  <!-- = The failover policy specifies the order in which the cluster  = -->
  <!-- = domain nodes should fail over.                                = -->
  <!-- ================================================================= -->
  <FailoverPolicy>
     <Mutual />
  </FailoverPolicy>


  <!-- ================================================================= -->
  <!-- = Specify all the details of the database partitions.           = -->
  <!-- ================================================================= -->
  <DB2PartitionSet>

    <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
       <VirtualIPAddress baseAddress="19.126.124.251"
                         subnetMask="255.255.255.0"
                         networkName="db2_public_network_0"/>
       <Mount filesystemPath="/hafs/db2inst1/NODE0000"/>
       <MutualPair systemPairNode1="hasys01" systemPairNode2="hasys02" />
    </DB2Partition>

    <DB2Partition dbpartitionnum="1" instanceName="db2inst1">
       <Mount filesystemPath="/hafs/db2inst1/NODE0001"/>
       <MutualPair systemPairNode1="hasys02" systemPairNode2="hasys01" />
    </DB2Partition>

    <DB2Partition dbpartitionnum="2" instanceName="db2inst1">
       <Mount filesystemPath="/hafs/db2inst1/NODE0002"/>
       <MutualPair systemPairNode1="hasys03" systemPairNode2="hasys04" />
    </DB2Partition>

    <DB2Partition dbpartitionnum="3" instanceName="db2inst1">
       <Mount filesystemPath="/hafs/db2inst1/NODE0003"/>
       <MutualPair systemPairNode1="hasys04" systemPairNode2="hasys03" />
    </DB2Partition>

  </DB2PartitionSet>


  <!-- ================================================================= -->
  <!-- = List of databases to be configured for High Availability      = -->
  <!-- ================================================================= -->
  <HADBSet instanceName="db2inst1">
    <HADB databaseName = "SAMPLE" />
    <HADB databaseName = "MYDB" />
  </HADBSet>


</DB2Cluster>
```

db2ha_sample_DPF_NPlusM.xml:

サンプル・ファイル db2ha_sample_DPF_NPlusM.xml は、新しいクラスター・ドメイン を指定するために DB2 高可用性インスタンス構成ユーティリティー (db2haicu) に渡す XML 入力ファイルの一例です。db2ha_sample_DPF_NPlusM.xml は sqllib/samples/ha/xml ディレクトリーにあります。

フィーチャー

db2ha_sample_DPF_NPlusM.xml サンプルは、**db2haicu** と XML 入力ファイルを使用して以下の詳細仕様のクラスター・ドメインを定義する方法を示しています。

- クォーラム装置: ネットワーク
- クラスター内コンピューター (クラスター・ドメイン・ノード) 数: 4
- フェイルオーバー・ポリシー: N プラス M
- データベース・パーティション数: 2
- 仮想 (サービス) IP アドレス数: 1
- フェイルオーバー対象共有マウント・ポイント数: 4

XML ソース

```
<!-- ================================================================== -->
<!-- = Use the DB2 High Availability Instance Configuration Utility  = -->
<!-- = (db2haicu) XML schema definition, db2ha.xsd, and specify      = -->
<!-- = IBM Tivoli System Automation for Multiplatforms (SA MP)       = -->
<!-- = Base Component as the cluster manager.                        = -->
<!-- ================================================================== -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
            clusterManagerName="TSA"
            version="1.0">

  <!-- ================================================================== -->
  <!-- = Create a cluster domain named db2HAdomain.                    = -->
  <!-- ================================================================== -->
  <ClusterDomain domainName="db2HAdomain">

    <!-- ================================================================== -->
    <!-- = Specify a network quorum device (IP address: 19.126.4.5).   = -->
    <!-- = The IP must be pingable at all times by each of the cluster = -->
    <!-- = domain nodes.                                               = -->
    <!-- ================================================================== -->
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

    <!-- ================================================================== -->
    <!-- = Create a network named db2_public_network_0 with an IP      = -->
    <!-- = network protocol.                                           = -->
    <!-- = This network contains four computers: hasys01, hasys02,     = -->
    <!-- = hasys03, and hasys04.                                       = -->
    <!-- = Each computer has a network interface card called eth0.     = -->
    <!-- = The IP address of eth0 on hasys01 is 19.126.124.30          = -->
    <!-- = The IP address of eth0 on hasys02 is 19.126.124.31          = -->
    <!-- = The IP address of eth0 on hasys03 is 19.126.124.32          = -->
    <!-- = The IP address of eth0 on hasys04 is 19.126.124.33          = -->
    <!-- ================================================================== -->
    <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                     physicalNetworkProtocol="ip">

      <Interface interfaceName="eth0" clusterNodeName="hasys01">
        <IPAddress baseAddress="19.126.124.30"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>
```

```
  <Interface interfaceName="eth0" clusterNodeName="hasys02">
    <IPAddress baseAddress="19.126.124.31"
               subnetMask="255.255.255.0"
               networkName="db2_public_network_0"/>
  </Interface>

  <Interface interfaceName="eth0" clusterNodeName="hasys03">
    <IPAddress baseAddress="19.126.124.32"
               subnetMask="255.255.255.0"
               networkName="db2_public_network_0"/>
  </Interface>

  <Interface interfaceName="eth0" clusterNodeName="hasys04">
    <IPAddress baseAddress="19.126.124.33"
               subnetMask="255.255.255.0"
               networkName="db2_public_network_0"/>
  </Interface>

</PhysicalNetwork>

<!-- ============================================================ -->
<!-- = Create a network named db2_private_network_0 with an IP     = -->
<!-- = network protocol.                                          = -->
<!-- = This network contains four computers: hasys01, hasys02,    = -->
<!-- = hasys03, and hasys04 (same as db2_public_network_0.)       = -->
<!-- = In addition to eth0, each computer has a network interface = -->
<!-- = card called eth1.                                          = -->
<!-- = The IP address of eth1 on hasys01 is 192.168.23.101        = -->
<!-- = The IP address of eth1 on hasys02 is 192.168.23.102        = -->
<!-- = The IP address of eth1 on hasys03 is 192.168.23.103        = -->
<!-- = The IP address of eth1 on hasys04 is 192.168.23.104        = -->
<!-- ============================================================ -->
<PhysicalNetwork physicalNetworkName="db2_private_network_0"
                 physicalNetworkProtocol="ip">

  <Interface interfaceName="eth1" clusterNodeName="hasys01">
    <IPAddress baseAddress="192.168.23.101"
               subnetMask="255.255.255.0"
               networkName="db2_private_network_0"/>
  </Interface>

  <Interface interfaceName="eth1" clusterNodeName="hasys02">
    <IPAddress baseAddress="192.168.23.102"
               subnetMask="255.255.255.0"
               networkName="db2_private_network_0"/>
  </Interface>

  <Interface interfaceName="eth1" clusterNodeName="hasys03">
    <IPAddress baseAddress="192.168.23.103"
               subnetMask="255.255.255.0"
               networkName="db2_private_network_0"/>
  </Interface>

  <Interface interfaceName="eth1" clusterNodeName="hasys04">
    <IPAddress baseAddress="192.168.23.104"
               subnetMask="255.255.255.0"
               networkName="db2_private_network_0"/>
  </Interface>

</PhysicalNetwork>

<!-- ============================================================ -->
<!-- = List the computers (cluster nodes) in the cluster domain.  = -->
<!-- ============================================================ -->
<ClusterNode clusterNodeName="hasys01"/>
<ClusterNode clusterNodeName="hasys02"/>
```

```
    <ClusterNode clusterNodeName="hasys03"/>
    <ClusterNode clusterNodeName="hasys04"/>

  </ClusterDomain>


  <!-- ================================================================= -->
  <!-- = The failover policy specifies the order in which the cluster  = -->
  <!-- = domain nodes should fail over.                                = -->
  <!-- ================================================================= -->
  <FailoverPolicy>
     <NPlusM />
  </FailoverPolicy>


  <!-- ================================================================= -->
  <!-- = Specify all the details of the database partitions            = -->
  <!-- ================================================================= -->
  <DB2PartitionSet>

    <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
       <VirtualIPAddress baseAddress="19.126.124.250"
                         subnetMask="255.255.255.0"
                         networkName="db2_public_network_0"/>
       <Mount filesystemPath="/ha_dpf1/db2inst1/NODE0000"/>
       <Mount filesystemPath="/hafs/NODE0000"/>
       <NPlusMNode standbyNodeName="hasys03" />
    </DB2Partition>

    <DB2Partition dbpartitionnum="1" instanceName="db2inst1">
       <Mount filesystemPath="/ha_dpf1/db2inst1/NODE0001"/>
       <Mount filesystemPath="/hafs/NODE0001"/>
       <NPlusMNode standbyNodeName="hasys04" />
    </DB2Partition>

  </DB2PartitionSet>

</DB2Cluster>
```

db2ha_sample_HADR.xml:

サンプル・ファイル db2ha_sample_DPF_HADR.xml は、新しいクラスター・ドメインを指定するために DB2 高可用性インスタンス構成ユーティリティー (db2haicu) に渡す XML 入力ファイルの一例です。db2ha_sample_HADR.xml は sqllib/samples/ha/xml ディレクトリーにあります。

フィーチャー

db2ha_sample_HADR.xml サンプルは、**db2haicu** と XML 入力ファイルを使用して以下の詳細仕様のクラスター・ドメインを定義する方法を示しています。

- クォーラム装置: ネットワーク
- クラスター内コンピューター (クラスター・ドメイン・ノード) 数: 2
- フェイルオーバー・ポリシー: HADR
- データベース・パーティション数: 1
- 仮想 (サービス) IP アドレス数: なし
- フェイルオーバー対象共有マウント・ポイント数: なし

XML ソース

```
<!-- ================================================================= -->
<!-- = Use the DB2 High Availability Instance Configuration Utility  = -->
<!-- = (db2haicu) XML schema definition, db2ha.xsd, and specify      = -->
<!-- = IBM Tivoli System Automation for Multiplatforms (SA MP)       = -->
<!-- = Base Component as the cluster manager.                        = -->
<!-- ================================================================= -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="db2ha.xsd"
            clusterManagerName="TSA"
            version="1.0">

  <!-- ================================================================= -->
  <!-- = Create a cluster domain named db2HAdomain.                    = -->
  <!-- ================================================================= -->
  <ClusterDomain domainName="db2HAdomain">

    <!-- ================================================================= -->
    <!-- = Specify a network quorum device (IP address: 19.126.4.5).   = -->
    <!-- = The IP must be pingable at all times by each of the cluster = -->
    <!-- = domain nodes.                                               = -->
    <!-- ================================================================= -->
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="19.126.4.5"/>

    <!-- ================================================================= -->
    <!-- = Create a network named db2_public_network_0 with an IP      = -->
    <!-- = network protocol.                                           = -->
    <!-- = This network contains two computers: hasys01 and hasys02.   = -->
    <!-- = Each computer has a network interface card called eth0.     = -->
    <!-- = The IP address of eth0 on hasys01 is 19.126.52.139          = -->
    <!-- = The IP address of eth0 on hasys01 is 19.126.52.140          = -->
    <!-- ================================================================= -->
    <PhysicalNetwork physicalNetworkName="db2_public_network_0"
                     physicalNetworkProtocol="ip">

      <Interface interfaceName="eth0" clusterNodeName="hasys01">
        <IPAddress baseAddress="19.126.52.139"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

      <Interface interfaceName="eth0" clusterNodeName="hasys02">
        <IPAddress baseAddress="19.126.52.140"
                   subnetMask="255.255.255.0"
                   networkName="db2_public_network_0"/>
      </Interface>

    </PhysicalNetwork>

    <!-- ================================================================= -->
    <!-- = Create a network named db2_private_network_0 with an IP     = -->
    <!-- = network protocol.                                           = -->
    <!-- = This network contains two computers: hasys01 and hasys02.   = -->
    <!-- = In addition to eth0, each computer has a network interface  = -->
    <!-- = card called eth1.                                           = -->
    <!-- = The IP address of eth1 on hasys01 is 192.168.23.101         = -->
    <!-- = The IP address of eth1 on hasys02 is 192.168.23.102         = -->
    <!-- ================================================================= -->
    <PhysicalNetwork physicalNetworkName="db2_private_network_0"
                     physicalNetworkProtocol="ip">

      <Interface interfaceName="eth1" clusterNodeName="hasys01">
        <IPAddress baseAddress="192.168.23.101"
                   subnetMask="255.255.255.0"
                   networkName="db2_private_network_0"/>
      </Interface>
```

```
      <Interface interfaceName="eth1" clusterNodeName="hasys02">
        <IPAddress baseAddress="192.168.23.102"
                   subnetMask="255.255.255.0"
                   networkName="db2_private_network_0"/>
      </Interface>

    </PhysicalNetwork>

    <!-- ============================================================= -->
    <!-- = List the computers (cluster nodes) in the cluster domain.   = -->
    <!-- ============================================================= -->
    <ClusterNode clusterNodeName="hasys01"/>
    <ClusterNode clusterNodeName="hasys02"/>

  </ClusterDomain>


  <!-- ================================================================= -->
  <!-- = The failover policy specifies the order in which the cluster  = -->
  <!-- = domain nodes should fail over.                                = -->
  <!-- ================================================================= -->
  <FailoverPolicy>
     <HADRFailover />
  </FailoverPolicy>


  <!-- ================================================================= -->
  <!-- = Specify all the details of the database partitions            = -->
  <!-- ================================================================= -->
  <DB2PartitionSet>
    <DB2Partition dbpartitionnum="0" instanceName="db2inst1" />
  </DB2PartitionSet>


  <!-- ================================================================= -->
  <!-- = List of HADR databases                                        = -->
  <!-- ================================================================= -->
  <HADRDBSet>
    <HADRDB databaseName="HADRDB"
            localInstance="db2inst1"
            remoteInstance="db2inst1"
            localHost="hasys01"
            remoteHost="hasys02" />
  </HADRDBSet>

</DB2Cluster>
```

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の前提条件

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用する前に実行しなければない一連のタスクがあります。

一般

データベース・マネージャー・インスタンスの所有者が **db2haicu** コマンドを実行するには、その前に、root 権限を持つユーザーが **preprpnode** コマンドを実行する必要があります。

preprpnode は、AIX の Reliable Scalable Cluster Technology (RSCT) ファイル・セット、および Linux の RSCT パッケージの一部です。**preprpnode** は、クラスター

内通信のためにノードの初期化操作を行います。**preprpnode** コマンドは、クラスターのセットアップの一部として実行されます。preprpnode について詳しくは、以下を参照してください。

- preprpnode コマンド (AIX)
- RSCT for Linux Technical Reference - preprpnode

RSCT について詳しくは、RSCT Administration Guide - What is RSCT? を参照してください。

また、root 権限を持つユーザーは iTCO_wdt および iTCO_vendor_support のモジュールを無効にする必要があります。

- SUSE では、/etc/modprobe.d/blacklist ファイルに以下の行を追加します。

```
alias iTCO_wdt off
alias iTCO_vendor_support off
```

- RHEL では、/etc/modprobe.conf ファイルに以下の行を追加します。

```
blacklist iTCO_wdt
blacklist iTCO_vendor_support
```

lsmod コマンドを使用して、モジュールが無効であることを確認することができます。

db2haicu を実行する前に、データベース・マネージャー・インスタンスの所有者は、以下のタスクを実行する必要があります。

- クラスターに追加するすべてのマシン上でサービス・ファイルを同期化します。
- クラスター・ドメインの作成に使用するデータベース・マネージャー・インスタンス用の **db2profile** スクリプトを実行します。
- **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。

DB2 高可用性災害時リカバリー (HADR)

HADR 機能を使用する場合は、以下のタスクを実行してください。

- すべての DB2 高可用性災害時リカバリー (HADR) データベースがそれぞれ 1 次およびスタンバイ・データベースの役割で開始され、すべての HADR の 1 次とスタンバイ・データベースのペアがピア状態にあることを確認します。
- すべての HADR データベースの **hadr_peer_window** を 120 秒以上の値に構成します。
- DB2 障害モニターを使用不可にします。

パーティション・データベース環境

高可用性として構成する複数のデータベース・パーティションがある場合には、以下のステップを実行してください。

- クラスター・ドメインに追加するすべてのマシン上で、**DB2_NUM_FAILOVER_NODES** レジストリー変数を構成します。
- (オプション) **db2haicu** を実行する前にデータベースを活動化します。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用したクラスター・ドメインの作成

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) をデータベース・マネージャー・インスタンスに対して初めて実行するとき、**db2haicu** はクラスター・ドメインと呼ばれる、クラスターのモデルを作成します。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) によって自動的に検出されるデータベース・パス:

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を初めて実行するときに、**db2haicu** は、クラスター構成に関連したデータベース構成情報をデータベース・システム内で検索します。

単一データベース・パーティション環境

単一データベース・パーティション環境では、**db2haicu** は以下のパスを自動的に検出します。

- インスタンス・ホーム・ディレクトリー・パス
- 監査ログ・パス
- 監査アーカイブ・ログ・パス
- 同期点マネージャー (SPM) ログ・パス
- DB2 診断ログ (**db2diag** ログ・ファイル) のパス
- 以下のデータベース関連パス
  - データベース・ログ・パス
  - データベース表スペース・コンテナー・パス
  - データベース表スペース・ディレクトリー・パス
  - ローカル・データベース・ディレクトリー

複数データベース・パーティション環境

複数データベース・パーティション環境では、**db2haicu** は以下のパスのみを自動的に検出します。

- データベース・ログ・パス
- データベース表スペース・コンテナー・パス
- データベース表スペース・ディレクトリー・パス
- ローカル・データベース・ディレクトリー

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用したクラスター・ドメインの保守

db2haicu を使用して、クラスター化した環境のクラスター・ドメイン・モデルを変更する場合、データベース・マネージャーは関連する変更を、データベース・マネージャー・インスタンスおよびクラスター構成に伝搬させます。

始める前に

db2haicu を使用してクラスター化された環境を構成する前に、クラスター・ドメインを作成および構成する必要があります。詳しくは、『DB2 高可用性インスタンス

構成ユーティリティー (db2haicu) を使用したクラスター・ドメインの作成』を参照してください。

このタスクについて

db2haicu 保守タスクには、データベースやクラスター・ノードなどのクラスター・エレメントをクラスター・ドメインに追加することや、クラスター・ドメインからエレメントを除去することが含まれています。**db2haicu** 保守タスクには、データベース・マネージャー・インスタンスのフェイルオーバー・ポリシーなどのクラスター・ドメインのエレメントの詳細を変更することも含まれています。

手順

1. **db2haicu** を実行します。

   保守モードで **db2haicu** を実行するとき、以下のような **db2haicu** はクラスター・ドメインで実行できる操作のリストを示します。
   - クラスター・ノード (ホスト名によって識別されるマシン) の追加または除去
   - ネットワーク・インターフェース (ネットワーク・インターフェース・カード) の追加または除去
   - データベース・パーティション (パーティション・データベース環境のみ) の追加または除去
   - DB2 高可用性災害時リカバリー (HADR) データベースの追加または除去
   - 高可用性データベースの追加または除去
   - マウント・ポイントの追加または除去
   - IP アドレスの追加または除去
   - クリティカルでないパスの追加または除去
   - 定期保守用データベース・パーティションおよび HADR データベースの移動
   - 現行のインスタンスのフェイルオーバー・ポリシーの変更
   - クラスター・ドメイン用の新しいクォーラム装置の作成
   - クラスター・ドメインの破棄
2. 実行するタスクを選択し、**db2haicu** が提示する後続の質問に答えます。

タスクの結果

データベース・マネージャーは、クラスター・ドメインにある情報を使用してクラスター・マネージャーに合わせます。**db2haicu** を使用してデータベースおよびクラスター・エレメントを構成するとき、これらのエレメントは、統合され自動化されたクラスター構成、および DB2 高可用性 (HA) フィーチャーによって提供される管理に組み込まれます。**db2haicu** を使用してデータベース・マネージャーのインスタンス構成を変更するとき、データベース・マネージャーが必要なクラスター・マネージャーの構成の変更を行うので、クラスター・マネージャーをその後に呼び出す必要はありません。

次のタスク

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) には、別個の診断ログがありません。データベース・マネージャーの診断ログ **db2diag** ログ・ファイル、

および **db2pd** ツールを使用して、**db2haicu** のエラーを調査、および診断することができます。 詳しくは、『DB2 高可用性インスタンス構成ユーティリティー (db2haicu) のトラブルシューティング』を参照してください。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) のトラブルシューティング

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) には、別個の診断ログがありません。データベース・マネージャーの診断ログ **db2diag** ログ・ファイル、および **db2pd** ツールを使用して、**db2haicu** のエラーを調査、および診断することができます。

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) の制約事項

DB2 高可用性インスタンス構成ユーティリティー (db2haicu) を使用する場合、いくつかの制約事項があります。

- 『ソフトウェアおよびハードウェア』
- 『構成タスク』
- 『使用上の注意』
- 150 ページの『推奨事項』

ソフトウェアおよびハードウェア

- **db2haicu** は IP バージョン 6 をサポートしていません。
- **db2haicu** は、AIX 以外のプラットフォームでは、論理ボリューム・マネージャー (LVM) をサポートしていません。

構成タスク

db2haicu を使用しても以下のタスクは実行できません。

- **db2haicu** を使用しても自動クライアント・リルートを構成できません。
- DB2 Database for Linux, UNIX, and Windows バージョン 9 から IBM Data Server バージョン 9.5 に、またはバージョン 9.5 からそれ以降のバージョンにアップグレードする場合、**db2haicu** を使用してクラスター構成をマイグレーションできません。クラスター構成をマイグレーションするには、以下のステップを実行する必要があります。
  1. 既存のクラスター・ドメインを削除する (存在する場合)。
  2. データベース・サーバーをアップグレードする。
  3. **db2haicu** を使用して新規のクラスター・ドメインを作成する。

使用上の注意

DB2 pureScale環境では db2haicu ユーティリティーはサポートされません。代わりに db2cluster ユーティリティーを使用してクラスター環境を構成します。

クラスター構成および管理アクティビティーを計画する場合、以下の **db2haicu** の使用上の注意を考慮してください。

- **db2haicu** は、通常は root 権限を必要とするいくつかの管理タスクを実行しますが、データベース・マネージャー・インスタンス所有者の特権を使用して実行し

ます。 root ユーザーによって実行される **db2haicu** の初期化によって、**db2haicu** は、インスタンス所有者特権しか持っていないにもかかわらず、必要な構成変更を実行できるようになります。

- ユーザーが新規クラスター・ドメインを作成した場合、**db2haicu** は、新規クラスター・ドメインに指定された名前が有効であることを検証しません。例えば、**db2haicu** は、名前が有効な長さであること、または有効な文字を含んでいること、または既存のクラスター・ドメインと同じ名前でないことを確認しません。
- **db2haicu** は、ユーザーが指定した情報およびクラスター・マネージャーに渡される情報を検証も妥当性検査もしません。例えば、**db2haicu** は、クラスター・オブジェクト名に関するクラスター・マネージャーのすべての制約事項を認識できないので、テキストを、有効な文字または有効な長さなどの点について妥当性検査せずに、クラスター・マネージャーに渡します。
- 新規クラスター・ドメインの作成および構成中に、エラーが発生し、**db2haicu** が失敗した場合、以下のステップを実行する必要があります。
  1. **-delete** パラメーターを指定して **db2haicu** を実行し、部分的に作成されたクラスター・ドメインのリソース・グループを除去します。
  2. **db2haicu** を再び呼び出して、新規クラスター・ドメインを再作成します。
- **db2haicu** は、**-delete** パラメーターを指定して実行すると、現行データベース・マネージャー・インスタンスに関連するリソース・グループを、これらのリソース・グループがロックされているかどうかを確認せずに、直ちに削除します。
- DB2 高可用性災害時リカバリー (HADR) 1 次データベースとスタンバイ・データベースのペアのデータベース・マネージャー・インスタンスに関連するリソース・グループを除去するために、以下のステップを実行します。
  1. 最初に、HADR スタンバイ・データベースのデータベース・マネージャー・インスタンスに対して、 **-delete** パラメーターを指定して **db2haicu** を実行します。
  2. さらに、HADR 1 次データベースのデータベース・マネージャー・インスタンスに対して、 **-delete** パラメーターを指定して **db2haicu** を実行します。
- **db2haicu** を使用して HADR リソース・グループから仮想 IP を削除するには、その仮想 IP が作成されたインスタンスから削除する必要があります。
- **db2haicu** を使用して実行を試みたクラスター操作がタイムアウトした場合、**db2haicu** はエラーを戻しません。クラスター操作がタイムアウトした場合、**db2haicu** 呼び出しの後に診断ログを調べない限り、または後続のクラスター・アクションが失敗し、この後続の失敗の調査中に、オリジナルのクラスター操作がタイムアウトしたと判断しない限り、操作がタイムアウトしたことを認識できません。
- 特定のデータベース・インスタンスのフェイルオーバー・ポリシーをアクティブ/パッシブに変更しようとした時、この構成操作が失敗するような 1 つの条件が存在した場合、このことについて、**db2haicu** はエラーを戻しません。現在オフラインであるマシンをアクティブ なマシンに指定した場合、**db2haicu** は、このマシンをアクティブ・マシンにしませんが、変更が失敗したことを示すエラーを戻しません。
- 共有ディスク構成の場合、**db2haicu** はネストされたマウント構成をサポートしません。 DB2 はディスク・マウント順序を適用しないからです。

- ネットワークにネットワーク・インターフェース・カード (NIC) を追加する場合、**db2haicu** を使用して、サブネット・マスクが異なる NIC を同一ネットワークに追加することはできません。サブネット・マスクが異なる NIC を同一ネットワークに追加する場合は、次の SA MP コマンドを使用します。

  ```
  mkequ <name> IBM.NetworkInterface:<eth0>:<node0>,...,<ethN>:<nodeN>
  ```

推奨事項

以下は、**db2haicu** の使用によるクラスターおよびデータベース・マネージャー・インスタンスの構成に関する推奨事項のリストです。

- /etc/fstab に項目を追加することによって、クラスターの新規マウント・ポイントを追加する場合、 **noauto** オプションを使用して、マウント・ポイントがクラスター内の複数のマシンに自動的にマウントされないようにします。例:

  ```
  dev/vpatha1      /db/svtpdb/NODE0010        ext3  noauto 0 0
  ```

DB2 クラスター・マネージャー API

DB2 クラスター・マネージャー API は、データベース・マネージャーが構成変更をクラスター・マネージャーに通信するために必要な関数のセットを定義します。

サポートされるクラスター管理ソフトウェア

クラスター管理用のソフトウェアによって、 DB2 データベース操作を、クラスターの 1 つのノード上にある障害が生じた 1 次データベースからクラスターの別のノード上にある 2 次データベースに転送できます。

DB2 データベースは、以下のクラスター管理用のソフトウェアをサポートします。

- IBM PowerHA SystemMirror for AIX (旧称 High Availability Cluster Multi-Processing for AIX または HACMP™)

  DB2 データベース製品を使用する PowerHA SystemMirror の構成方法について詳しくは、http://www.redbooks.ibm.com/abstracts/sg247363.html?Open を参照してください。

- Tivoli System Automation for Multiplatforms。

  DB2 データベース製品を使用する Tivoli System Automation の構成方法について詳しくは、http://www.redbooks.ibm.com/abstracts/sg247363.html?Open を参照してください。

- Microsoft Cluster Server (Windows オペレーティング・システム版)

  DB2 データベース製品を使用する Microsoft Cluster Server の構成方法について詳しくは、http://www.redbooks.ibm.com/abstracts/sg247363.html?Open を参照してください。

- Sun Cluster、または VERITAS Cluster Server (Solaris オペレーティング・システム版)。

  Sun Cluster についての情報は、IBM Software Library Web サイト (http://www.ibm.com/software/sw-library/) から入手できる、「DB2 Universal Database™ and High Availability on Sun Cluster 3.X」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。VERITAS Cluster Server についての詳細は、

「DB2 UDB and High Availability with VERITAS Cluster Server」というタイトルのホワイト・ペーパーを参照してください。これは、「IBM Support and downloads」Web サイト (http://www.ibm.com/support/docview.wss?uid=swg21045033>) から入手できます。

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

IBM PowerHA SystemMirror for AIX (旧称 High Availability Cluster Multi-Processing for AIX または HACMP)

IBM PowerHA SystemMirror for AIX は、クラスター管理ソフトウェアです。PowerHA SystemMirror 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ます。あるノードでこのメッセージの送信が停止すると、PowerHA SystemMirror はクラスター内の他のノードでフェイルオーバーを起動し、障害が発生したノードが修復されると、PowerHA SystemMirror はそれを再びクラスターに統合します。

イベントには 2 つのタイプ、つまり、PowerHA SystemMirror 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・コンポーネントのパラメーター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。これは、クラスター内のノードがフェイルしたとき、および PowerHA SystemMirror がクラスター内の他のノードでのフェイルオーバーを開始したときに発生します。リカバリー処理の一部として実行する処理を計画する場合、 PowerHA SystemMirror では 2 つのフェイルオーバー・オプション、つまり「ホット (またはアイドル) スタンバイ」と「相互テークオーバー」を指定できます。

注: PowerHA SystemMirror を使用している時は、**db2iauto** ユーティリティーを以下のように使用して、DB2 インスタンスがブート時に開始されないようにしてください。

```
db2iauto -off InstName
```

ここで *InstName* は、インスタンスのログイン名。

クラスター構成

ホット・スタンバイ 構成では、テークオーバー・ノードではない AIX プロセッサー・ノードは他のワークロードを一切実行していません。*相互テークオーバー* 構成では、テークオーバー・ノードである AIX プロセッサー・ノードは他のワークロードを実行しています。

一般的には、パーティション・データベース環境では、 DB2 データベースは、パーティションが各ノードにある状態で相互テークオーバー・モードで実行されます。 1 つの例外として、カタログ・パーティション部品がホット・スタンバイ構成の一部となるシナリオがあります。

計画時の考慮点の 1 つは、大きいクラスターの管理方法です。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。例えば、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・リレーションシップ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 PowerHA SystemMirror 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。 PowerHA SystemMirror 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。

PowerHA SystemMirror フェイルオーバー・リカバリーでは、物理ノードにリソース・グループ割り当てを事前定義 (カスケード ともいう) することができます。フェイルオーバーのリカバリー手順では、物理ノードにリソース・グループを固定しない、変動割り当て (ローテーティング ともいう) を行うことができます。 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 PowerHA SystemMirror が操作するものです。フェイルオーバーおよび再統合の操作は、作成されるリソース・グループのタイプ、およびリソース・グループに入れられるノード数によって指定されます。

例えば、DB2 データベース・パーティション (論理ノード) を考慮してみます。そのログと表スペースのコンテナーが外部ディスクに置かれ、他のノードがそのディスクにリンクされていた場合、それら他のノードは外部ディスクにアクセスし、(テークオーバー・ノード上にある) データベース・パーティションを再始動することができます。 PowerHA SystemMirror では、このような操作が自動化されます。 PowerHA SystemMirror は、 DB2 インスタンスの主要なユーザー・ディレクトリーが使用する NFS ファイル・システムをリカバリーする場合にも使用できます。

パーティション・データベース環境で DB2 データベースのリカバリーを計画する場合には、その一環として PowerHA SystemMirror 資料を精読してください。概念、計画、インストール、管理の手引きに目を通した後、環境に合ったリカバリー・アーキテクチャーを構築してください。既知の障害点に基づいて識別した、リカバリーを要する各サブシステムについては、必要な PowerHA SystemMirror クラスターとリカバリー・ノード (ホット・スタンバイまたは相互テークオーバー) を識別してください。

ディスクとアダプターはどちらも、外部ディスク構成でミラーリングすることをぜひお勧めします。 PowerHA SystemMirror 用に構成する DB2 物理ノードの場合、ボリューム・グループ上のノードを共有外部ディスクから構成変更できるように配慮する必要があります。相互テークオーバー構成でこのような設定を行う場合、対になったノードが競合することなく互いのボリューム・グループにアクセスできるよう、計画を立てることが必要です。パーティション・データベース環境では、これは、すべてのコンテナー名が、全データベースをまたがって固有でなければならないことを意味します。

固有のものにする 1 つの方法は、名前の一部にデータベース・パーティション番号を含めることです。 SMS または DMS コンテナーを作成するときに、コンテナー

のストリング構文にノード式を指定できます。式を指定するときは、ノード番号をコンテナー名の一部とすることができます。また、追加の引数を指定する場合は、これらの引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、引数「 $N」(ブランク]$N) を使用します。引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 7. コンテナーを作成するための引数 : 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+ 数値]	" $N+1011"	1016
ブランク]$N% 数値]	" $N%3"	2
ブランク]$N+ 数値]% 数値]	" $N+12%13"	4
ブランク]$N% 数値]+ 数値]	" $N%3+20"	22

注:

1. % は剰余演算子です。
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
```

PowerHA SystemMirror 用 DB2 データベース・パーティションの構成

構成が済むと、インスタンス中の各データベース・パーティションは PowerHA SystemMirror 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 PowerHA SystemMirror 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。

ホット・スタンバイまたは相互テークオーバー・ノードのいずれかで PowerHA SystemMirror のフェイルオーバーまたはリカバリーを構成するための補助機構として、スクリプト・ファイルが DB2 Enterprise Server Edition にパッケージされています。そのスクリプト・ファイルは、単一ノード用は `rc.db2pe.ee` であり、複数ノード用は `rc.db2pe.eee` です。これらは `sqllib/samples/hacmp/es` ディレクトリーにあります。適切なファイルを PowerHA SystemMirror クラスター内の各システムの `/usr/bin` にコピーし、ファイル名を `rc.db2pe` に変更してください。

また、`rc.db2pe` の内部から、フェイルオーバーまたは相互テークオーバー構成で DB2 バッファー・プールのサイズをカスタマイズできます。 (バッファー・プール・サイズは、1 つの物理ノードで 2 つのデータベース・パーティションを稼働する場合に適切なリソース割り振りを確保できるように構成できます。)

PowerHA SystemMirror イベント・モニターおよびユーザー定義イベント

特定のノードでプロセスが停止してしまう場合にフェイルオーバー操作を開始することも、ユーザー定義イベントの一例です。イベントは、クラスターのセットアップの一環として、ユーザー定義のイベントとして手動で構成する必要があります。

高可用性を持つ IBM DB2 データベース環境のインプリメンテーションと設計の詳細情報については、IBM Software Library Web サイト (http://www.ibm.com/software/sw-library/) を参照してください。

関連情報:

PowerHA SystemMirror インフォメーション・センター

IBM Tivoli System Automation for Multiplatforms (Linux および AIX)

IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) は、クラスター内の 1 つのデータベース・システムから別のデータベース・システムに、ユーザー、アプリケーション、およびデータを円滑に自動切り替えするための、クラスター管理ソフトウェアです。Tivoli SA MP は、プロセス、ファイル・システム、IP アドレスなどの IT リソースの制御を自動化します。

Tivoli SA MP は、リソースとして認識されるものの可用性を自動的に管理するフレームワークを提供します。リソースの例は次のとおりです。

- 制御するために開始、モニター、および停止スクリプトを作成できるソフトウェアの一部。
- Tivoli SA MP がアクセス権限を付与されている、すべてのネットワーク・インターフェース・カード (NIC)。つまり、Tivoli SA MP は、アクセス権限が付与されている NIC の間で IP アドレスをフローティングすることで、ユーザーが使用するすべての IP アドレスの可用性を管理します。

例えば、DB2 インスタンスと高可用性災害時リカバリー・フィーチャーのどちらにも、開始、停止、およびモニター・コマンドがあります。したがって、これらのリ

ソースを自動的に管理するために、Tivoli SA MP スクリプトを作成できます。密接に関連したリソース (例えば、同じノード上で同時に集合的に実行するもの) をリソース・グループ と呼びます。

DB2 リソース

単一パーティション DB2 環境では、単一の DB2 インスタンスがサーバー上で実行されます。この DB2 インスタンスは、データ (それ自体の実行可能イメージと、インスタンスが所有するデータベース) に対するローカル・アクセス権限を持っています。リモート・クライアントがこの DB2 インスタンスにアクセス可能な場合、未使用の IP アドレスをこの DB2 インスタンスに割り当てる必要があります。

DB2 インスタンス、ローカル・データ、および IP アドレスはすべて、Tivoli SA MP で自動化する必要があるリソースと見なされます。これらのリソースは密接に関連しているため (例えば、同じノード上で同時に集合的に実行する)、リソース・グループと呼ばれます。

リソース・グループ全体はクラスター内の 1 つのノード上で連結されます。フェイルオーバーの場合は、そのリソース・グループ全体が別のノード上で開始されます。

グループ内のリソース間には以下の従属関係が存在します。

- DB2 インスタンスは、ローカル・ディスクより後に開始しなければならない
- DB2 インスタンスは、ローカル・ディスクより前に停止しなければならない
- HA IP アドレスはインスタンスと連結しなければならない

ディスク・ストレージ

DB2 データベースでは、ローカル・データ・ストレージに以下のリソースを使用できます。

- ロー・ディスク (例えば、`/dev/sda1`)
- 論理ボリューム・マネージャー (LVM) によって管理される論理ボリューム
- ファイル・システム (例えば、ext3、jfs)

DB2 データは、全体を 1 つ以上のロー・ディスク、論理ボリューム、ファイル・システムに保管することができます。あるいは、その 3 つすべてを混成したものに保管することもできます。実行可能プログラムは何らかのファイル・システム上に存在する必要があります。

HA IP アドレスに関する DB2 データベースの要件

DB2 データベースには、IP アドレスに関する特別な要件はありません。インスタンスが高可用性であると見なされるために、高可用性のある IP アドレスを定義する必要はありません。ただし、保護された IP アドレス (存在する場合) がデータへのクライアントのアクセス・ポイントとなっており、このアドレスがすべてのクライアントによく知られたものでなければならないことを覚えておくのは重要です。実際、この IP アドレスは、クライアントが **CATALOG TCPIP NODE** コマンドで使用するものと同じであることが推奨されています。

Tivoli SA MP リソース・グループ

IBM Tivoli System Automation for Multiplatforms は、Linux ベースのクラスター内にあるプロセス、アプリケーション、IP アドレス、その他のリソースを自動化することで、高可用性を提供する製品です。IT リソース (IP アドレスなど) を自動化するには、そのリソースを Tivoli SA MP に定義する必要があります。さらに、これらのリソースがすべて少なくとも 1 つのリソース・グループに含まれていなければなりません。これらのリソースを常に同じマシン上でホストする必要がある場合、それらがすべて同じリソース・グループに入っている必要があります。

Tivoli SA MP を使用して管理および自動化するアプリケーションはすべて、リソースとして定義する必要があります。アプリケーション・リソースは通常、汎用リソース・クラス IBM.Application で定義されます。このリソース・クラスには、リソースを定義する複数の属性がありますが、それらのうち、少なくとも以下の 3 つはアプリケーション固有です。

- StartCommand
- StopCommand
- MonitorCommand

これらのコマンドは、スクリプトまたはバイナリー実行可能プログラムです。

DB2 環境での Tivoli SA MP のセットアップ

Tivoli SA MP を DB2 環境で稼働するようセットアップするための詳しい構成情報については、IBM Software Library Web サイト (http://www.ibm.com/software/sw-library/) で「Tivoli System Automation」を検索してください。

Microsoft Failover Clustering のサポート (Windows)

Microsoft Failover Clustering は、Windows オペレーティング・システム上でサーバーのクラスターをサポートします。これはサーバーまたはアプリケーションの障害を自動的に検出および対応し、サーバーのワークロードの平衡を取ることができます。

概要

Microsoft Failover Clustering は、Windows サーバー・オペレーティング・システムのフィーチャーです。高可用性およびデータとアプリケーションのより簡単な管理のために、クラスターへの 2 台のサーバー (DataCenter Server では最大 4 台のサーバー) の接続をサポートします。 Failover Clustering はさらに、自動的にサーバーまたはアプリケーションの障害を検出し、リカバリーすることもできます。それを使用して、サーバーのワークロードを移動して、マシン使用率の平衡を取り、ダウン時間を取らずに計画保守を行うことが可能になります。

以下の DB2 データベース製品は、Microsoft Failover Clustering をサポートします。

- DB2 Connect サーバー製品 (DB2 Connect Enterprise Edition、DB2 Connect Application Server Edition、DB2 Connect Unlimited Edition for iSeries® および DB2 Connect Unlimited Edition for zSeries®)
- DB2 Advanced Enterprise Server Edition

- DB2 Enterprise Server Edition
- DB2 Express Edition
- DB2 Workgroup Server Edition

DB2 Failover Clustering のコンポーネント

クラスターとは、複数のノードの構成であり、各ノードは独立したコンピューター・システムです。クラスターは、ネットワーク・クライアントには単一サーバーのように見えます。

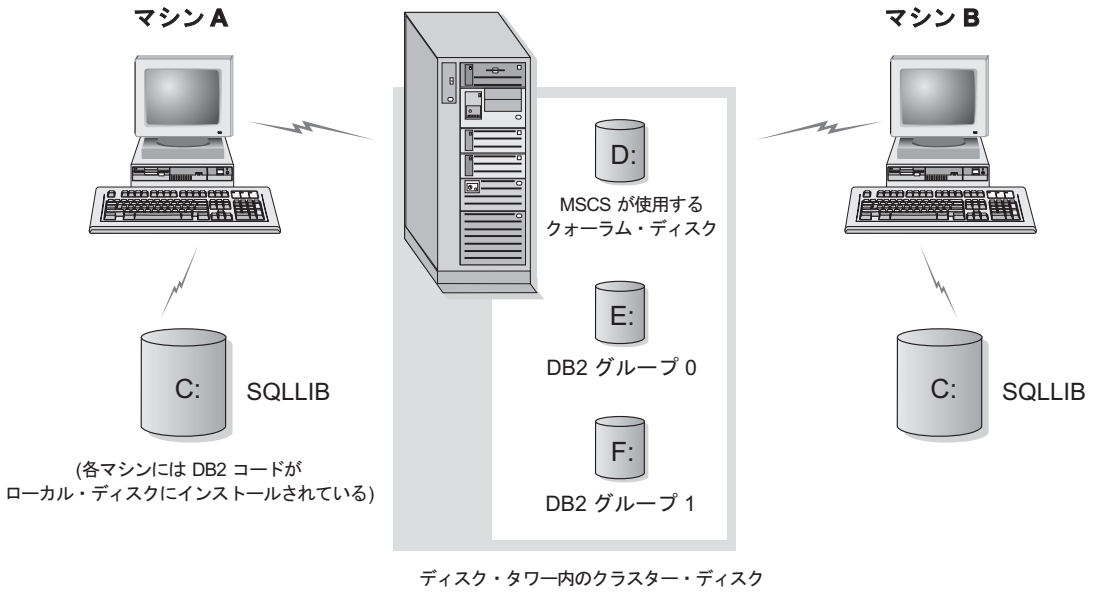


図 4. Failover Clustering の構成の例

Failover Clustering クラスター内のノードは、 1 つ以上の共有ストレージ・バスおよび 1 つ以上の物理的に独立したネットワークを使用して接続されています。サーバーだけを接続していてクライアントをクラスターに接続していないネットワークのことを、*プライベート・ネットワーク*といいます。クライアント接続をサポートするネットワークのことを、*パブリック・ネットワーク*といいます。各ノードには、1 つ以上のローカル・ディスクがあります。各共有ストレージ・バスは、1 つ以上のディスクにアタッチします。共有バスの各ディスクは、一度にクラスターの 1 つのノードだけが所有します。 DB2 ソフトウェアは、ローカル・ディスクに常駐します。 DB2 データベース・ファイル (例えば表、索引、ログ・ファイル) は、共有ディスク上に常駐します。 Microsoft Failover Clustering はクラスター内のロー・パーティションの使用をサポートしないため、 DB2 を構成して Microsoft Failover Clustering 環境でロー・デバイスを使用することは不可能です。

DB2 リソース

Microsoft Failover Clustering 環境では、リソースはクラスタリング・ソフトウェアにより管理されるエンティティーです。例えば、ディスク、IP アドレス、または汎用サービスは、リソースとして管理することができます。 DB2 は、DB2 Server と呼ばれるそれ自体のリソース・タイプを作成することにより Microsoft Failover Clustering と統合します。各 DB2 Server リソースが DB2 インスタンスを管理し、

パーティション・データベース環境での実行時には各 DB2 Server リソースがデータベース・パーティションを管理します。インスタンス名がその DB2 Server リソースの名前になります。ただし、パーティション・データベース環境の場合は、インスタンス名およびデータベース・パーティション (またはノード) 番号の両方によって DB2 Server リソースの名前が構成されます。

オンライン接続前およびオンライン接続後のスクリプト

スクリプトは、DB2 リソースがオンラインになる前、オンラインになった後に実行できます。それらのスクリプトはそれぞれ、オンライン接続前およびオンライン接続後のスクリプトと呼ばれます。オンライン接続前およびオンライン接続後のスクリプトは、DB2 およびシステム・コマンドを実行できる .BAT ファイルです。

DB2 の複数インスタンスを同一のマシン上で実行している場合は、オンライン接続前およびオンライン接続後のスクリプトを使用して、両方のインスタンスが正常に開始されるように構成を調整することができます。フェイルオーバーが発生した場合には、オンライン接続後のスクリプトを使用して、手動でデータベース・リカバリーを実行できます。さらに、オンライン接続後のスクリプトは、任意のアプリケーションまたは DB2 に依存するサービスを開始するためにも使用されます。

DB2 グループ

関連する、または従属するリソースは、リソース・グループに編成されます。 1 つのグループのすべてのリソースは、クラスター・ノード間を 1 つの単位として移動します。例えば、典型的な DB2 単一パーティション・クラスター環境において、以下のリソースを含む DB2 グループがあるでしょう。

1. DB2 リソース。DB2 リソースは DB2 インスタンス (またはノード) を管理します。
2. IP アドレス・リソース。 IP アドレス・リソースを使用すると、クライアント・アプリケーションが DB2 サーバーに接続することが可能になります。
3. Network Name リソース。 Network Name リソースを使用すると、IP アドレスを使用してではなく名前を使用して、クライアント・アプリケーションが DB2 サーバーに接続することが可能になります。 Network Name リソースは、IP アドレス・リソースと従属関係にあります。 Network Name リソースは、オプションです。 (Network Name リソースを構成すると、フェイルオーバー・パフォーマンスに影響するかもしれません。)
4. 1 つ以上の Physical Disk リソース。それぞれの Physical Disk リソースは、クラスター内の共有ディスクを管理します。

注: 同じグループの他のすべてのリソースがオンラインになってからのみ DB2 サーバーが開始されるように、 DB2 リソースは他のすべてのリソースと従属関係になるように構成されます。

フェイルオーバーの構成

2 つのタイプの構成が使用可能です。

- アクティブ/パッシブ
- 相互テークオーバー

パーティション・データベース環境では、必ずしもすべてのクラスターが同タイプの構成を持つ必要はありません。いくつかのクラスターをアクティブ/パッシブを使用するようにセットアップし、他のクラスターを相互テークオーバー用にセットアップすることができます。例えば、DB2 インスタンスが 5 台のワークステーションで構成される場合、それらのマシンのうち 2 台を相互テークオーバー構成を使用するようにセットアップし、2 台をパッシブ・スタンバイ構成を使用するようにセットアップし、残りの 1 台はフェイルオーバー・サポート用に構成しないでおくことが可能です。

アクティブ/パッシブ構成

アクティブ/パッシブ構成では、Microsoft Failover Clustering クラスターの 1 つのマシンがフェイルオーバー・サポート専用となり、他のマシンはデータベース・システムに参加します。データベース・システムに参加しているマシンに障害が起こると、そのマシン上のデータベース・サーバーはフェイルオーバー・マシンで開始します。パーティション・データベース環境のあるマシン上で複数の論理ノードを実行し、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 5 は、アクティブ/パッシブ構成の例を示しています。

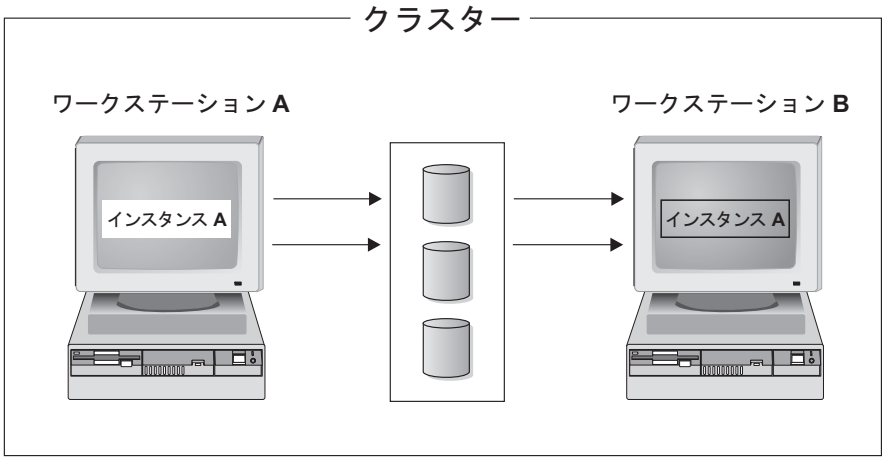


図 5. アクティブ/パッシブ構成

相互テークオーバー構成

相互テークオーバー構成では、両方のワークステーションがデータベース・システムに参加します (つまり、各マシンに実行しているデータベース・サーバーが最低 1 つある)。 Microsoft Failover Clustering クラスターのワークステーションのいずれかに障害が起こると、障害が起きたマシン上のデータベース・サーバーは他のマシンで実行を開始します。相互テークオーバー構成では、あるマシン上のデータベース・サーバーは、別のマシン上のデータベース・サーバーとは別個に障害を起こす場合があります。指定されたどの時点であっても、すべてのデータベース・サーバーはどのマシン上でもアクティブであることができます。 160 ページの図 6 に相互テークオーバー構成の例を示します。

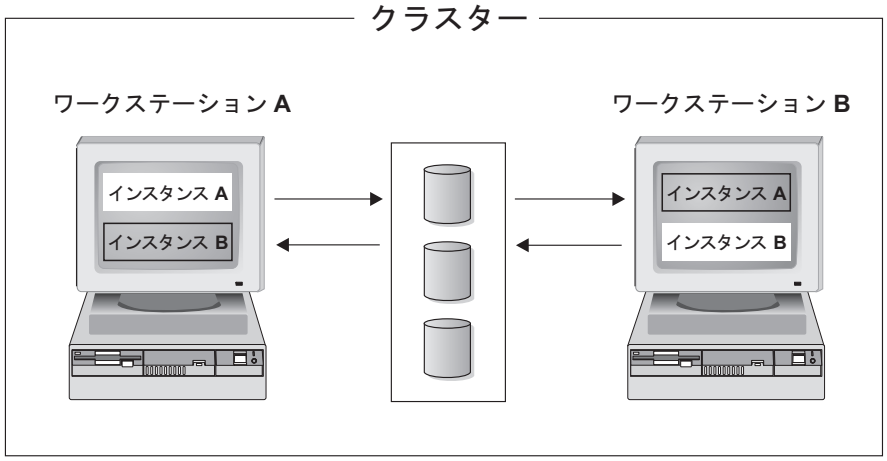


図 6. 相互テークオーバー構成

高可用性を持つ IBM DB2 データベース環境の Windows オペレーティング・システム上でのインプリメンテーションと設計の詳細情報については、IBM Software Library Web サイト (http://www.ibm.com/software/sw-library/) を参照してください。

Windows Server 2008 フェイルオーバー・クラスタリングのサポート

Windows Server 2008 フェイルオーバー・クラスター上で実行するようにパーティション DB2 データベース・システムを構成するには、以下のようにします。

1. IBM Software Library の Web サイト (http://www.ibm.com/software/sw-library/) から入手できる、「Implementing IBM DB2 Universal Database V8.1 Enterprise Server Edition with Microsoft Cluster Server」というタイトルのホワイト・ペーパーで説明されている手順を実行してください。
2. Windows Server 2008 の Failover Clustering フィーチャーの変更のために、以下の追加のセットアップが必要になる可能性があります。
   - Windows Server 2008 フェイルオーバー・クラスターでは、Windows クラスター・サービスは特殊な Local System アカウントの下で実行されますが、Windows Server 2003 では、Windows クラスター・サービスは管理者アカウントの下で実行されます。このことは、DB2 リソース (`db2server.dll`) の操作に影響します。このリソースは、クラスター・サービス・アカウントのコンテキストで実行されます。

     Windows フェイルオーバー・クラスターで **DB2_EXTSECURITY** レジストリー変数が `YES` に設定されている場合、DB2ADMNS グループおよび DB2USERS グループはドメイン・グループでなければなりません。

     Windows フェイルオーバー・クラスター上で複数のパーティション・インスタンスが稼働している場合、INSTPROF パスをネットワーク・パスに設定しなければなりません (例えば、`¥¥`*`NetName`*`¥DB2MSCS-DB2¥DB2PROFS`)。**db2mscs** コマンドを使用して DB2 データベース・システムをクラスタリングする場合、この設定は自動的に行われます。

     Windows Server 2008 フェイルオーバー・クラスターが形成されると、新しいクラスターを表すコンピューター・オブジェクトが Active Directory 内に作成されます。例えば、クラスターの名前が MYCLUSTER の場合は、コンピュー

ター・オブジェクト MYCLUSTER が Active Directory 内に作成されます。ユーザーが複数のパーティション・インスタンスをクラスタリングする場合で、**DB2_EXTSECURITY** レジストリー変数が YES に設定されている場合、このコンピューター・オブジェクトを DB2ADMNS グループに追加しなければなりません。これを行うのは、DB2 リソース DLL が ¥¥*NetName*¥DB2MSCS-DB2¥DB2PROFS パスにアクセスできるようにするためです。例えば、DB2 管理者グループが MYDOMAIN¥DB2ADMNS の場合、コンピューター・オブジェクト MYCLUSTER をこのグループに追加しなければなりません。最後に、コンピューター・オブジェクトを DB2ADMNS グループに追加した後に、クラスター内で両方のノードをリブートしなければなりません。

- Windows Server 2008 Failover Clustering では、「クラスター・ファイル共有リソース」がサポートされなくなりました。代わりにクラスター・ファイル・サーバーを使用します。ファイル共有 (通常のファイル共有) は、クラスター・ファイル・サーバー・リソースに基づくようになります。Microsoft では、クラスター内で作成されたクラスター・ファイル・サーバーはネーム解決に Domain Name System (DNS) を使用する必要があります。複数のパーティション・インスタンスの実行時には、ファイル共有をサポートするためにファイル・サーバー・リソースが必要です。db2mscs.cfg ファイル内で指定される **NETNAME_NAME**、**NETNAME_VALUE**、および **NETNAME_DEPENDENCY** パラメーターの値が、ファイル・サーバー・リソースおよびファイル共有リソースの作成に使用されます。*NetName* は IP アドレスに基づき、この *NetName* は DNS 内になければなりません。例えば、db2mscs.cfg ファイルに以下のパラメーターが含まれている場合には、ファイル共有 ¥¥MSCSV¥DB2MSCS-DB2 が作成されます。

  ```
  ...
  NETNAME_NAME = MSCSN
  NETNAME_VALUE = MSCSV
  ...
  ```

  名前 MSCSV は、DNS で登録されていなければなりません。登録されていない場合は、DNS 解決に失敗した時点で、DB2 クラスター用に作成されたファイル・サーバーまたはファイル共有が失敗します。

Solaris オペレーティング・システムのクラスター・サポート

DB2 では、Solaris オペレーティング・システムで使用可能な Sun Cluster、Veritas Cluster Server (VCS) という 2 つのクラスター・マネージャーがサポートされています。

注: Sun Cluster 3.0 または Veritas Cluster Server を使用している時は、**db2iauto** ユーティリティーを以下のように使用して、DB2 インスタンスがブート時に開始されないようにしてください。

```
db2iauto -off InstName
```

ここで *InstName* は、インスタンスのログイン名。

高可用性

データ・サービスをホストするコンピューター・システムには、数多くの異なるコンポーネントがあり、各コンポーネントにはそれに関連した「平均故障間隔」(MTBF) があります。 MTBF は、コンポーネントが使用可能な状態にある平均時間

です。質の高いハード・ディスクの MTBF は、100 万時間のオーダー (約 114 年) です。これは長期間に思えますが、200 個のディスクのうち 1 つは、 6 カ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 つまたは複数のパブリック・ネットワーク・インターフェース、およびいくつかの共有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 7 で図示されています。

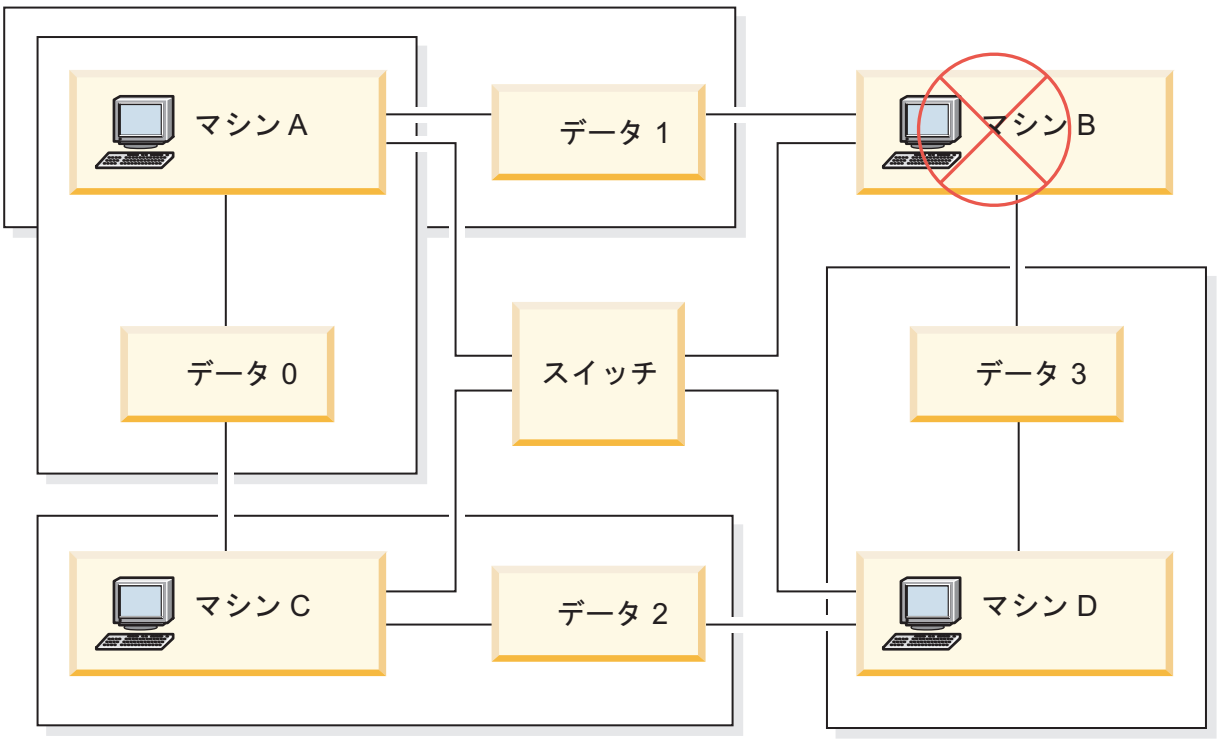


図 7. フェイルオーバー： マシン B のデータに障害が起こると、サービスはそのクラスターの別のマシンに移るため、データは以後もアクセス可能です。

私設ネットワーク・インターフェースは、クラスター内のマシン間でハートビート・メッセージおよび制御メッセージを送信するために使用します。パブリック・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 つ以上の論理パブリック・ネットワーク・インターフェースと一連のディスクが関連付けられています。 HA データ・サービスのクライアントは、 TCP/IP 経由でデータ・サービスの論理ネットワーク・インターフェースにのみ接続します。フェイルオーバーが起き

ると、データ・サービスは、論理ネットワーク・インターフェースおよびディスクのセットとともに、別のマシンに移動します。

HA クラスターの利点の 1 つは、サポート・スタッフの援助なしでデータ・サービスをリカバリーできることと、いつでもそれを行えることです。もう 1 つの利点は、冗長度です。マシン自体を含め、クラスター内のすべての部分に冗長度があります。クラスターは、どんな Single Point of Failure があっても、存続することができます。

高可用性データ・サービスはそれぞれ性質がかなり異なる場合がありますが、いくつかの共通要件があります。高可用性データ・サービスのクライアントは、データ・サービスがどのマシン上にあっても、データ・サービスのネットワーク・アドレスおよびホスト名が同じであり、同じ方法で要求を出すことができると予期します。

可用性の高い Web サーバーにアクセスしている Web ブラウザーのことを考えてみてください。要求は URL とともに発行されます。 URL には、ホスト名と、web サーバー上のファイルへのパスの両方が含まれます。ブラウザーは、 web サーバーのフェイルオーバー後もホスト名とパスが同じであると予期します。ブラウザーが web サーバーからファイルをダウンロードしているときに、サーバーがフェイルオーバーされると、ブラウザーは要求を再発行する必要があります。

データ・サービスの可用性は、データ・サービスがユーザーから使用可能である時間の合計により測定されます。可用性の最も一般的な測定単位は、「アップ時間」のパーセンテージです。これはしばしば、複数の「9」によって示されます。

```
99.99% => サービスは、1 年で (最大) 52.6 分ダウンする。
99.999% => サービスは、1 年で (最大) 5.26 分ダウンする。
99.9999% => サービスは、1 年で (最大) 31.5 秒ダウンする。
```

HA クラスターを設計し、テストするときは、

1. クラスターの管理者がシステムに精通しており、フェイルオーバーの発生時に起きることを知っていることを確かめます。
2. クラスターの各部分に正しく冗長度があり、障害が起きたときに素早く置き換えられることを確かめます。
3. 制御環境でテスト・システムに障害を起こし、システムが毎回正確にフェイルオーバーされることを確かめます。
4. それぞれのフェイルオーバーの理由を記録します。これはそれほど頻繁に起きることではありませんが、クラスターを不安定にする問題に対処するために重要です。例えば、クラスターの一部が 1 カ月に 5 回フェイルオーバーを起こした場合、その理由を調べてそれを修正してください。
5. フェイルオーバーが起きたときに、そのことがクラスターのサポート・スタッフに知らされることを確かめます。
6. クラスターが過負荷にならないようにします。フェイルオーバーの後で、残りのシステムが引き続き許容レベルでワークロードを処理できることを確かめてください。
7. よく障害の起きるコンポーネント (ディスクなど) をチェックし、問題が起きる前に置き換えます。

フォールト・トレランス

データ・サービスの可用性を上げる別の方法は、フォールト・トレランスです。フォールト・トレラント・マシンには、すべての冗長度が組み込まれており、 CPU やメモリーを含む任意の部分で起きる単一の障害に対処することができます。フォールト・トレラント・マシンは、すき間産業で最も良く使用され、通常、このマシンをインプリメントするには費用がかかります。地理的に別の場所にあるマシンを含む HA クラスターには、これらの場所のサブセットだけに影響を与える災害から、リカバリーすることができるという利点があります。

HA クラスターは可用性を上げる最も一般的な方法です。なぜなら、拡張が容易で、使いやすく、そして比較的インプリメントするために費用がかからないからです。

Sun Cluster 3.0 (およびそれ以降) のサポート:

DB2 データベース・ソリューションを Solaris オペレーティング・システム・クラスターで実行する計画の場合、Sun Cluster 3.0 を使用してクラスターを管理することができます。高可用性エージェントは DB2 データベースと Sun Cluster 3.0 の間の仲介者としての役割を果たします。

このトピックでの Sun Cluster 3.0 のサポートに関する記述は、Sun Cluster 3.0 以降のバージョンにも適用されます。

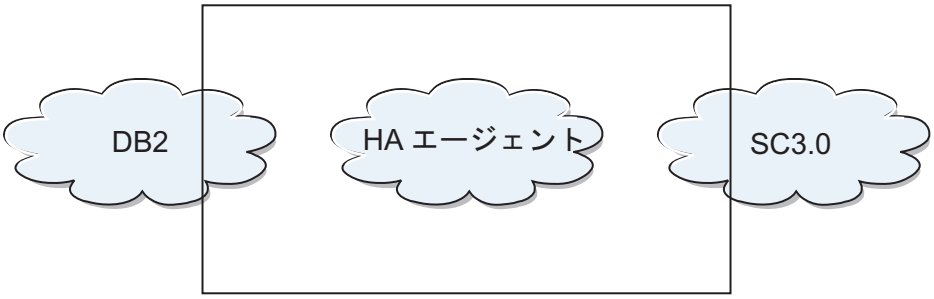


図 8. DB2 データベース、Sun Cluster 3.0、および高可用性: DB2 データベース、Sun Cluster 3.0 と高可用性エージェントとの間の関係

フェイルオーバー

Sun Cluster 3.0 は、アプリケーション・フェイルオーバーを使用可能にすることにより高可用性を提供します。各ノードは周期的にモニターされ、クラスター・ソフトウェアは、障害が起きたプライマリー・ノードから指定されたセカンダリー・ノードへ自動的にクラスター対応アプリケーションを再配置します。フェイルオーバーが発生すると、クライアントにはサービスに短い中断があるかもしれず、サーバーに再接続する必要があるかもしれません。とはいえ、クライアントは、アプリケーションやデータにアクセスするために使用している物理サーバーには気付かないでしょう。クラスター内の別のノードがプライマリー・ノードの障害時に自動的にホストのワークロードを処理することを許可することにより、Sun Cluster 3.0 では、大幅にダウン時間を削減し、生産性を上げることができます。

マルチホスト・ディスク

Sun Cluster 3.0 には、マルチホスト・ディスク装置が必要です。これは、ディスクが複数のノードに一度に接続されることがあるということを意味します。 Sun Cluster 3.0 環境では、マルチホスト記憶装置により、ディスク装置が高可用性になります。マルチホスト記憶装置にあるディスク装置は、代替サーバー・ノードを経由するデータへの物理パスがあるため、単一ノードの故障を容認できます。マルチホスト・ディスクは、プライマリー・ノードを経由してグローバルにアクセス可能です。クライアント要求があるノードを経由してデータにアクセスしており、そのノードに障害が起こった場合、その要求は、同じディスクへの直接接続を持つ別のノードへ切り替わります。ボリューム・マネージャーは、マルチホスト・ディスクのデータ冗長化のために、ミラーリング構成または RAID5 構成を提供します。現行では、Sun Cluster 3.0 はボリューム・マネージャーとして、 Solstice DiskSuite および VERITAS Volume Manager をサポートします。マルチホスト・ディスクをディスク・ミラーおよびストライピングと結合させるなら、ノード障害および個々のディスク障害の両方を保護します。

グローバル・デバイス

グローバル・デバイスは、装置の物理的な位置にかかわらず、任意のノードからクラスター内の任意の装置への可用性の高いアクセスを、クラスター全体にわたり提供するために使用されます。すべてのディスクは、グローバル・ネーム・スペース内に、割り当てられたデバイス ID (DID) を付けて組み込まれ、グローバル・デバイスとして構成されます。そのため、ディスク自体はすべてのクラスター・ノードから可視状態になります。

ファイル・システムおよびグローバル・ファイル・システム

クラスター・ファイル・システムまたはグローバル・ファイル・システムは、(1 つのノード上の) カーネルと、 (1 つ以上のディスクへの物理接続を持つノード上の) 元となるファイル・システム・ボリューム・マネージャーとの間のプロキシーです。クラスター・ファイル・システムは、1 つ以上のノードへの物理接続を持つグローバル・デバイスに依存しています。クラスター・ファイル・システムは、元となるファイル・システムおよびボリューム・マネージャーから独立しています。現行では、クラスター・ファイル・システムは、 Solstice DiskSuite または VERITAS Volume Manager のいずれかを使用して UFS 上に作成できます。ディスク上のファイル・システムがクラスター・ファイル・システムとしてグローバルにマウントされた場合にのみ、データは、すべてのノードに対して使用可能になります。

デバイス・グループ

すべてのマルチホスト・ディスクは、Sun Cluster のフレームワークで制御しなければなりません。 Solstice DiskSuite または VERITAS Volume Manager で管理されるディスク・グループは、まずマルチホスト・ディスク上に作成されます。その後、それらは Sun Cluster ディスク・デバイス・グループに登録されます。ディスク・デバイス・グループは、一種のグローバル・デバイスです。マルチホスト・デバイス・グループは、高い可用性を持っています。現行でデバイス・グループを管理しているノードに障害が起こった場合には、ディスクは、代替パスを経由してアクセス可能になります。デバイス・グループを管理しているノードの障害により、リカバリーおよび整合性チェックを行うために必要な時間を除いては、デバイス・グループへ

のアクセスに影響しません。その時間中は、すべての要求は、システムがデバイス・グループを使用可能にするまでブロックされます (このことはアプリケーションには知らされません)。

リソース・グループ・マネージャー (RGM)

RGM は、高可用性のための機構を備え、それぞれのクラスター・ノード上でデーモンとして実行されます。それは、すでに構成されているポリシーに基づいて、選択したノード上のリソースを自動的に開始および停止します。RGM の使用により、ノード障害の時にリソースを高可用性にしたり、影響を受けたノード上のリソースを停止することによりリブートし、別のノード上で開始したりすることができます。さらに RGM は、リソース障害を検出したり障害が起きたリソースを別のノード上に再配置することができる特定のリソースのモニターを、自動的に開始および停止できます。

データ・サービス

データ・サービスという用語は、単一サーバー上ではなく、クラスター上で実行するように構成されているサード・パーティー・アプリケーションを説明するために用いられます。データ・サービスには、アプリケーションを開始、停止、およびモニターするアプリケーション・ソフトウェアおよび Sun Cluster 3.0 ソフトウェアが含まれます。 Sun Cluster 3.0 は、クラスター内でアプリケーションを制御またはモニターするために使用されるデータ・サービス・メソッドを提供します。これらのメソッドは、リソース・グループ・マネージャー (RGM) の制御下で実行されます。 RGM は、メソッドを使用してクラスター・ノード上のアプリケーションを開始、停止、およびモニターします。それらのメソッドは、クラスター・フレームワーク・ソフトウェアおよびマルチホスト・ディスクとともに、アプリケーションが高可用性を備えたデータ・サービスになるようにします。高可用性を備えたデータ・サービスとして、それらのメソッドは、単一の障害の後にそのクラスター内で発生するかなりの数のアプリケーション中断を防ぎます。それは、障害がノード上、インターフェース・コンポーネント上、またはアプリケーション自体のいずれで起きたかにかかわらず行われます。さらに RGM は、ネットワーク・リソース (論理ホスト名および共有アドレス) およびアプリケーション・インスタンスを含む、クラスター内のリソースを管理します。

リソース・タイプ、リソース、およびリソース・グループ

リソース・タイプは、以下のもので構成されます。

1. クラスター上で実行されるソフトウェア・アプリケーション。
2. アプリケーションをクラスター・リソースとして管理するために RGM によりコールバック・メソッドとして使用されるコントロール・プログラム。
3. クラスターの静的構成の一部を形成するプロパティーのセット。

RGM は、リソース・タイプ・プロパティーを使用して、特定のタイプのリソースを管理します。

リソースは、プロパティーおよびそのリソース・タイプの値を継承します。それは、クラスター上で実行中の基礎となるアプリケーションのインスタンスです。各インスタンスには、クラスター内での固有の名前が必要です。各リソースは、リソース・グループ内に構成される必要があります。 RGM は、すべてのリソースを、オンラインおよびオフラインで同じノードの 1

つのグループにまとめます。 RGM がリソース・グループをオンラインまたはオフラインにする時には、 RGM はそのグループの個々のリソース上でコールバック・メソッドを呼び出します。

リソース・グループが現行でオンラインになっているノードのことを、そのリソース・グループのプライマリー・ノードまたはそのプライマリーと呼びます。それぞれのプライマリーが、リソース・グループのマスターになります。各リソース・グループは、関連した Nodelist プロパティーがあり、それは、クラスター管理者により設定され、リソース・グループのプライマリーまたはマスターである可能性があるものをすべて識別します。

VERITAS Cluster Server のサポート:

DB2 データベース・ソリューションを Solaris オペレーティング・システム・クラスターで実行する計画の場合、VERITAS Cluster Server を使用してクラスターを管理することができます。 VERITAS Cluster Server は、異機種混合環境の広範なアプリケーションを管理することができます。このサーバーは、ストレージ・エリア・ネットワーク (SAN) および従来型のクライアント/サーバー環境の両方において、最大 32 ノードのクラスターをサポートします。

ハードウェア要件

以下のものは、現行で VERITAS Cluster Server がサポートするハードウェアのリストです。

- サーバー・ノード:
  - Solaris 2.6 以降を実行する Sun Microsystems の SPARC/Solaris サーバー (最小で 128 MB の RAM)。
- ディスク装置:
  - EMC Symmetrix、IBM Enterprise Storage Server®、HDS 7700 および 9xxx、Sun T3、 Sun A5000、Sun A1000、Sun D1000 および VCS 2.0 以降がサポートするその他の任意のディスク装置。 VERITAS 担当者はどのディスク・サブシステムがサポートされているかを確認できますし、VCS 資料を参照することもできます。
  - 典型的な環境では、DB2 バイナリー用にミラーリングされた専用ディスクが (各クラスター・ノードに) 必要であり、 DB2 データ用にノード間の共有ディスクが必要です。
- ネットワーク相互接続:
  - パブリック・ネットワーク接続用に、IP ベースのアドレッシングをサポートする任意のネットワーク接続が必要です。
  - ハートビート接続 (クラスターへの内部接続) 用に、冗長化されたハートビート接続が必要です。この要件は、サーバーごとに 2 つのイーサネット・コントローラーを追加して使用するか、サーバーごとに 1 つのイーサネット・コントローラーを追加し、クラスターごとに 1 つの共有 GABdisk を使用することにより満たすことができます。

ソフトウェア要件

以下の VERITAS ソフトウェア・コンポーネントが検定済みの構成です。

- VERITAS Volume Manager 3.2 以降、VERITAS File System 3.4 以降、VERITAS Cluster Server 2.0 以降。

- DB Edition for DB2 for Solaris 1.0 以降。

VERITAS Cluster Server にはボリューム・マネージャーは必要ないため、容易にインストール、構成、および管理をするために VERITAS Volume Manager を使用することを強くお勧めします。

フェイルオーバー

VERITAS Cluster Server は、アプリケーション・フェイルオーバーを使用可能にすることによりアプリケーション・サービス (DB2 データベースなど) の可用性を管理する、可用性クラスタリング・ソリューションです。それぞれの個々のクラスター・ノードおよびそれに関連したソフトウェア・サービスの状態は、定期的にモニターされます。アプリケーション・サービス (この場合では、DB2 データベース・サービス) を中断してしまうような障害が発生した時、VERITAS Cluster Server または VCS HA-DB2 Agent のいずれかが障害を検出し、自動的にサービスをリストアするためのステップに入ります。サービスのリストアのために取られるステップには、同じノード上で DB2 データベース・システムを再始動したり、そのクラスターの別のノードに DB2 データベース・システムを移動し、そのノードで再始動することが含まれるかもしれません。アプリケーションを新規のノードにマイグレーションする必要がある場合、VERITAS Cluster Server は、アプリケーションに関連したすべてのもの (すなわち、ネットワーク IP アドレス、元となるストレージの所有権) を新規ノードに移動しますので、サービスが実際には別のノードで実行されていることをユーザーは気付かないでしょう。ユーザーは、以後も同一の IP アドレスを使用してサービスにアクセスできますが、それらのアドレスは、異なるクラスター・ノードを指すようになります。

VERITAS Cluster Server でフェイルオーバーが発生すると、ユーザーはサービスの中断に気付くかもしれませんし、気付かないかもしれません。これは、そのアプリケーション・サービスにおいてクライアントが使用している接続のタイプ (ステートフルまたはステートレス) によります。 (DB2 データベースのような) ステートフル接続でのアプリケーション環境では、ユーザーはサービスの短い中断に気付き、フェイルオーバーが完了した後に再接続する必要があるかもしれません。 (NFS のような) ステートレス接続でのアプリケーション環境では、ユーザーはサービスの短い遅延に気付くかもしれませんが、一般的には中断には気付かず、またログオンする必要もないでしょう。

クラスター・ノード間で自動的にマイグレーションできるサービスとしてアプリケーションをサポートすることにより、 VERITAS Cluster Server は、ダウン時間を短縮できるだけでなく、計画されたダウン時間 (保守やアップグレードのための) に関連した停止の継続時間を短縮することもできます。フェイルオーバーは手動で開始することもできます。ハードウェアまたはオペレーティング・システムのアップグレードを特定のノードで実施する必要がある場合、 DB2 データベース・システムをそのクラスターの別のノードにマイグレーションし、アップグレードを行い、その後 DB2 データベース・システムを元のノードにマイグレーションできます。

このタイプのクラスタリング環境で使用するために推奨されているアプリケーションは、破損に耐えられるものでなければなりません。破損に耐えられるアプリケーションは、コミット済みデータの整合性を維持しながら予期し

ない破損からリカバリーすることができます。破損に耐えられるアプリケーションはしばしば、クラスター・フレンドリーな アプリケーションと言われます。 DB2 データベース・システムは、破損に耐えられるアプリケーションです。

共有ストレージ

VCS HA-DB2 Agent とともに使用した場合、Veritas Cluster Server は、共有ストレージが必要です。共有ストレージとは、クラスター内の複数のノードへの物理接続を持つストレージです。共有ストレージに常駐のディスク装置は、障害があっても、ディスク装置への物理パスが 1 つ以上の代替クラスター・ノードを経由して存在するため、ノード障害に耐えることができます。

VERITAS Cluster Server の制御を通して、クラスター・ノードは、「ディスク・グループ」と呼ばれる論理構成を通して共有ストレージにアクセスできます。ディスク・グループは、論理的に定義されたストレージ・デバイス (その所有権はクラスターのノード間で自動的にマイグレーションされる) の集合を表します。ディスク・グループは、任意の特定の時に、単一のノードにのみインポートすることができます。例えば、Disk Group A が Node 1 にインポートされた後、Node 1 に障害が起きた場合、 Disk Group A を障害が起きたノードからエクスポートし、そのクラスターの新規ノードにインポートすることができます。 VERITAS Cluster Server は、単一クラスター内の複数のディスク・グループを同時に制御することができます。

ディスク・グループ定義を許可することに加え、ボリューム・マネージャーは、ミラーリングまたは RAID 5 を使用して、共有ストレージに冗長データ構成を提供することができます。 VERITAS Cluster Server は、VERITAS Volume Manager および Solstice DiskSuite を論理ボリューム・マネージャーとしてサポートします。共有ストレージをディスク・ミラーおよびストライピングと結合させるなら、ノード障害および個々のディスクまたはコントローラー障害の両方を保護できます。

VERITAS Cluster Server Global Atomic Broadcast (GAB) および Low Latency Transport (LLT)

ノードがハードウェアおよびソフトウェア状況に関する情報を交換したり、クラスターのメンバーシップを追跡したり、この情報をすべてのクラスター・ノードにわたり同期できるようにするために、ノード間通信メカニズムが、クラスター構成内になければなりません。 Low Latency Transport (LLT) 全体にわたり実行される Global Atomic Broadcast (GAB) 機能は、VERITAS Cluster Server がこのことを行うために用いられる高速で待ち時間が少ないメカニズムを備えています。 GAB は、各クラスター・ノード上にカーネル・モジュールとしてロードされ、確実にすべてのノードが同時に状況更新情報を入手するようにする Atomic Broadcast のメカニズムを提供します。

カーネル間通信能力の効力により、LLT はクラスター・ノード間で交換および同期する必要があるすべての情報に対して、高速で、待ち時間が少ないトランスポートを提供します。 GAB は、LLT の上で実行されます。VERITAS Cluster Server は、IP をハートビート機構として使用しませんが、さらに他の 2 つの信頼性のあるオプションを備えています。 LLT とともに使用する GAB は、ハートビート機構として作動するように構成でき

ますし、 GABdisk はディスク・ベースのハートビートとして構成できます。ハートビートには冗長化された接続を使用しなければなりません。それらの接続は、2 本のクラスター・ノード間のプライベート・イーサネット接続か、 1 本のプライベート・イーサネット接続と 1 本の GABdisk 接続のいずれかになります。 2 本の GABdisk はサポートされておりません。それはノード間のクラスター状況の交換には、プライベート・イーサネット接続が必要だからです。

GAB または LLT についての詳細、またはそれらを VERITAS Cluster Server 構成で構成する方法については、「VERITAS Cluster Server 2.0 User's Guide for Solaris」をご覧ください。

バンドルおよびエンタープライズ・エージェント

エージェントとは、特定のリソースまたはアプリケーションの可用性を管理するために設計されたプログラムです。エージェントが開始されると、それは VCS から必要な構成情報を取得し、その後周期的にリソースまたはアプリケーションをモニターし、VCS を状況とともに更新します。一般にはエージェントは、リソースをオンラインにしたり、リソースをオフラインにしたり、あるいはリソースをモニターし、4 種類のサービス (開始、停止、モニターおよびクリーン) を提供するために使用されます。開始および停止は、リソースをオンラインまたはオフラインにするために使用され、モニターは、特定のリソースまたはアプリケーションの状況を見るためにそれをテストするために使用され、クリーンはリカバリー処理で使用されます。

VERITAS Cluster Server の一部として様々なバンドル・エージェントが組み込まれており、 VERITAS Cluster Server のインストール時にインストールされます。バンドル・エージェントは、通常クラスター構成にある定義済みリソース・タイプ (すなわち、IP、マウント、プロセス、および共有) を管理する VCS プロセスで、それらはクラスターのインストールおよび構成をかなりの程度単純化する上で役立ちます。 VERITAS Cluster Server には、20 を超えるバンドル・エージェントがあります。

エンタープライズ・エージェントは、どちらかというと DB2 データベース・アプリケーションのような特定のアプリケーションに焦点が向けられています。 VCS HA-DB2 Agent は、エンタープライズ・エージェントと考えることができ、それは、VCS とのインターフェースを VCS Agent フレームワークを通して行います。

VCS リソース、リソース・タイプ、およびリソース・グループ

リソース・タイプとは、モニターされる VCS クラスター内部のリソースを定義するために使用されるオブジェクト定義のことをいいます。リソース・タイプには、リソース・タイプ名およびそのリソースに関連していて高可用性の観点からみて顕著なプロパティーのセットが含まれます。リソースは、プロパティーおよびそのリソース・タイプの値を継承し、リソース名は、クラスター全体にわたり固有のものでなければなりません。

リソースには、2 つのタイプがあります。永続および標準 (非永続) です。永続リソースとは、モニターされてはいても VCS によりオンラインまたはオフラインにならない、ネットワーク・インターフェース・コントローラー (NIC) のようなリソースのことをいいます。標準リソースとは、オンラインおよびオフラインの状況が VCS により制御されるもののことをいいます。

モニターされる最低レベルのオブジェクトはリソースであり、様々なリソース・タイプがあります (すなわち、共有、マウント)。各リソースは、リソース・グループに構成される必要があり、 VCS は特定のリソース・グループのすべてのリソースをまとめてオンラインおよびオフラインにします。リソース・グループをオンラインまたはオフラインにするために、 VCS はそのグループの各リソースのために開始または停止メソッドを呼び出します。リソース・グループには、2 つのタイプがあります。フェイルオーバーおよび並列です。可用性の高い DB2 データベース構成では、それがパーティション・データベース環境になっていてもいなくても、フェイルオーバー・リソース・グループを使用します。

「プライマリー」または「マスター」ノードとは、リソースをホストする可能性があるノードのことをいいます。 systemlist と呼ばれるリソース・グループ属性は、クラスター内のどのノードが特定のリソース・グループのプライマリーになれるかを指定するために使用されます。 2 つのノードがあるクラスターでは、たいてい両方のノードは systemlist に含まれますが、いくつもの高可用性を持つアプリケーションを管理するようなより大規模なマルチノード・クラスターでは、 (最低のレベルのリソースにより定義される) 特定のアプリケーション・サービスが特定のノードに決してフェイルオーバーされないようにすることを保証するための要件があるかもしれません。

リソース・グループ間に従属関係を定義することができ、VERITAS Cluster Server は、さまざまなリソース障害の影響を評価する上で、さらにリカバリーを管理する上でこのリソース・グループ従属関係階層に依存しています。例えば、リソース・グループ DB2 がすでに正常に開始済みでないかぎりリソース・グループ ClientApp1 がオンラインにならない場合には、リソース・グループ ClientApp1 は、リソース・グループ DB2 に依存していると考えられます。

パーティション・データベース環境におけるクロックの同期化

データベース・パーティション・サーバー全体で相対的な同期がとれたシステム・クロックを維持し、データベース操作がスムーズに行われ、順方向リカバリー性に制限が加わらないようにします。データベース・パーティション・サーバー間の時間差にトランザクションの操作および通信遅延時間を加えたものは、 *max_time_diff* (ノード間最大時間差) データベース・マネージャー構成パラメーターの値より小さくしてください。

ログ・レコードのタイム・スタンプがトランザクションの順序を確実に示すようにするために、パーティション・データベース環境の 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、および SQLOGCTL.LFH ファイルに格納されている仮想タイム・スタンプを使用します。しかし、システム・クロックを進めると、ログ・クロックはそれに合わせて自動的に進みます。システム・クロックを遅らせることはできますが、ログのクロックを遅らせることはできず、システム・クロックをこの時刻に合わせるまでは、ずっと 進んだ時刻のままです。このとき、両方のクロックは同期しています。このことは、データベース・ノードで発生するシステム・クロック・エラーが短期間であっても、データベース・ログのタイム・スタンプではその影響が長く続く場合があることを意味します。

仮説的な例として、データベース・パーティション・サーバー A のシステム・クロックを、2003 年であるのを間違って 2005 年 11 月 7 日に設定したものとします。また、その誤りは訂正されましたが、そのデータベース・パーティション・サーバーのデータベース・パーティションで更新トランザクションがコミットされた後 であったとします。そのデータベースが連続的に使用されていてその間で定期的に更新されていると、2003 年 11 月 7 日から 2005 年 11 月 7 日までの間の任意の時点は、ロールフォワード・リカバリーでは実際には処理不能になります。データベース・パーティション・サーバー A でコミット (COMMIT) が実行されると、データベース・ログのタイム・スタンプは 2005 に設定され、データベース・ログのクロックは 2005 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 (2003 年 11 月 7 日) を超えた最初のタイム・スタンプで操作は停止します。

DB2 ではシステム・クロックに対する更新を制御することはできませんが、*max_time_diff* データベース・マネージャー構成パラメーターを指定しておくと、次のような問題が発生するのを防ぐことができます。

- このパラメーターに指定できる値は、1 分から 24 時間までです。
- 非カタログ・パーティションに最初の接続要求が出されると、データベース・パーティション・サーバーはその時間をデータベースのカタログ・パーティションに送信します。カタログ・パーティションはその時間を受け取ると、接続を要求するデータベース・パーティションの時間と自分自身の時間が、*max_time_diff* パラメーターで指定された範囲内であることをチェックします。この範囲を超えると、接続は拒否されます。
- 更新トランザクションがデータベース内の 3 つ以上のデータベース・パーティション・サーバーに関係している場合は、トランザクションは関係するデータベース・パーティション・サーバー間で同期がとれていることを確認した後、更新をコミットします。複数のデータベース・パーティション・サーバーの時間差が、*max_time_diff* で指定した値を超えていると、トランザクションはロールバックされ、他のデータベース・パーティション・サーバーへ正しくない時間が伝搬されるのを防ぎます。

クライアント/サーバーのタイム・スタンプの変換

ここでは、クライアント/サーバー環境でのタイム・スタンプの生成について説明します。

- ロールフォワード操作のために現地時間を指定する場合は、戻されるすべてのメッセージも現地時間となります。

  注: すべての時刻は、サーバー上および (パーティション・データベース環境では) カタログ・データベース・パーティション上で変換されます。

- タイム・スタンプ・ストリングは、サーバー上で GMT に変換されますので、時刻はクライアントの時間帯ではなく、サーバーの時間帯を表します。クライアントがサーバーとは異なる時間帯にある場合には、サーバーの現地時間を使用する必要があります。
- 夏時間調整時のため、タイム・スタンプ・ストリングの時刻変更が近づいている場合には、その停止時刻が時刻変更の前か後かを知ることは、正確にその時刻を指定するために重要です。

第 5 章 高可用性ソリューションの管理と保守

いったん、作成、構成、開始した DB2 データベースの高可用性ソリューションが稼働をはじめると、すぐに実行すべきアクティビティーがあります。クライアント・アプリケーションから使用可能な状態に保つために、その DB ソリューションをモニター、保守、および修理することが必要です。

手順

データベース・システムを稼働させると、以下のような事柄をモニターして、それに応答する必要があります。

1. ログ・ファイルを管理します。

   ログ・ファイルが大きくなってくると、アーカイブが必要となります。また一部のログ・ファイルは、リストア操作で使用できるようにコピーまたは移動が必要です。
2. 保守アクティビティーを実行します。
   - ソフトウェアのインストール
   - ハードウェアのアップグレード
   - データベース表の再編成
   - データベース・パフォーマンスのチューニング
   - データベース・バックアップ
3. 1 次データベースと 2 次データベースまたはスタンバイ・データベースを同期し、フェイルオーバーが円滑に行われるようにします。
4. ハードウェアまたはソフトウェア内の予期しない障害を識別して、応答します。

ログ・ファイルの管理

DB2 データベース・マネージャーは番号スキームを使用してログ・ファイルに名前を付けます。この命名の方針は、ログ・ファイルの再使用とログの順序に影響を与えます。また、クライアント・アプリケーション接続を持たない DB2 データベースは、次のクライアント・アプリケーションがそのデータベース・サーバーに接続するときに新規のログ・ファイルを使用します。 DB2 Data Server データベース・ロギング動作のこれら 2 つの側面は、ログ・ファイルの管理の選択に影響を与えます。

データベース・ロギングを管理するときは、以下の項目を考慮に入れてください。

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

– データベース構成ファイルがロールフォワード・リカバリー不可 に変更された場合
– S9999999.LOG が使用された場合

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

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

- データベースが (**ACTIVATE DATABASE** コマンドによって) アクティブにされていない場合は、すべてのアプリケーションがそのデータベースから切断されると、DB2 データベース・マネージャーにより現行のログ・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 データベース・マネージャーにより新しいログ・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・ファイル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 (175 ページの図 9 を参照)。例えば、バックアップ 2 をリカバリーする場合、使用できるログ・シーケンスは 2 通りあります。全データベースのリカバリー時に、特定の時点までロールフォワード操作を実行し、ログの終わりまで達しないうちに処理を停止すると、新しいログ・シーケンスが作成されます。 2 つのログ・シーケンスを結合することはできません。オンライン・バックアップ・イメージが最初のログ・シーケンスの全体にわたっている場合は、ロールフォワード操作を完了するのにこのログ・シーケンスを使用する必要があります。

  リカバリー後に新規のログ・シーケンスを作成した場合は、古いログ・シーケンスの表スペースのバックアップ・イメージはすべて無効になります。このことは通常リストア時に認識されますが、データベースのリストア操作の直後に表スペースのリストア操作を行うと、リストア・ユーティリティーは古いログ・シーケンスの表スペースのバックアップ・イメージを認識できません。データベースが

実際にロールフォワードされるまで、使用されるログ・シーケンスは不明です。表スペースが古いログ・シーケンスにある場合は、表スペースのロールフォワード操作によってそれを「収集」しなければなりません。無効なバックアップ・イメージを使用してリストア操作を行うと、正常に完了することもありますが、その表スペースの表スペース・ロールフォワード操作は失敗し、表スペースはリストア・ペンディング状態のままになります。

例えば、表スペース・レベルのバックアップ操作であるバックアップ 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 9 を参照)。データベース・レベルのバックアップ・イメージのバックアップ 2 を使用してリストアおよびロールフォワードを実行する場合は、 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・イメージのバックアップ 3 を使って表スペースのリストアとロールフォワード・リカバリーを行うことはできません。

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

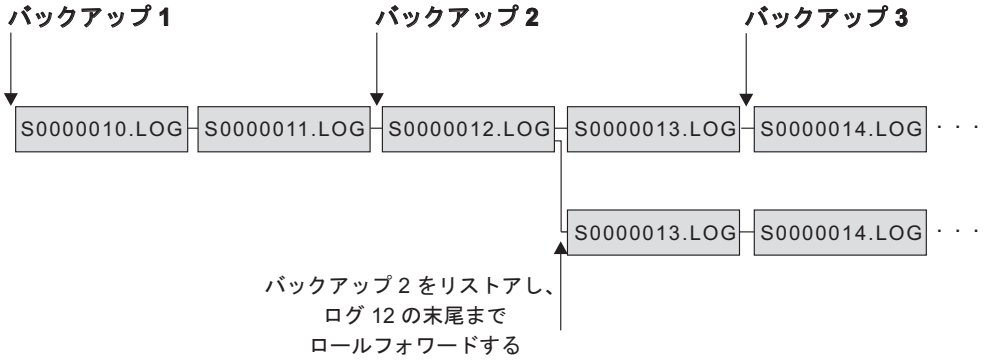


図 9. ログ・ファイル名の再使用

オンデマンドのログのアーカイブ

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

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

db2tapemgr を使用したログ・アーカイブ

db2tapemgr ユーティリティーを使用して、磁気テープ装置にアーカイブ・ログ・ファイルを保管することができます。**db2tapemgr** ユーティリティーはログ・ファイルをディスクから指定の磁気テープ装置にコピーし、コピーされたログ・ファイルの新規ロケーションが反映されるようにリカバリー履歴ファイルを更新します。

構成

データベース構成パラメーター **logarchmeth1** に、テープにコピーするログ・ファイルのディスク上のロケーションを設定します。 **db2tapemgr** ユーティリティーは、コピーするログ・ファイルを見つけるためにこの **logarchmeth1** 値を読み取ります。パーティション・データベース環境では、コピーされるログ・ファイルを含むデータベース・パーティションごとに **logarchmeth1** 構成パラメーターを設定する必要があります。

db2tapemgr ユーティリティーは **logarchmeth2** データベース構成パラメーターを使用しません。

STORE および DOUBLE STORE パラメーター

db2tapemgr コマンドを **STORE** または **DOUBLE STORE** いずれかのパラメーターを付けて発行し、アーカイブ・ログをディスクからテープに転送します。

- **STORE** パラメーターは、ある範囲またはすべてのログ・ファイルを、ログ・アーカイブ・ディレクトリーから、指定した磁気テープ装置に保管し、ディスクからそれらのファイルを削除します。
- **DOUBLE STORE** パラメーターは、ヒストリー・ファイルをスキャンして、ログが以前にテープに保管されたかどうかを確認します。
  - ログが以前に保管されていない場合、**db2tapemgr** はログ・ファイルをテープに保管しますが、それをディスクからは削除しません。
  - ログが以前に保管されている場合、**db2tapemgr** はログ・ファイルをテープに保管し、それをディスクから削除します。

  アーカイブ・ログのコピーをテープとディスクに重複して維持する場合、または同じログを 2 つの異なるテープに保管する場合に、**DOUBLE STORE** を使用します。

db2tapemgr コマンドを **STORE** または **DOUBLE STORE** いずれかのパラメーターを付けて発行する場合、**db2tapemgr** ユーティリティーは、ヒストリー・ファイルの中で、**logarchmeth1** 構成パラメーターがディスクに設定されている項目を最初にスキャンします。ディスクにあるはずなのにディスクにないファイルを検出した場合、そのユーティリティーは警告を出します。保管するログ・ファイルを **db2tapemgr** ユーティリティーが検出しなかった場合、その操作は停止し、実行するものがないことを通知するメッセージが出されます。

RETRIEVE パラメーター

db2tapemgr コマンドを **RETRIEVE** パラメーターを付けて発行すると、ファイルをテープからディスクに転送します。

- **RETRIEVE ALL LOGS** または **LOGS** *n* **TO** *n* パラメーターを使用して、指定した基準を満たすすべてのアーカイブ・ログを検索し、それらをディスクにコピーします。
- **RETRIEVE FOR ROLLFORWARD TO POINT-IN-TIME** パラメーターを使用して、ロールフォワード操作を実行するために必要なすべてのアーカイブ・ログを検索し、それらをディスクにコピーします。
- **RETRIEVE HISTORY FILE** パラメーターを使用して、ヒストリー・ファイルをテープから検索し、それをディスクにコピーします。

動作

- **db2tapemgr** ユーティリティーがディスク上のログ・ファイルを検出した場合、テープ・ヘッダーを読み取り、ログ・ファイルをテープに書き込めることを確認します。さらに、現在テープ上にあるファイルの履歴を更新します。更新に失敗した場合、操作は停止し、エラー・メッセージが表示されます。
- テープが書き込み可能である場合、**db2tapemgr** ユーティリティーはログをテープにコピーします。ファイルのコピーが完了すると、ログ・ファイルはディスクから削除されます。最後に、**db2tapemgr** ユーティリティーはヒストリー・ファイルをテープにコピーし、それをディスクから削除します。
- **db2tapemgr** ユーティリティーは、ログ・ファイルをテープに追加しません。保管操作でテープ全体を使い切らない場合には、未使用のスペースが無駄になります。
- **db2tapemgr** ユーティリティーはログ・ファイルを所定のテープに一度だけ保管します。この制約事項は、テープが伸びることなど、磁気テープ・メディアへの書き込みに固有の問題を避けるために存在しています。
- パーティション・データベース環境では、**db2tapemgr** ユーティリティーは、一度に 1 つのデータベース・パーティションに対してのみ実行されます。**db2tapemgr** コマンドの **ON DBPARTITIONNUM** パラメーターを使用してデータベース・パーティション番号を指定し、それぞれのデータベース・パーティションごとに該当するコマンドを実行する必要があります。さらに、各データベース・パーティションが磁気テープ装置にアクセスできることを確認する必要もあります。
- DB2 pureScale環境では **db2tapemgr** ユーティリティーはサポートされません。

例

以下の例は、**db2tapemgr** コマンドを使用して、すべてのログ・ファイルをデータベース・パーティション番号 0 にあるデータベース・サンプルの 1 次アーカイブ・ログ・パスから磁気テープ装置に保管し、それらをアーカイブ・ログ・パスから除去する方法を示しています。

```
db2tapemgr db sample on dbpartitionnum 0 store on /dev/rmt0.1 all logs
```

以下の例は、最初の 10 個のログ・ファイルを 1 次アーカイブ・ログ・パスから磁気テープ装置に保管し、それらをアーカイブ・ログ・パスから除去する方法を示しています。

```
db2tapemgr db sample on dbpartitionnum store on /dev/rmt0.1 10 logs
```

以下の例は、最初の 10 個のログ・ファイルを 1 次アーカイブ・ログ・パスから磁気テープ装置に保管してから同じログ・ファイルを 2 番目のテープに保管し、それらをアーカイブ・ログ・パスから除去する方法を示しています。

```
db2tapemgr db sample on dbpartitionnum double store on /dev/rmt0.1 10 logs
db2tapemgr db sample on dbpartitionnum double store on /dev/rmt1.1 10 logs
```

以下の例は、すべてのログ・ファイルをテープから取得してディレクトリーに入れる方法を示しています。

```
db2tapemgr db sample on dbpartitionnum retrieve all logs from /dev/rmt1.1
      to /home/dbuser/archived_logs
```

ユーザー出口プログラムの使用によるログ・ファイルのアーカイブと検索の自動化

ログ・ファイルのアーカイブと検索は、ユーザー出口プログラムを作成することによって自動化できます。このユーザー出口プログラムは、アーカイブまたは検索操作を実行するために DB2 データベース・マネージャーが呼び出すものです。

DB2 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すと、次の一連の処理が発生します。

- データベース・マネージャーはユーザー出口プログラムに制御を渡します。
- データベース・マネージャーはユーザー出口プログラムにパラメーターを渡します。
- それらの処理が完了したら、ユーザー出口プログラムはデータベース・マネージャーに戻りコードを戻します。

構成

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リトリーブを行う前に、 **logarchmeth1** データベース構成パラメーターが USEREXIT に設定されていることを確認してください。これにより、データベースをロールフォワード・リカバリーすることもできるようになります。

ユーザー出口プログラムの要件

- ユーザー出口プログラムの実行可能ファイル名は db2uext2 にする必要があります。
- 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、移動ではなくコピーする必要があります。アクティブ・ログ・パスからログ・ファイルを削除または移動しないでください。アクティブ・ログ・パスからログ・ファイルを除去すると、障害が発生した場合に DB2 データベースでのリカバリーが正常に行えなくなる可能性があります。

  DB2 データベースでリカバリーを行う場合は、ログ・ファイルがアクティブ・ログ・パスになければなりません。 DB2 データベース・サーバーは、アーカイブ・ログ・ファイルを、リカバリーのために必要がなくなったときにアクティブ・ログ・パスから除去します。
- ユーザー出口プログラムはエラー条件を処理する必要があります。これが必要なのは、DB2 データベース・マネージャーが処理できる戻り条件セットに限りがあるためです。

181 ページの『ユーザー出口のエラー処理』を参照してください。

- DB2 データベース・マネージャー・インスタンスが呼び出せるユーザー出口プログラムはそれぞれ 1 つだけです。そのため、ユーザー出口プログラムで実行する可能性のある操作ごとにセクションを組み込んで、ユーザー出口プログラムを設計する必要があります。

ユーザー出口プログラムの例

サンプル・ユーザー出口プログラムは、サポートされているすべてのプラットフォーム用に提供されています。特定の要件に合わせて、これらのプログラムを変更することができます。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がコメントされています。

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・ファイルを削除または移動しないでください。(これはデータベース・リカバリー時に問題を引き起こすことがあります。) DB2 は、アーカイブ・ログ・ファイルを、リカバリーのために必要がなくなったときにアクティブ・ログ・パスから除去します。

以下に、 DB2 Data Server 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 オペレーティング・システム**

  UNIX オペレーティング・システム用の DB2 Data Server 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 オペレーティング・システムには 4 つのサンプル・ユーザー出口プログラムがあります。

  – `db2uext2.ctsm`

    このサンプルは Tivoli Storage Manager を使用して、データベース・ログ・ファイルのアーカイブとリトリーブを行います。

  – `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リトリーブを行います。

  – `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リトリーブを行います。

  – `db2uxt2.cxbsa`

このサンプルは、X/Open グループにより発行された XBSA Draft 0.8 で作動します。データベース・ログ・ファイルのアーカイブとリトリーブに使用できます。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 Data Server のユーザー出口サンプル・プログラムは、sqllib¥samples¥c サブディレクトリーにあります。提供されているサンプルは C 言語でコーディングされていますが、ユーザー出口プログラムは別のプログラミング言語で作成することもできます。

  ユーザー出口プログラムは、実行可能ファイルで、その名前は db2uext2 でなければなりません。

  Windows オペレーティング・システムには 2 つのサンプル・ユーザー出口プログラムがあります。

  – db2uext2.ctsm

    このサンプルは Tivoli Storage Manager を使用して、データベース・ログ・ファイルのアーカイブとリトリーブを行います。

  – 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リトリーブを行います。

ユーザー出口プログラムの呼び出し形式

DB2 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。

コマンド構文

```
db2uext2 -OS<os> -RL<db2rel> -RQ<request> -DB<dbname>
-NN<nodenum> -LP<logpath> -LN<logname> -AP<tsmpasswd>
-SP<startpage> -LS<logsize>
```

os
インスタンスが実行されているプラットフォームを指定します。有効な値は次のとおりです。 AIX、Solaris、HP-UX、SCO、Linux、および NT。

db2rel DB2 リリース・レベルを指定します。例えば、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。例えば、5 など。

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。例えば、次のとおりです。
/u/database/log/path/ または d:¥logpath¥

logname
アーカイブまたはリトリーブされるログ・ファイルの名前を指定します。例えば 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デバイスの、4 KB のオフセット・ページの数を指定します。

logsize ログ・エクステントのサイズを 4 KB のページ数で指定します。パラメーターは、ロー・デバイスを使用してロギングを取る場合に限り有効です。

ユーザー出口のエラー処理

ユーザー出口プログラムを作成してログ・ファイルのアーカイブと検索を自動化する場合、ユーザー出口プログラムは、それを呼び出した DB2 データベース・マネージャーに戻りコードを渡します。 DB2 データベース・マネージャーは限られたリストにある特定のエラー・コードしか処理できません。しかし、ユーザー出口プログラムは、オペレーティング・システム・エラーなど、多種多様なエラー条件に遭遇する可能性があります。ユーザー出口プログラムは、遭遇するエラー条件を、データベース・マネージャーが処理できるエラー・コードにマップする必要があります。

表 8 では、ユーザー出口プログラムによって戻されることのある戻りコードを示し、これらのコードがどのようにデータベース・マネージャーによって解釈されるかを説明します。戻りコードが表にリストされていない場合は、値 32 の場合と同じように扱われます。

表 8. ユーザー出口プログラム戻りコード

戻りコード	説明
0	正常実行。
4	一時リソース・エラーが検出された。[a]
8	オペレーター要介入。[a]
12	ハードウェア・エラー。[b]
16	ユーザー出口プログラムまたはそのプログラムで使用されているソフトウェア機能にエラー。[b]
20	ユーザー出口プログラムに渡された 1 つまたは複数のパラメーターにエラー。指定されたパラメーターをユーザー出口プログラムが正しく処理しているか調べてください。[b]
24	ユーザー出口プログラムが検出されなかった。 [b]
28	入出力 (I/O) の失敗またはオペレーティング・システムが原因で生じたエラー。[b]
32	ユーザー出口プログラムがユーザーにより終了させられた。[b]
255	ユーザー出口プログラムが、実行可能ファイルのライブラリー・ファイルをロードできなかったことが原因で生じたエラー。[c]

表 8. ユーザー出口プログラム戻りコード (続き)

戻りコード	説明
[a] アーカイブおよびリトリーブ要求の場合、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・ファイルへのリトリーブ要求に 4 または 8 が継続して戻されると、DB2 は成功するまで再試行を継続します。(このことはロールフォワード操作、または複製ユーティリティーが使用する **db2ReadLog** API への呼び出しについても当てはまります。) [b] ユーザー出口要求は、5 分間中断します。この間は、エラー状態を引き起こした要求を含めて、すべての要求が無視されます。この 5 分間の中断の後、次の要求が処理されます。この要求がエラーなしに処理された場合、新規ユーザー出口要求は継続します。 DB2 は、以前に失敗または中断されたアーカイブ要求を再発行します。再試行時に 8 より大きい戻りコードが生成される場合、要求はさらに 5 分間中断されます。このような 5 分間の中断は、問題が訂正されるまで、あるいはデータベースを停止して再始動するまで繰り返されます。すべてのアプリケーションがデータベースから切断されると、DB2 は、以前に正常にアーカイブされなかった可能性のあるすべてのログ・ファイルに対するアーカイブ要求を発行します。ユーザー出口プログラムがログ・ファイルのアーカイブに失敗した場合、使用しているディスクがログ・ファイルでいっぱいになることがあり、パフォーマンスが低下することがあります。ディスクがいっぱいになると、データベース・マネージャーはデータベース更新に関するアプリケーション要求を受け入れなくなります。ログ・ファイルをリトリーブするためにユーザー出口プログラムが呼び出されると、ロールフォワード・リカバリーは中断されますが、 ROLLFORWARD STOP オプションが指定されていない限り停止することはありません。停止 (STOP) オプションが指定されていなかった場合は、問題を訂正してリカバリーを再開することができます。 [c] ユーザー出口プログラムによってエラー・コード 255 が戻された場合、プログラムが実行可能ファイルのライブラリー・ファイルをロードできなかった可能性があります。これを検証するには、ユーザー出口プログラムを手動で起動してください。さらに情報が表示されます。 **注:** アーカイブおよびリトリーブ操作中、0、および 4 以外のすべての戻りコードについて、アラート・メッセージが発行されます。アラート・メッセージには、ユーザー出口プログラムからの戻りコード、およびユーザー出口プログラムに備えられた入力パラメーターのコピーが含まれています。	

ログ・ファイルの割り振りと除去

クラッシュ・リカバリーに必要なログ・ファイルは、アクティブ・ログと呼ばれます。無限ロギングを有効にしていない限り、クラッシュ・リカバリーに必要となる可能性のあるログ・ファイルが、アクティブ・ログ・パスから削除されることはありません。

無限ロギングが有効になっている場合は、さらに多くのアクティブ・ログ・ファイルのためにスペースを空ける必要が生じると、データベース・マネージャーは、アクティブ・ログ・ファイルをアーカイブし、その名前を変更して新規ログ・ファイルを作成します。無限ロギングを使用していてクラッシュ・リカバリーが必要とされる場合には、クラッシュ・リカバリーを完了させるために、アーカイブ・ログ・パスからログ・ファイルを取り出すことが必要となる可能性があります。

logarchmeth1 データベース構成パラメーターが OFF に設定されていない場合、満杯のログ・ファイルは、クラッシュ・リカバリーにもはや必要ではなくなった時に

初めて、削除対象の候補になります。ただし、無限ロギングが有効になっている場合は別です (この場合、ログ・ファイルは代わりにアーカイブ・ログ・パスに移動される可能性があります)。

logarchmeth1 または **logarchmeth2** が OFF、LOGRETAIN、USEREXIT 以外の値に設定されている場合は、アーカイブ・ログ・ファイルを圧縮できるようにすると、アーカイブ・ログ・ファイルに必要なディスク・スペース量を減らすのに便利です。

新規ログ・ファイルを割り振り、古いログ・ファイルを除去するプロセスは、**logarchmeth1** および **logarchmeth2** データベース構成パラメーターの設定に従属しています。

logarchmeth1 および logarchmeth2 が、OFF に設定されている場合
循環ロギングが使用されます。循環ロギングでは、クラッシュ・リカバリーはサポートされますが、ロールフォワード・リカバリーはサポートされません。

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

すべてのログ・ファイルがアクティブで、循環ロギング・プロセスが最初のログ・ファイルに折り返せなかった場合に、ログ・フルの状態が発生する可能性があります。 2 次ログ・ファイルは、すべての 1 次ログ・ファイルがアクティブで満杯の時に作成されます。 2 次ログ・ファイルは、データベースが非アクティブ化される場合や、使用しているスペースがアクティブ・ログ・ファイル用に必要な場合に削除されます。

logarchmeth1 または logarchmeth2 が LOGRETAIN に設定される
アーカイブ・ロギングは、使用されます。データベースは、リカバリー可能です。ロールフォワード・リカバリーおよびクラッシュ・リカバリーの両方は、使用可能になります。データベース・マネージャーは、ログ・ファイルを管理しません。ログ・ファイルをアーカイブしたら、アクティブ・ログ・パスからログ・ファイルを削除して、そのディスク・スペースを新しいログ・ファイル向けに再使用できるようにする必要があります。どのログ・ファイルがアーカイブされたログなのかを判別するには、**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** 値よりも小さいシーケンス番号のログは、アクティブではないのでアーカイブして削除することが可能です。

logarchmeth1 または logarchmeth2 が OFF または LOGRETAIN 以外の値に設定される
アーカイブ・ロギングは、使用されます。データベースは、リカバリー可能です。ロールフォワード・リカバリーおよびクラッシュ・リカバリーの両方は、使用可能になります。ログ・ファイルは満杯になると、データベース・マネージャーによって自動的にアーカイブされます。

ログ・ファイルは削除されません。むしろ、新しいログ・ファイルが必要になり、利用できるログ・ファイルがない場合には、アーカイブ・ログ・ファイルがリネームされ、再使用されます。アーカイブ・ログ・ファイルを閉じて、アーカイブ・ログ・ファイル・ディレクトリーにコピーしても、そのフ

ファイルは削除もリネームもされません。データベース・マネージャーは新規ログ・ファイルが必要になるまで待った後、最も古いアーカイブ・ログをリネームします。リカバリー中にデータベース・ディレクトリーに移動したログ・ファイルは、それがもはや必要ではなくなった時、リカバリー処理中に除去されます。

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

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

データベースが再始動すると、データベース・ログ・ディレクトリーにあるログの最小の数は、 **logprimary** データベース構成パラメーターを使用して構成できる 1 次ログの数と等しくなります。1 次ログの数以上のものがログ・ディレクトリーに検出されることもありえます。この状態は、データベースがシャットダウンした時点でのログ・ディレクトリー内の空のログの数が、データベースが再始動した時点での **logprimary** 構成パラメーターの値より大きい場合に起こります。これは、**logprimary** 構成パラメーターが、シャットダウンするデータベースと再始動するデータベースとの間で変更された場合、または 2 次ログが割り振られ、決して使用されない場合に発生します。

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

ログ・ファイルをバックアップ・イメージに含める

オンライン・バックアップ操作を実行する際には、データベースのリストアおよびリカバリーに必要なログ・ファイルをバックアップ・イメージに含めることを指定できます。つまり、災害時リカバリー・サイトにバックアップ・イメージを送る必要がある場合、ログ・ファイルを個別に送信したり、自分でそれらをパッケージする必要はないということです。さらに、オンライン・バックアップの整合性を保つ

ために必要なログ・ファイルを決める必要もなくなります。これにより、正常なリカバリーに必要なログ・ファイルを削除してしまうことをある程度防げます。

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

バックアップ・イメージからログ・ファイルをリストアするには、**RESTORE DATABASE** コマンドの **LOGTARGET** オプションを使用し、 DB2 サーバーに存在する完全修飾パスを指定します。そうすると、データベース・リストア・ユーティリティーは、イメージ内のログ・ファイルをターゲット・パスに書き込みます。同じ名前のログ・ファイルがターゲット・パスに存在する場合、リストア操作は失敗し、エラーが戻されます。**LOGTARGET** オプションが指定されない場合、ログ・ファイルはバックアップ・イメージからリストアされません。

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

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

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

ログ・ファイルを含むオンライン・バックアップ・イメージからリストアされたデータベースをロールフォワードする場合、エラー SQL1268N が表示されることがあります。このエラーは、ログの検索中に受け取ったエラーのためにロールフォワード・リカバリーが停止したことを示します。このエラーは、バックアップ・イメージをリストアしようとしているターゲット・システムに、ソース・システムがトランザクション・ログのアーカイブに使用する機能に対するアクセス権がない場合に生成されます。

データベースのバックアップ時に **BACKUP DATABASE** コマンドの **INCLUDE LOGS** オプションを指定し、その後にこのバックアップ・イメージを使用するリストア操作とロールフォワード操作を実行すると、バックアップ・イメージにログが含まれていても、依然として DB2 はデータベースのロールフォワード時に追加のトランザクション・ログを検索します。標準的なロールフォワードの動作として、ログが見つからなくなるまで追加のトランザクション・ログを検索し続けます。同じタイム・

スタンプのログ・ファイルが複数存在することもあります。したがって DB2 は、データベースをロールフォワードしている特定時点と一致する 1 つ目のタイム・スタンプを検出しても、そのタイム・スタンプのあるログ・ファイルが他にも存在する可能性があるので、即時に停止しません。代わりに DB2 は、指定された特定時点より大きいタイム・スタンプを検出するまで、トランザクション・ログを検索し続けます。

追加ログを検出できなくなると、ロールフォワード操作は正常に終了します。しかし、追加のトランザクション・ログ・ファイルの検索中にエラーが生じると、エラー SQL1268N が戻されます。エラー SQL1268N が戻される場合、その理由は初期リストア中に特定のデータベース構成パラメーターがリセットされたか上書きされたためです。該当するデータベース構成パラメーターのうち 3 つは TSM パラメーターの **tsm_nodename**、**tsm_owner**、および **tsm_password** です。これらのパラメーターはすべて NULL にリセットされます。ログの終わりまでロールフォワードするには、これらのデータベース構成パラメーターをリセットし、ロールフォワード操作以前のソース・システムに対応させる必要があります。代わりに、**ROLLFORWARD DATABASE** コマンドの発行時に **NORETRIEVE** オプションを指定することもできます。こうすると、DB2 データベース・システムは他の場所で存在しない可能性のあるトランザクション・ログの取得を試行しません。

注:

1. このフィーチャーは、オフライン・バックアップではサポートされていません。
2. ログがオンライン・バックアップ・イメージに含まれる場合、バージョン 8.2 より前の DB2 データベース・リリースでは、結果のイメージをリストアすることはできません。

不慮のログ・ファイル消失の回避

データベースをドロップしたり、ポイント・イン・タイム指定ロールフォワード・リカバリーを実行する必要がある状況で、将来のリカバリー操作で必要になるかもしれないログ・ファイルを消失してしまう可能性があります。そのようなケースでは、現行データベース・ログ・パス・ディレクトリーにあるすべてのログのコピーを作成することが大切です。

以下のシナリオを考慮に入れてください。

- リストア操作の前にデータベースをドロップする計画をしている場合、 **DROP DATABASE** コマンドを実行する前に、アクティブ・ログ・パスにあるログ・ファイルを保管する必要があります。それらのログ・ファイルのうちいくつかは、データベースをドロップする前にアーカイブされていなかった可能性があるため、そのデータベースがリストアされた後、それらのログ・ファイルがロールフォワード・リカバリーに必要になるかもしれません。通常、**RESTORE** コマンドを実行する前にデータベースをドロップする必要はありません。ただし、データベースは **RESTORE** コマンドが失敗する程度にまで損傷しているため、そのデータベースをドロップ (または **DROP DATABASE** コマンドの **AT DBPARTITIONNUM** パラメーターを指定して、1 つのデータベース・パーティションのデータベースをドロップ) しなければならない場合があります。さらに、白紙状態から開始するために、リストア操作の前にデータベースをドロップするように決定するかもしれません。
- 特定の時点までのデータベースのロールフォワード操作を実行する場合は、指定するタイム・スタンプの後のログ・データは上書きされます。ポイント・イン・

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

保守が可用性に与える影響の最小化

DB2 データベース・ソリューションでは、ソフトウェアまたはハードウェアのアップグレード、データベース・パフォーマンスのチューニング、データベース・バックアップ、統計収集、および業務目的のモニターなどの保守を実行する必要があります。こうした保守の実行がご使用のソリューションの可用性に与える影響を最小限に抑えるには、オフライン保守を注意深くスケジューリングすることや、DB2 フィーチャーと機能を使用してオンライン保守が可用性に与える影響を少なくすることが関係しています。

始める前に

ご使用の DB2 データベース・ソリューションの可用性に保守が与える影響を最小にとどめるには、下記のステップを実行する前に、次の事柄を行う必要があります。

- 自動保守を構成する。
- 高可用性災害時リカバリー (HADR) フィーチャーをインストールする。

手順

1. 必要な保守を実行するために自動保守を許可します。

   DB2 データベースでは、多くのデータベース保守アクティビティーを自動化できます。いったん自動保守を構成すると、その保守を実行するために追加ステップを行わずに済みます。

2. DB2 高可用性災害時リカバリー (HADR) のローリング・アップグレードを使用して、他の保守アクティビティーが与える影響を最小に抑えます。

   ソフトウェアまたはハードウェアをアップグレードする場合、またはデータベース・マネージャー構成パラメーターの一部を変更する場合に HADR フィーチャーを使用すると、可用性の中断を最小限に抑えつつこうした変更を行えます。HADR によって可能になるこうしたシームレスな変更は、ローリング・アップグレードと呼ばれます。

   保守アクティビティーによっては、HADR 環境でも、保守を実行する前にデータベースをシャットダウンしなければならない場合があります。一部の条件下では、HADR データベースと通常のデータベースのシャットダウンの手順が若干

異なります。接続先のクライアント・アプリケーションによって HADR データベースが開始される場合には、**DEACTIVATE DATABASE** コマンドを使用する必要があります。

DB2 高可用性災害時リカバリー (HADR) の停止

DB2 高可用性災害時リカバリー (HADR) フィーチャーを使用している場合、1 次またはスタンバイ・データベースで保守を実行するために HADR 操作を停止する必要になる場合があります。 HADR 操作の停止は、保守を行うデータベースでのみ行ってください。 HADR の使用を完全に停止するには、両方のデータベースで HADR を停止してください。

このタスクについて

警告: 指定されたデータベースを停止するものの、その役割を HADR 1 次データベースまたはスタンバイ・データベースのままにしておきたい場合は、STOP HADR コマンドを発行しないでください。**STOP HADR** コマンドを発行するとデータベースは標準データベースになり、HADR データベースとして運用を再開するためには再初期設定が必要になることがあります。代わりに、**DEACTIVATE DATABASE** コマンドを発行してください。

標準のデータベースに **STOP HADR** コマンドを発行すると、エラーが戻されます。

手順

1 次またはスタンバイ・データベースで HADR 操作を停止するには、次のようにします。

- HADR 操作を停止するデータベースの CLP から **STOP HADR** コマンドを発行します。

  次の例では、HADR 操作はデータベース SOCKS で停止します。

  ```
  STOP HADR ON DATABASE SOCKS
  ```

  このコマンドを非アクティブの 1 次データベースに対して発行する場合、データベースは標準データベースに切り替わり、オフラインのままになります。

  このコマンドを非アクティブのスタンバイ・データベースに対して発行する場合、データベースは標準データベースに切り替わり、ロールフォワード・ペンディング状態となって、オフラインのままになります。

  このコマンドをアクティブな 1 次データベースで発行する場合、スタンバイ・データベースへのログの送信が停止し、 1 次データベースで HADR エンジン・ディスパッチ可能単位 (EDU) がすべてシャットダウンされます。データベースは、標準データベースに切り替わり、オンラインのままになります。トランザクション処理は続行可能です。データベースの役割を 1 次データベースに戻す場合は、START HADR AS PRIMARY コマンドを発行できます。

  このコマンドをアクティブなスタンバイ・データベースに対して発行する場合、エラー・メッセージが戻され、スタンバイ・データベースを標準データベースに変更する前にそれを非アクティブにする必要があることが示されます。

- アプリケーションから、**db2HADRStop** アプリケーション・プログラミング・インターフェース (API) を呼び出します。
- IBM Data Studio から、**STOP HADR** コマンドのタスク・アシストを開きます。

関連情報:

高可用性災害時リカバリー (HADR) 環境におけるデータベースの活動化と非活動化

標準データベースがクライアント接続によって開始される場合、データベースは最後のクライアントの切断時にシャットダウンされます。HADR 1 次データベースがクライアント接続によって開始される場合は、**ACTIVATE DATABASE** コマンドを使用してデータベースを開始するのと同じです。クライアント接続によって開始された HADR 1 次データベースをシャットダウンするには、明示的に **DEACTIVATE DATABASE** コマンドを発行することが必要です。

ロールフォワード・ペンディング状態の標準データベースに対しては、**ACTIVATE DATABASE** および **DEACTIVATE DATABASE** コマンドが適用されません。ロールフォワードを継続するか、ロールフォワードを停止するか、または **START HADR** を使用してデータベースを HADR スタンバイ・データベースとして開始することしかできません。いったんデータベースを HADR スタンバイ・データベースとして開始したなら、**ACTIVATE DATABASE** および **DEACTIVATE DATABASE** コマンドを使用してデータベースを開始したり停止したりすることができます。

1 次データベースをアクティブにする方法は以下のとおりです。

- クライアント接続
- **ACTIVATE DATABASE** コマンド
- IBM Data Studio の **ACTIVATE DATABASE** コマンドに関するタスク・アシスト
- **START HADR** コマンドに AS PRIMARY オプションを指定する

1 次データベースを非アクティブにする方法は以下のとおりです。

- **DEACTIVATE DATABASE** コマンド

  注: **DEACTIVATE DATABASE** コマンドまたは sqle_deactivate_db API を使用して、切断済みピア状態の HADR 1 次データベースを非アクティブにすると、データベースが不整合状態になります。データベースは再始動時にクラッシュ・リカバリーが必要になり、このデータベースが再始動するまでオフライン・バックアップを取ることはできません。
- IBM Data Studio の **DEACTIVATE DATABASE** コマンドに関するタスク・アシスト
- **db2stop** コマンドに **FORCE** パラメーターを指定する

スタンバイ・データベースをアクティブにする方法は以下のとおりです。

- **ACTIVATE DATABASE** コマンド
- IBM Data Studio の **ACTIVATE DATABASE** コマンドに関するタスク・アシスト
- **START HADR** コマンドに AS STANDBY オプションを指定する

スタンバイ・データベースを非アクティブにする方法は以下のとおりです。

- **DEACTIVATE DATABASE** コマンド

- IBM Data Studio の **DEACTIVATE DATABASE** コマンドに関するタスク・アシスト
- **db2stop** コマンドに **FORCE** パラメーターを指定する

HADR ペアのシャットダウンにおける推奨順序

n

警告: STOP HADR コマンドは 1 次またはスタンバイ、あるいはその両方で HADR を停止するために使用できますが、これは注意して使用する必要があります。指定されたデータベースを停止するが、その HADR 1 次データベースまたはスタンバイ・データベースの役割を保ちたい場合は、**STOP HADR** コマンドを発行しないでください。**STOP HADR** コマンドを発行すると、そのデータベースは標準データベースになり、HADR データベースとして運用を再開するためには再初期設定が必要になることがあります。代わりに、**DEACTIVATE DATABASE** コマンドを発行してください。

HADR 操作のシャットダウンのみを行う場合は、次のようにして HADR ペアをシャットダウンすることが推奨されています。

1. 1 次データベースを非活動化します
2. 1 次データベースの DB2 を停止します
3. スタンバイ・データベースを非活動化します
4. スタンバイ・データベースの DB2 を停止します

DB2 高可用性災害時リカバリー (HADR) 環境での表スペース・リバランス操作の考慮事項

ALTER TABLESPACE REBALANCE ステートメントまたは ALTER TABLESPACE USING STOGROUP ステートメントを使用して、1 次データベースに対するリバランス操作を開始できます。このステートメントは、スタンバイ・データベースに対して再生され、対応するリバランス操作が開始されます。

リバランス操作中、ALTER TABLESPACE ステートメントに REBALANCE SUSPEND 節を付けて指定すると、1 次データベースに対するリバランス操作を中断することができます。中断されたリバランス操作を再開するには、ALTER TABLESPACE ステートメントに REBALANCE RESUME 節を付けて指定します。

スタンバイ・データベースは、ALTER TABLESPACE REBALANCE SUSPEND ステートメントを再生するときも、アクティブ状態を維持します。1 次データベースに対するリバランスが中断されているため、スタンバイが新規の 1 次データベースとして引き継ぐと、この新しい 1 次データベースに対するリバランス操作は中断され、新しいスタンバイ・データベースに対してリバランス操作が暗黙的に再開されます。

スプリット・ミラーを使用して、データベースをクローン・データベースまたはスタンバイ・データベースとしてリストアすると、中断されていた表スペースのリバランス操作が、データベースの開始と同時に再開されます。

DB2 高可用性災害時リカバリー (HADR) 環境でのローリング更新とローリング・アップグレードの実行

ソフトウェアまたはハードウェアをアップグレードするとき、DB2 データベース・システムを更新するとき、あるいはデータベース構成パラメーターを変更するときには、高可用性災害時リカバリー (HADR) 環境でこの手順を使用します。この手順を使用すると、アップグレード・プロセスの間、データベース・サービスはずっと使用可能な状態になります。ただし、処理があるデータベースから別のデータベースへ切り替えられるときには、そのときだけ一時的にサービスが中断します。

複数スタンバイを使用すれば、更新プロセスまたはアップグレード・プロセスの間、高可用性災害時リカバリー保護を続行できます。

始める前に

HADR のシステム要件を確認します。 69 ページの『高可用性災害時リカバリー (HADR) のシステム要件』を参照してください。

ローリング・アップグレードを開始する前に、HADR ペアをピア状態にしておく必要があります。

注: DB2 データベース・システム・フィックスパックおよびアップグレードはすべて、実動システムに適用する前に、テスト環境にインプリメントする必要があります。

このタスクについて

この手順は、DB2 データベース・システムの古いバージョンから新しいバージョンへのアップグレードでは機能しません。例えば、この手順は、バージョン 8 からバージョン 9 のデータベース・システムのアップグレードには使用できません。この手順は、例えばフィックスパックを適用するなどの、ある修正レベルから別の修正レベルにデータベース・システムにローリング更新を行う場合にのみ使用できます。ローリング更新中には、スタンバイ・データベースの修正レベル (例えば、フィックスパック・レベル) は、新規レベルをテストするために、短期間だけ、1 次データベースよりも新しくても構いません。ただし、この構成は長期間維持しないようにしてください。異なるレベル間で互換性がなくなる可能性のあるフィーチャーを使用するリスクを減らすためです。 1 次データベースのデータベース・システムの修正レベルがスタンバイ・データベースよりも新しい場合、1 次データベースおよびスタンバイ・データベースは、互いに接続しません。

この手順は、DB2 HADR 構成パラメーターを更新する場合には機能しません。HADR 構成パラメーターへの更新は別に実行する必要があります。HADR では、1 次とスタンバイのパラメーターが同じでなければならないので、1 次データベースとスタンバイ・データベースの両方を同時に非アクティブにし、更新する必要があるかもしれません。

手順

HADR 環境でローリング・アップグレードを実行するには、次のようにします。

1. スタンバイ・データベースが存在するシステムをアップグレードします。

a. **DEACTIVATE DATABASE** コマンドを使用して、スタンバイ・データベースをシャットダウンします。
b. 必要な場合、スタンバイ・データベースのインスタンスをシャットダウンします。
c. ソフトウェア、ハードウェア、または DB2 構成パラメーターのうち、1 つ以上を変更します。

   注: ローリング・アップグレードの実行時に HADR 構成パラメーターを変更することはできません。
d. 必要な場合、スタンバイ・データベースのインスタンスを再始動します。
e. **db2pd** コマンドを使用して、スタンバイ・データベースを再始動します。
f. スタンバイ・データベースがピア状態になったことを確認します。このことを確認するには、**GET SNAPSHOT** コマンドを使用します。

2. 1 次データベースの役割とスタンバイ・データベースの役割を切り替えます。
   a. スタンバイ・データベースで **TAKEOVER HADR** コマンドを発行します。
   b. クライアントを新しい 1 次データベースに誘導します。これは自動クライアント・リルートを使用して実行できます。

      注: スタンバイ・データベースが 1 次データベースとしてテークオーバーするため、ここで新しい 1 次データベースがアップグレードされます。 DB2 データベース・システム・フィックスパックを適用している場合、**TAKEOVER HADR** コマンドを発行すると、元の 1 次データベースの役割がスタンバイ・データベースに変更されます。しかし、このコマンドでは、新しいスタンバイ・データベースは新しく更新した 1 次データベースに接続しません。新しいスタンバイ・データベースは以前のバージョンの DB2 データベース・システムを使用するため、更新された 1 次データベースによって生成される新しいログ・レコードが理解されない場合があり、その場合にはシャットダウンしてしまいます。新しいスタンバイ・データベースを新しい 1 次データベースと再接続するために (つまり、HADR ペアをリフォームする)、新しいスタンバイ・データベースも更新する必要があります。
3. ステップ 1 と同じ手順を使用して、元の 1 次データベース (つまり、現在のスタンバイ・データベース) をアップグレードします。これが完了したら、両方のデータベースがアップグレードされ、 HADR ピア状態で相互に接続されます。HADR システムには、完全なデータベース・サービスと、完全な高可用性保護機能が備えられています。
4. オプション: 元の構成に戻すには、ステップ 2 のように、1 次データベースの役割とスタンバイ・データベースの役割を切り替えます。

   ローリング・アップグレード時に HADR スタンバイ・データベースの読み取りフィーチャーを使用可能にするには、オプションのステップ 4 を据え置き、次の手順を実行します。最初の接続時に内部 DB2 パッケージのバインドが行われますが、これは 1 次データベースでしか正常に完了できません。読み取り操作が実行される前にスタンバイ・データベース上の内部 DB2 パッケージの整合性を確実なものとするため、以下のステップが必要になります。
5. 以下のようにして、スタンバイ・データベース上で HADR スタンバイ・データベースの読み取りフィーチャーを使用可能にします。

a. スタンバイ・データベースで **DB2_HADR_ROS** レジストリー変数を ON に設定します。
b. **DEACTIVATE DATABASE** コマンドを使用して、スタンバイ・データベースをシャットダウンします。
c. スタンバイ・データベースのインスタンスを再始動します。
d. **ACTIVATE DATABASE** コマンドを使用して、スタンバイ・データベースを再始動します。
e. スタンバイ・データベースが PEER 状態になったことを確認するには、**GET SNAPSHOT** コマンドを使用します。

6. 以下のようにして、1 次データベースの役割とスタンバイ・データベースの役割を切り替えます。
   a. スタンバイ・データベースで **TAKEOVER HADR** コマンドを発行します。
   b. クライアントを新しい 1 次データベースに誘導します。
7. ステップ 5 の同じ手順を繰り返して、新しいスタンバイ・データベース上で HADR スタンバイ・データベースの読み取りフィーチャーを使用可能にします。
8. オプション: 元の構成に戻すには、ステップ 2 のように、1 次データベースの役割とスタンバイ・データベースの役割を切り替えます。

高可用性災害時リカバリー (HADR) を自動化した環境でのローリング・アップグレード

統合高可用性 (HA) フィーチャーを使用して HADR を自動化している場合、ソフトウェア (オペレーティング・システムまたは DB2 データベース・システム) やハードウェアのアップグレード、またはデータベース構成パラメーターの変更を行うには、追加の手順が必要です。自動 HADR 環境でローリング・アップグレードを実行するには、以下の手順を使用します。

始める前に

「手順」セクションの説明にある手順を実行するには、以下の前提を理解しておく必要があります。

- 2 つの DB2 インスタンス (この例では、各ノードに stevera という名前のインスタンス) があります。
- 2 つのノード (grom04 および grom03) があります。最初に HADR 1 次データベースをホスティングするのは、grom04 マシンです。
- これらのインスタンスは、当初 DB2 V9.8 GA コードで実行されていました。
- これらのインスタンスは、HADR フェイルオーバーを統合 HA によって制御するように構成されています。クラスター・ドメインの名前は test です。

注: DB2 データベース・システム・フィックスパックおよびアップグレードはすべて、実動システムに適用する前に、テスト環境にインプリメントする必要があります。

ローリング・アップグレードを開始する前に、HADR ペアをピア状態にしておく必要があります。

制約事項

この手順は、DB2 データベース・システムの古いバージョンから新しいバージョンへのマイグレーションでは機能しません。例えば、この手順は、バージョン 8 からバージョン 9 のデータベース・システムのマイグレーションには使用できません。この手順は、例えばフィックスパックを適用するなどの、ある修正レベルから別の修正レベルにデータベース・システムをアップデートする場合にのみ使用できます。

この手順は、DB2 HADR 構成パラメーターを更新する場合には機能しません。HADR 構成パラメーターへの更新は別に実行する必要があります。HADR では、1 次とスタンバイのパラメーターが同じでなければならないので、1 次データベースとスタンバイ・データベースの両方を同時に非アクティブにし、更新する必要があるかもしれません。

手順

1. 最初のシステム状態を表示します。

```
root@grom03:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  ------------------  -------------   -------  -------
test   Online    2.4.7.1             No              12347    12348

root@grom03:# lssam
Online IBM.ResourceGroup:db2_stevera_grom03_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom03_0-rs
                '- Online IBM.Application:db2_stevera_grom03_0-rs:grom03
Online IBM.ResourceGroup:db2_stevera_grom04_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom04_0-rs
                '- Online IBM.Application:db2_stevera_grom04_0-rs:grom04
Online IBM.ResourceGroup:db2_stevera_stevera_SVTDB-rg Nominal=Online
        |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs
                |- Offline IBM.Application:db2_stevera_stevera_SVTDB-rs:grom03
                '- Online IBM.Application:db2_stevera_stevera_SVTDB-rs:grom04
        '- Online IBM.ServiceIP:db2ip_9_26_124_22-rs
                |- Offline IBM.ServiceIP:db2ip_9_26_124_22-rs:grom03
                '- Online IBM.ServiceIP:db2ip_9_26_124_22-rs:grom04

root@grom03:# lsrpnode
Name     OpState   RSCTVersion
-----    -------   ------------
grom03   Online    2.4.7.1
grom04   Online    2.4.7.1
```

   この例から、grom03 上のスタンバイ・インスタンスをアップグレードする必要があることがわかります。アップグレードするには、grom03 でホスティングされているすべてのリソース・グループを停止します。

2. スタンバイ・ノード上のすべてのリソース・グループを停止し、変更を確認します。

```
root@grom03:# chrg -o Offline db2_stevera_grom03_0-rg

root@grom03:# lssam g db2_stevera_grom03_0-rg
Offline IBM.ResourceGroup:db2_stevera_grom03_0-rg Nominal=Offline
        '- Offline IBM.Application:db2_stevera_grom03_0-rs
                '- Offline IBM.Application:db2_stevera_grom03_0-rs:grom03
```

3. クラスター・ノード (スタンバイ・ノード) を停止し、変更を確認します。

```
root@grom03:# stoprpnode grom03

root@grom03:# lsrpdomain
```

```
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
----   -------   -----------------   -------------   ------   ------
test   Offline   2.4.7.1             No              12347    12348
```

4. DB2 フィックスパックをスタンバイ・ノードにインストールします。

   オプション: AIX では、当該フィックスパックの前提条件になっている RSCT のインストールが必要となる場合があります。

```
root@grom03:# ./installFixPack -b /opt/ibm/db2/V9.8
DBI1017I  installFixPack is updating the DB2 product(s) installed in
      location /opt/ibm/db2/V9.8.
```

   DB2 フィックスパックのインストールが開始されます。

5. ノードを開始してリソース・グループをオンラインにします。

   インストールが正常に完了したら、以下のようにします。

```
root@grom03:# startrpdomain test

root@grom03:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  -----------------   -------------   -------  -------
test   Online    2.4.7.1             Yes             12347    12348

root@grom03:# chrg -o Online db2_stevera_grom03_0-rg
```

6. フィックスパックが適用され、HADR が再度ピア状態になったことを確認します。

```
stevera@grom03% db2level

stevera@grom03% db2pd hadr db SVTDB
```

7. テークオーバーを実行します。

   他のノード (この場合は grom04) をアップグレードするには、テークオーバーを実行して、ノード grom03 で HADR 1 次データベースをホスティングします。

```
root@grom03:# su - stevera

stevera@grom03% db2 takeover hadr on db SVTDB
DB20000I  TAKEOVER HADR ON DATABASE コマンドが正常に実行されました。
(The TAKEOVER HADR ON DATABASE command completed successfully.)

root@grom03:# lssam
Online IBM.ResourceGroup:db2_stevera_grom03_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom03_0-rs
                '- Online IBM.Application:db2_stevera_grom03_0-rs:grom03
Online IBM.ResourceGroup:db2_stevera_grom04_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom04_0-rs
                '- Online IBM.Application:db2_stevera_grom04_0-rs:grom04
Online IBM.ResourceGroup:db2_stevera_stevera_SVTDB-rg Nominal=Online
        |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs
                |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs:grom03
                '- Offline IBM.Application:db2_stevera_stevera_SVTDB-rs:grom04
        '- Online IBM.ServiceIP:db2ip_9_26_124_22-rs
                |- Online IBM.ServiceIP:db2ip_9_26_124_22-rs:grom03
                '- Offline IBM.ServiceIP:db2ip_9_26_124_22-rs:grom04
```

8. ノード grom04 でアップグレードを実行します。

```
root@grom03:# ssh root@grom04

root@grom04:# chrg -o Offline db2_stevera_grom04_0-rg
```

```
root@grom04:# lssam g db2_stevera_grom04_0-rg
Offline IBM.ResourceGroup:db2_stevera_grom04_0-rg Nominal=Offline
        '- Offline IBM.Application:db2_stevera_grom04_0-rs
                '- Offline IBM.Application:db2_stevera_grom04_0-rs:grom04

root@grom04:# stoprpnode grom04
```

オプション: AIX では、当該フィックスパックの前提条件になっている RSCT のインストールが必要となる場合があります。

```
root@grom04:# ./installFixPack -b /opt/ibm/db2/V9.8
DBI1017I  installFixPack is updating the DB2 product(s) installed in
      location /opt/ibm/db2/V9.8
```

DB2 フィックスパックのインストールが開始されます。

インストールが正常に完了したら、以下のようにします。

```
root@grom04:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  ------------------  --------------  -------  -------
test   Offline   2.4.7.1             Yes             12347    12348

root@grom04:# startrpdomain test

root@grom04:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  ------------------  --------------  -------  -------
test   Online    2.4.7.1             Yes             12347    12348

root@grom04:# chrg -o Online db2_stevera_grom04_0-rg
```

9. フィックスパックが適用されていることを確認 (**db2level** を実行) し、 HADR がピア状態であることを確認 (db2pd hadr db svtdb を実行) します。

```
root@grom04:# su - stevera

stevera@grom04% db2pd -hadr -db svtdb

Database Partition 0 -- Database SVTDB -- Active -- Up 0 days 00:00:05

HADR Information:
Role     State    SyncMode    HeartBeatsMissed    LogGapRunAvg (bytes)
-----    ------   ---------   ----------------    --------------------
Standby  Peer     Sync        0                   0

ConnectStatus   ConnectTime                            Timeout
--------------  ------------                           --------
Connected       Tue May 5 13:20:58 2009 (1241544058)   120

PeerWindowEnd                          PeerWindow
--------------                         -----------
Tue May  5 13:25:58 2009 (1241544358)  300

LocalHost                              LocalService
----------                             -------------
grom04                                 55555

RemoteHost                             RemoteService      RemoteInstance
-----------                            --------------     ---------------
grom03                                 55555              stevera

PrimaryFile    PrimaryPg    PrimaryLSN
------------   ----------   -----------
S0000001.LOG   1            0x0000000003389487
```

```
StandByFile    StandByPg    StandByLSN          StandByRcvBufUsed
------------   ----------   -----------         ------------------
S0000001.LOG   1            0x0000000003389487  0%

root@grom04:# lssam
Online IBM.ResourceGroup:db2_stevera_grom03_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom03_0-rs
                '- Online IBM.Application:db2_stevera_grom03_0-rs:grom03
Online IBM.ResourceGroup:db2_stevera_grom04_0-rg Nominal=Online
        '- Online IBM.Application:db2_stevera_grom04_0-rs
                '- Online IBM.Application:db2_stevera_grom04_0-rs:grom04
Online IBM.ResourceGroup:db2_stevera_stevera_SVTDB-rg Nominal=Online
        |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs
                |- Online IBM.Application:db2_stevera_stevera_SVTDB-rs:grom03
                '- Offline IBM.Application:db2_stevera_stevera_SVTDB-rs:grom04
        '- Online IBM.ServiceIP:db2ip_9_26_124_22-rs
                |- Online IBM.ServiceIP:db2ip_9_26_124_22-rs:grom03
                '- Offline IBM.ServiceIP:db2ip_9_26_124_22-rs:grom04
```

10. TSA ドメインをマイグレーションします。

```
root@grom04:# export CT_MANAGEMENT_SCOPE=2

root@grom04:# runact -c IBM.PeerDomain CompleteMigration Options=0
Resource Class Action Response for CompleteMigration

root@grom04:# samctrl -m

Ready to Migrate! Are you Sure? [Y|N]:.

Y
```

11. クラスター・コンポーネントの **MixedVersions** の設定が Yes ではなくなったことを確認します。

```
root@grom04:# lsrpdomain
Name   OpState   RSCTActiveVersion   MixedVersions   TSPort   GSPort
-----  --------  ------------------  --------------  -------  -------
test   Online    2.5.1.2             No              12347    12348
```

12. アクティブ・バージョン番号 (AVN) が、 HA マネージャーのインストール・バージョン番号 (IVN) と一致することを確認します。

```
root@grom04:# lssrc ls IBM.RecoveryRM |grep VN
   Our IVN         : 2.2.0.7
   Our AVN         : 2.2.0.7
```

13. オプション: grom04 マシンを HADR 1 次 (元の状態) にするために、grom04 マシンでインスタンス所有者「stevera」としてテークオーバーを実行します。

スプリット・ミラーを使用したデータベースのクローン作成

DB2 pureScale 環境以外の環境でクローン・データベースを作成する場合は、以下の手順に従ってください。クローン・データベースに書き込むことはできますが、一般的に、レポート作成など、読み取り専用の処理で使用されます。

このタスクについて

1 次データベースがログをアーカイブするように構成されている場合、クローン・データベースも、同じログ・アーカイブ構成を共有します。アーカイブ・ログの場所がクローン・データベースからアクセス可能である場合、クローン・データベースは、 1 次データベースと同じ場所にログ・ファイルをアーカイブする可能性があります。その場合、両方のデータベースのリカバリー可能性に影響を与えることがあります。クローン・データベースは、最初は 1 次データベースと異なるログ・チ

ェーンを使用しますが、 1 次データベースが、最終的にクローン・データベースと同じログ・チェーン値を使用するようになる場合があります。リカバリー可能性に関する問題を回避するには、 **db2inidb** コマンドを実行する前に、クローン・データベースのログのアーカイブ先を、1 次データベースとは別の場所に変更する必要があります。

クローン・データベースをバックアップして、そのバックアップ・イメージを元のシステムにリストアしたり、また元のシステムで作成したログ・ファイルを使用してロールフォワードしたりすることはできません。クローン・データベースは、I/O をサスペンドした時のデータベースの瞬間的なコピーを提供するだけのものです。つまり、クローンに対して **db2inidb** コマンドを実行すると、コミットされていなかった他の未完了の処理はすべてロールバックされます。

手順

データベースのクローンを作成する方法は、以下のとおりです。

1. 次のコマンドを使用して、1 次データベース上で入出力書き込み操作をサスペンドします。

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

   注: データベースが中断状態のときは、他のユーティリティーやツールを実行しないようにしてください。データベースのコピーを作成するだけにしてください。オプションで、**SET WRITE SUSPEND** を発行する前にすべてのバッファー・プールをフラッシュして、リカバリー・ウィンドウを最小化することができます。それには FLUSH BUFFERPOOLS ALL ステートメントを使用します。
2. 適切なオペレーティング・システム・レベルおよびストレージ・レベルのコマンドを使用して、 1 次データベースから 1 つまたは複数のスプリット・ミラーを作成します。

   注: ボリューム・ディレクトリーを含め、データベース・ディレクトリー全体をコピーするようにしてください。さらに、データベース・ディレクトリー外にある、ログ・ディレクトリーおよびコンテナー・ディレクトリーもコピーする必要があります。この情報を収集するには、DBPATHS 管理ビューを参照してください。このビューは、分割する必要のあるデータベースのすべてのファイルとディレクトリーを表示します。
3. 次のコマンドを使用して、1 次データベース上で入出力書き込み操作を再開します。

   ```
   db2 set write resume for database
   ```
4. 2 次システムのミラー・データベースをカタログします。

   注: デフォルトでは、ミラー・データベースは、 1 次データベースと同じシステムに存在できません。これは、同じディレクトリー構造を持ち、1 次データベースと同じインスタンス名を使用する、 2 次システム上に置く必要があります。ミラー・データベースを、1 次データベースと同じシステムに置かなければならない場合、 **db2relocatedb** ユーティリティーか、 **db2inidb** コマンドの **RELOCATE USING** オプションを使用して、このことを実現できます。
5. 次のコマンドを使用して、2 次システムでデータベース・インスタンスを開始します。

```
db2start
```

6. 2 次システムでミラー・データベースを初期化します。

   ```
   db2inidb database_alias as snapshot
   ```

   必要であれば、**db2inidb** コマンドの RELOCATE USING オプションを指定して、クローン・データベースを再配置します。

   ```
   db2inidb database_alias as snapshot relocate using relocatedbcfg.txt
   ```

   ここで、relocatedbcfg.txt ファイルには、データベースを再配置するのに必要な情報が示されています。

   注:

   - このコマンドにより、分割時に未了であったトランザクションがロールバックされ、新規ログ・チェーン・シーケンスが開始されます。そのため、1 次データベースからのいずれのログもクローン・データベース上で適用すること (ロールフォワード・リカバリーで用いること) はできません。
   - 1 次データベースがログをアーカイブするように構成されている場合、クローン・データベースも、同じログ・アーカイブ構成を共有します。これは、クローン・データベースが、1 次データベースが使用するのと同じ場所にログ・ファイルをアーカイブしようとすることを意味します (クローン・データベースからその場所がアクセス可能な場合)。クローン・データベースは、最初は 1 次データベースと異なるログ・チェーンを使用しますが、 1 次データベースが、最終的にクローン・データベースと同じログ・チェーン値を使用するようになる場合があります。このため、クローン・データベースがアーカイブしたログ・ファイルの上に、 1 次データベースがログ・ファイルをアーカイブする (またはその逆) 可能性があります。これは、両方のデータベースのリカバリー可能性に影響を与えることがあります。この問題を回避するには、クローン・データベースのログのアーカイブ先を、 1 次データベースとは別の場所に変更する必要があります。

DB2 pureScale 環境でスプリット・ミラーを使用してデータベースのクローンを作成する

DB2 pureScale環境でクローン・データベースを作成するには、以下の手順に従ってください。クローン・データベースに書き込むことはできますが、一般的に、レポート作成など、読み取り専用の処理で使用されます。

このタスクについて

1 次データベースがログをアーカイブするように構成されている場合、クローン・データベースも、同じログ・アーカイブ構成を共有します。アーカイブ・ログの場所がクローン・データベースからアクセス可能である場合、クローン・データベースは、 1 次データベースと同じ場所にログ・ファイルをアーカイブする可能性があります。その場合、両方のデータベースのリカバリー可能性に影響を与えることがあります。クローン・データベースは、最初は 1 次データベースと異なるログ・チェーンを使用しますが、 1 次データベースが、最終的にクローン・データベースと同じログ・チェーン値を使用するようになる場合があります。リカバリー可能性に

関する問題を回避するには、 **db2inidb** コマンドを実行する前に、クローン・データベースのログのアーカイブ先を、1 次データベースとは別の場所に変更する必要があります。

クローン・データベースをバックアップして、そのバックアップ・イメージを元のシステムにリストアしたり、また元のシステムで作成したログ・ファイルを使用してロールフォワードしたりすることはできません。クローン・データベースは、I/O をサスペンドした時のデータベースの瞬間的なコピーを提供するだけのものです。つまり、クローンに対して **db2inidb** コマンドを実行すると、コミットされていなかった他の未完了の処理はすべてロールバックされます。

手順

データベースのクローンを作成する方法は、以下のとおりです。

1. 1 次クラスターの設定を取り出してインポートすることにより、2 次クラスター上に General Parallel File System (GPFS) を構成します。1 次クラスター上で、以下の GPFSコマンドを実行します。

   ```
   mmfsctl filesystem syncFSconfig -n remotenodefile
   ```

   ここで、*remotenodefile* は、2 次クラスター内のホストのリストです。
2. 次のコマンドを使用して、クラスター・マネージャーのドメインをリストします。

   ```
   db2cluster -cm -list -domain
   ```
3. 次のコマンドを使用して、各ホスト上のクラスター・マネージャーを停止します。

   ```
   db2cluster -cm -stop -host host -force
   ```

   注: このコマンドを発行するホストを最後にシャットダウンする必要があります。
4. 次のコマンドを使用して、2 次システムで GPFS クラスターを停止します。

   ```
   db2cluster -cfs -stop -all
   ```
5. 次のコマンドを使用して、1 次データベース上で入出力書き込み操作をサスペンドします。

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

   注: データベースが中断状態のときは、他のユーティリティーやツールを実行しないようにしてください。データベースのコピーを作成するだけにしてください。オプションで、**SET WRITE SUSPEND** を発行する前にすべてのバッファー・プールをフラッシュして、リカバリー・ウィンドウを最小化することができます。それには FLUSH BUFFERPOOLS ALL ステートメントを使用します。
6. 次のコマンドを使用して、サスペンドしてコピーする必要のあるファイル・システムを判断します。

   ```
   db2cluster -cfs -list -filesystem
   ```
7. 次のコマンドを使用して、データまたはログ・データを含む各 GPFS ファイル・システムをサスペンドします。

   ```
   /usr/lpp/mmfs/bin/mmfsctl filesystem suspend-write
   ```

ここで *filesystem* は、データまたはログ・データを含むファイル・システムを表しています。

注: GPFS ファイル・システムがサスペンドされている間は、書き込み操作のみがブロックされます。

8. 適切なオペレーティング・システム・レベルおよびストレージ・レベルのコマンドを使用して、 1 次データベースから 1 つまたは複数のスプリット・ミラーを作成します。

   注: ボリューム・ディレクトリーを含め、データベース・ディレクトリー全体をコピーするようにしてください。さらに、データベース・ディレクトリー外にあるログ・ディレクトリー (すべてのログ・ストリーム・サブディレクトリーも含む)、およびコンテナー・ディレクトリーもコピーする必要があります。

9. サスペンドした各 GPFS ファイル・システムに対して次のコマンドを使用して、ファイル・システムをサスペンド状態から再開します。

   ```
   /usr/lpp/mmfs/bin/mmfsctl filesystem resume
   ```

   ここで *filesystem* は、データまたはログ・データを含む、サスペンドされたファイル・システムを表しています。

10. 1 次データベース上で入出力書き込み操作を再開します。

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

11. 次のコマンドを使用して、2 次システムで GPFS クラスターを開始します。

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

12. 次のコマンドを使用して、クラスター・マネージャーを開始します。

    ```
    db2cluster -cm -start -domain domain
    ```

13. 2 次システムのミラー・データベースをカタログします。

    注: デフォルトでは、ミラー・データベースは、 1 次データベースと同じシステムに存在できません。これは、同じディレクトリー構造を持ち、1 次データベースと同じインスタンス名を使用する、 2 次システム上に置く必要があります。ミラー・データベースを、1 次データベースと同じシステムに置かなければならない場合、 **db2relocatedb** ユーティリティーか、 **db2inidb** コマンドの **RELOCATE USING** オプションを使用して、このことを実現できます。

14. 次のコマンドを使用して、2 次システムでデータベース・インスタンスを開始します。

    ```
    db2start
    ```

15. 次のコマンドを使用して、2 次システムでミラー・データベースを初期化します。

    ```
    db2inidb database_alias as snapshot
    ```

    必要であれば、**db2inidb** コマンドの **RELOCATE USING** オプションを指定して、クローン・データベースを再配置します。

    ```
    db2inidb database_alias as snapshot relocate using relocatedbcfg.txt
    ```

    ここで、relocatedbcfg.txt ファイルには、データベースを再配置するのに必要な情報が示されています。

注:

- このコマンドにより、分割時に未了であったトランザクションがロールバックされ、新規ログ・チェーン・シーケンスが開始されます。そのため、1 次データベースからのいずれのログもクローン・データベース上で適用すること (ロールフォワード・リカバリーで用いること) はできません。
- 1 次データベースがログをアーカイブするように構成されている場合、クローン・データベースも、同じログ・アーカイブ構成を共有します。ログのアーカイブ先がクローン・データベースからアクセス可能であれば、スタンバイ・データベースは、ロールフォワード実行中に、そこからログ・ファイルを自動的にリトリーブします。ただし、データベースがロールフォワード・ペンディング状態ではなくなると、クローン・データベースは、1 次データベースが使用しているのと同じ場所に、ログ・ファイルをアーカイブしようとします。スタンバイ・データベースは、最初は 1 次データベースと異なるログ・チェーンを使用しますが、 1 次データベースが、最終的にクローン・データベースと同じログ・チェーン値を使用するようになる場合があります。このため、クローン・データベースがアーカイブしたログ・ファイルの上に、 1 次データベースがログ・ファイルをアーカイブする (またはその逆) 可能性があります。これは、両方のデータベースのリカバリー可能性に影響を与えることがあります。この問題を回避するには、クローン・データベースのログのアーカイブ先を、 1 次データベースとは別の場所に変更する必要があります。

シナリオ: システム・クロックの変更

システム・クロックを調整または変更するときに、DB2 データベース・マネージャーを停止する必要はありません。 DB2 Database for Linux, UNIX, and Windows は、世界中の夏時間調整の変更を年に 2 回、問題なく正常に処理します。

全システムのクロックを同期化する NTP を使った構成も完全にサポートされています。

このタスクについて

システム時刻を変更する際に覚えておくと良いベスト・プラクティスがいくつかあります。

制約事項

ほとんどのシナリオでは、システム・クロックを変更しても、影響はまったくありません。

時刻が大幅に変更される場合、次の 2 つのシチュエーションに注意する必要があります。

- ポイント・イン・タイム・リカバリーを実行する場合、時刻の大幅な変更を意識する必要があります。
- 関数定義には、タイム・スタンプの形式で、その作成日時が含まれます。関数呼び出しの際、DB2 Database for Linux, UNIX, and Windows は、関数定義の解決を試みます。関数解決の一環として、作成時に関数定義に記録されたタイム・ス

タンプ値がチェックされます。関数が作成された時刻よりも前の時刻にシステム・クロックを戻す場合、DB2 Database for Linux, UNIX, and Windows はそれらの関数への参照を解決しません。

手順

これら 2 つのシチュエーションを回避するためのベスト・プラクティスは、次のとおりです。

1. 時刻を進める場合、ステップ 3 に進みます。
2. *X* 分時刻を戻す場合、次のようにします。
   a. 過去 *X* 分以内に新規関数が作成されておらず、*X* 分以内に更新トランザクションが発生していない場合、変更を実行する時刻を選択します。
   b. ステップ a で説明されている時刻が分からない場合でも、DB2 Database for Linux, UNIX, and Windows オンラインによって、システム・クロックの時刻を *X* 分戻すことができます。ただし、次の含意を受け入れる必要があります。
      - ポイント・イン・タイム・リカバリーを使用してその *X* 分以内の時点にリカバリーすることができない可能性があります。つまり、その X 分以内に実行された更新トランザクションのサブセットをリカバリーできない可能性があります。
      - 変更前 *X* 分以内に作成された関数は、変更後 X 分の間は解決されない可能性があります。
3. システム・クロックを変更します。

タスクの結果

概略されているベスト・プラクティスに従うことにより、システム・クロック変更時のポイント・イン・タイム・リカバリーまたは関数解決におけるあらゆる潜在的な問題を回避することができます。

1 次データベースとスタンバイ・データベースの同期化

高可用性ストラテジーの 1 つに、1 次データベースで障害が発生した場合にその 1 次データベースと 2 次またはスタンバイ・データベースで操作をテークオーバーさせるというものがあります。スタンバイ・データベースが障害の発生した 1 次データベースのためにデータベース操作をテークオーバーしなければならない場合、スタンバイ・データベースには全く同じデータを含まれていて、処理中のトランザクションすべてについても把握している必要があります。さらに、1 次データベース・サーバーで障害が発生していなかった場合と全く同様の方法でデータベース処理を続行しなければなりません。1 次データベースのコピーとなるようスタンバイ・データベースを継続的に更新するプロセスのことを同期と言います。

始める前に

1 次データベースとスタンバイ・データベースを同期するには、その前に以下の事柄を行う必要があります。

- 1 次データベースとスタンバイ・データベースを作成および構成します。
- 1 次データベースとスタンバイ・データベース間の通信を構成します。

- 同期ストラテジーを選択します (例えば、ログ・シッピング、ログ・ミラーリング、サスペンド入出力とディスク・ミラーリング、または HADR。)

  1 次データベース・サーバーとスタンバイ・データベース・サーバーの同期を維持するためのストラテジーを、以下に幾つか記します。

  – 1 次データベースからスタンバイ・データベースにログを送り、スタンバイ・データベースにそれらのログをロールフォワードします。
  – 1 次データベースとスタンバイ・データベースの両方にデータベース・ログを同時に作成します。ログ・ミラーリングと呼ばれています。
  – ディスク・ミラーリングでサスペンド入出力サポートを使用して 1 次データベースのコピーを定期的に作成し、そのミラーを分割して、新しいスタンバイ・データベース・サーバーとしてそのコピーを初期化します。
  – さらに、DB2 高可用性災害時リカバリー (HADR) フィーチャーなどの可用性フィーチャーを使用して、1 次データベースとスタンバイ・データベースの同期を維持します。

手順

1. 1 次データベースと 2 次データベースまたはスタンバイ・データベースを同期するためにログを使用している場合、必要なログ管理を実行するように DB2 データベースを構成してください。 例えば、DB2 データベースでログのミラーリングを行う場合、**mirrorlogpath** 構成パラメーターをログの 2 番目のコピーを保管する場所に設定します。
2. DB2 データベースのサスペンド入出力機能を使用して 1 次データベースのディスク・ミラーを分割している場合には、以下を実行する必要があります。
   a. 1 次データベースのディスク・ミラーリングを初期化します。
   b. 1 次データベースのミラーを分割する必要がある場合、『スプリット・ミラーのスタンバイ・データベースとしての使用』に記されている指示に従ってください。
3. HADR フィーチャーを使用して 1 次データベースとスタンバイ・データベースの同期を管理している場合、HADR 用に DB2 データベースを構成し、DB2 データベースにおいて 1 次データベースとスタンバイ・データベースの同期を許可してください。

DB2 高可用性災害時リカバリー (HADR) において複製される操作

DB2 高可用性災害時リカバリー (HADR) では、データベース・ログを使用して、データを 1 次データベースからスタンバイ・データベースに複製します。ログはスタンバイ・データベースで再生されるため、スタンバイ・データベースは一部のアクティビティーが原因で 1 次データベースに後れを取る場合があります。アクティビティーによっては頻繁に記録されるものがあるため、大量のログ・ファイルが生成されることでストレージの問題が発生する場合もあります。ログを使ってデータをスタンバイ・データベースに複製することは可用性の中核を成すストラテジーですが、ロギング自体はソリューションの可用性に負の影響を与える可能性があります。保守ストラテジーは賢明に設計し、ロギングによる負の影響が最小限に抑えられるようにシステムを構成するとともに、ロギングによってトランザクション・データを保護することができるようにしてください。

高可用性災害時リカバリー (HADR) では、以下の操作が、 1 次データベースからスタンバイ・データベースへ複製されます。

- データ定義言語 (DDL)
- データ操作言語 (DML)
- バッファー・プール操作
- 表スペース操作
- オンライン再編成
- オフライン再編成
- ストアード・プロシージャーおよびユーザー定義関数 (UDF) のメタデータ (ただし、関連オブジェクトまたはライブラリー・ファイルではない)

オンライン再編成時には、すべての操作が詳細にログに記録されます。そのため、HADR は、より標準的なデータベース更新の場合よりも、スタンバイ・データベースを遅れさせることなく、操作を複製できます。しかし、この動作では、大量のログ・レコードが生成されるため、システムに大きな影響が出る可能性があります。

オフライン再編成は、オンライン再編成ほど詳細にログに記録されませんが、通常は、再編成の影響を受けた数百か数千の行ごとに、操作がログに記録されます。つまり、スタンバイ・データベースは、各ログ・レコードを待機してから、多数の更新を一度に再生するため、後れを取る可能性があるということです。オフライン再編成がクラスター化されていない場合、再編成操作全体の完了後に、1 つのログ・レコードが生成されます。この方法では、スタンバイ・データベースが 1 次データベースに後れを取らない役割を果たす上で、大変大きな影響を与えます。スタンバイ・データベースは、1 次データベースからログ・レコードを受け取った後に再編成全体を実行します。

HADR は、ストアード・プロシージャー、UDF オブジェクト、およびライブラリー・ファイルを複製しません。 1 次データベースとスタンバイ・データベースの両方で、同じパスにファイルを作成する必要があります。スタンバイ・データベースが、参照されているオブジェクトまたはライブラリー・ファイルを検出できない場合、スタンバイ・データベースでのストアード・プロシージャーまたは UDF の呼び出しは失敗します。

DB2 高可用性災害時リカバリー (HADR) において複製されない操作

DB2 高可用性災害時リカバリー (HADR) では、データベース・ログを使用して、データを 1 次データベースからスタンバイ・データベースに複製します。ログに記録されない操作は、1 次データベースで可能ですが、スタンバイ・データベースに複製されません。ログに記録されない操作 (例えば履歴ファイルの更新) をスタンバイ・データベースに反映させる場合は、さらに別のステップが必要になります。

以下は、1 次データベース上の操作がスタンバイ・データベースに複製されない場合の例です。

- NOT LOGGED INITIALLY オプションを指定して作成された表は複製されません。 HADR スタンバイ・データベースが 1 次データベースとしてテークオーバーした後にそのような表にアクセスしようとすると、エラーが出されます。

- ログに記録される LOB 列はすべて複製されます。ログに記録されない LOB 列は複製されません。しかし、それらのスペースは、列値として 2 進ゼロを使用し、スタンバイ・データベースに割り振られます。
- **UPDATE DATABASE CONFIGURATION** および **UPDATE DATABASE MANAGER CONFIGURATION** コマンドを使用したデータベース構成に対する更新は、複製されません。
- データベース構成およびデータベース・マネージャー構成パラメーターは複製されません。
- ユーザー定義関数 (UDF) の場合、データベースから見て外部にあるオブジェクト (関連オブジェクトおよびライブラリー・ファイルなど) への変更は複製されません。それらの変更は、他の方法でスタンバイ上にセットアップする必要があります。
- リカバリー履歴ファイル (db2rhist.asc) とそれに対する変更は、 1 次データベースからスタンバイ・データベースへ自動的に送られるわけではありません。

  REPLACE HISTORY FILE パラメーターを指定した **RESTORE DATABASE** コマンドを発行することにより、 (1 次データベースのバックアップ・イメージから入手した) 履歴ファイルの初期コピーを、スタンバイ・データベースに置くことができます。

  ```
  RESTORE DB KELLY REPLACE HISTORY FILE
  ```

  HADR が初期設定され、1 次データベースで後続のバックアップ・アクティビティーが実行されると、スタンバイ・データベースの履歴ファイルは古くなってしまいます。しかし、履歴ファイルのコピーは各バックアップ・イメージに保管されます。次のコマンドを使用して、バックアップ・イメージから履歴ファイルを抽出することによって、スタンバイ・データベースの履歴ファイルを更新することができます。

  ```
  RESTORE DB KELLY HISTORY FILE
  ```

  データベース・ディレクトリー内の履歴ファイルを 1 次データベースからスタンバイ・データベースにコピーする場合、通常のオペレーティング・システム・コマンドは使用しないでください。コピー実行時に 1 次がファイルを更新中である場合、履歴ファイルが破損する恐れがあります。

  テークオーバー操作が行われ、スタンバイ・データベースに最新の履歴ファイルがある場合、新しい 1 次データベースでのバックアップおよびリストア操作により、履歴ファイルに新しいレコードが生成され、元の 1 次データベースで生成されたレコードとシームレスに混合させられます。履歴ファイルが古いか、項目が欠落している場合、自動増分リストアはできない可能性があります。代わりに、手動での増分リストア操作が必要になります。

DB2 高可用性災害時リカバリー (HADR) スタンバイ・データベースの状態

高可用性災害時リカバリー (HADR) スタンバイ・データベースは常に、5 つの状態 (ローカル・キャッチアップ、リモート・キャッチアップ・ペンディング、リモート・キャッチアップ、ピア、または切断済みピア) のいずれかになっています。こ

れらの状態はログ・シッピング状態によって定義されます。ログ再生は、状態に関係なく並行して行われ、使用可能なログはすべて再生されます。

1 次ログの位置、スタンバイ・ログの受信位置、およびスタンバイ・ログの再生位置はすべて、HADR の標準的なモニター・インターフェースで報告されます (MON_GET_HADR 表関数、および **-hadr** パラメーターを指定した **db2pd** コマンドを使用)。スタンバイの状態は HADR_STATE フィールドで報告されます。 1 次データベースがスタンバイ・データベースに接続されている場合、モニター・インターフェースでスタンバイの状態が HADR_STATE として報告されます。それ以外の場合は、DISCONNECTED と報告されます。

208 ページの図 10 には、さまざまなスタンバイ・データベース状態における進行状況が示されています。

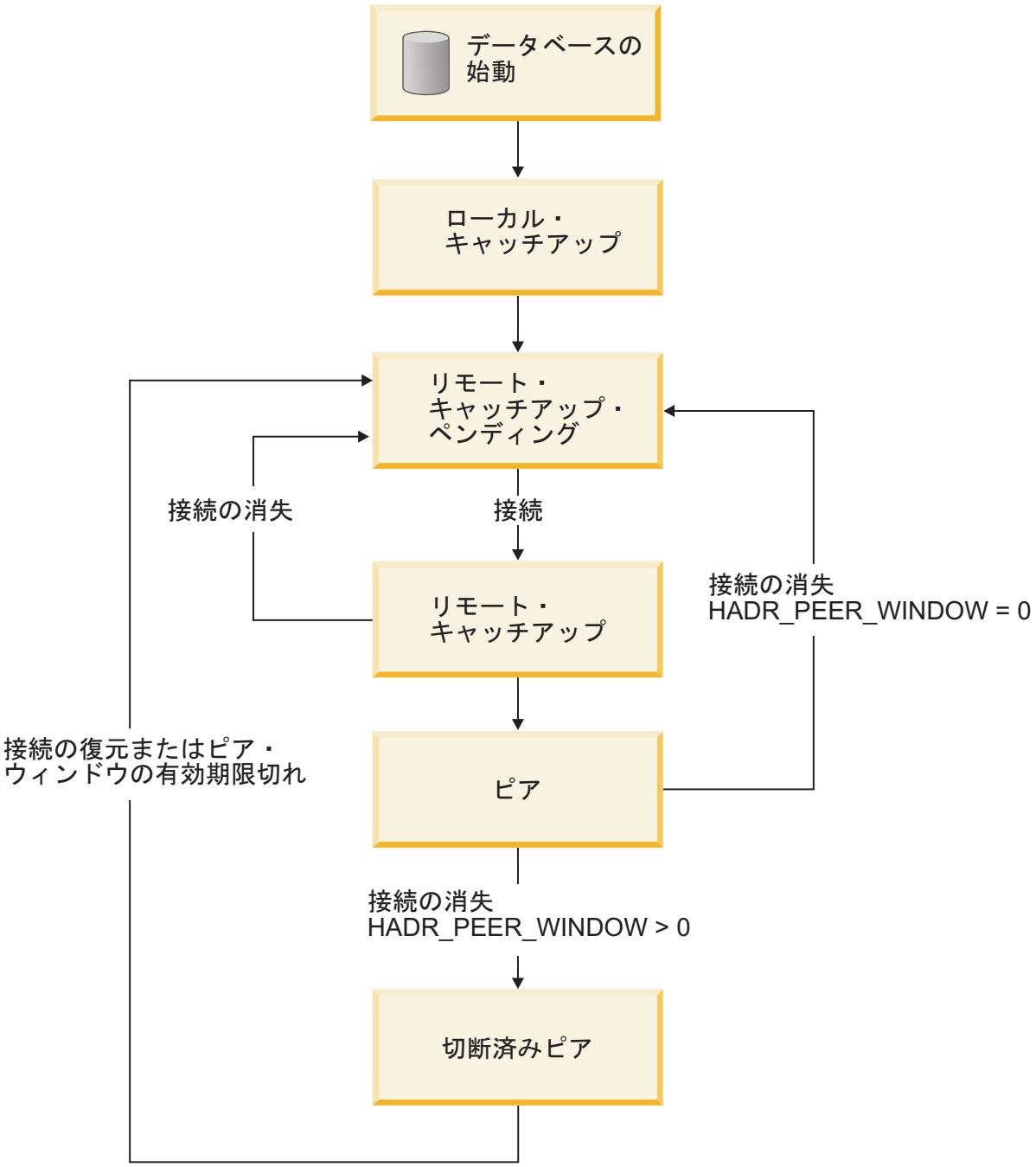


図 10. スタンバイ・データベースの状態

ローカル・キャッチアップ状態

HADR フィーチャーを使用する場合、スタンバイ・データベースが開始されるとローカル・キャッチアップ状態になり、そのローカル・ログ・パスのログ・ファイルが読み取られ、ローカルで使用可能なログが判別されます。この状態では、ログ・アーカイブ方式を構成した場合でも、ログはアーカイブからリトリーブされません。また、この状態では、1 次データベースへの接続は必要ありません。ただし、接続が存在しない場合、スタンバイ・データベースは 1 次データベースに接続しようとします。ローカル・ログ・ファイルの最後になったら、スタンバイ・データベースは、リモート・キャッチアップ・ペンディング状態になります。

リモート・キャッチアップ・ペンディング状態

リモート・キャッチアップ・ペンディング状態になると、1 次データベースへの接続が確立されていない場合、スタンバイは接続されるまで待機します。接続が確立された後、スタンバイは 1 次の現在のログ・チェーン情報を取得します。ログ・アーカイブが構成されている場合は、これでスタンバイがアーカイブからログ・ファイルをリトリーブして、ログ・ファイルが有効であることを確認できます。

リモート・キャッチアップ状態およびピア状態では、スタンバイは 1 次から切断された場合、リモート・キャッチアップ・ペンディング状態に戻ります。接続が再確立されると、スタンバイはアーカイブからログをリトリーブしようとします。したがって、共有アーカイブ装置を構成する場合、スタンバイでは別個のアーカイブ装置を使用する場合より多くの使用可能なログを検出できる可能性があります。この動作では、1 次データベースへの影響を最小限に抑えるために、HADR 接続を介した 1 次からのシッピングより、アーカイブからのリトリーブが優先されます。

リモート・キャッチアップ状態

リモート・キャッチアップ状態では、1 次データベースは、そのログ・パスから、またはログ・アーカイブ方式を使用してログ・データを読み取り、ログ・データはスタンバイ・データベースに送信されます。スタンバイ・データベースが 1 次データベースのディスク上のログ・データをすべて受け取ると、1 次データベースとスタンバイ・データベースはピア状態になります。SUPERASYNC 同期モードを使用している場合、1 次とスタンバイがピア状態になることはありません。これらは完全にリモート・キャッチアップ状態になるため、ピア状態での 1 次ログの書き込みがブロックされる可能性がなくなります。

1 次データベースとスタンバイ・データベースがリモート・キャッチアップ状態であるときにそれらのデータベースの間の接続が消失する場合、スタンバイ・データベースは、リモート・キャッチアップ・ペンディング状態になります。

ピア状態

ピア状態では、1 次がログ・ページをディスクにフラッシュしたときは常に、ログ・データが 1 次のログ書き込みバッファーから直接スタンバイに送られます。HADR 同期モードでは、ログ・データが受信されたことを示す確認通知メッセージをスタンバイが送信するまで 1 次が待機するかどうかを指定します。ログ・ページは常にスタンバイ・データベースのローカル・ログ・ファイルに書き込まれます。古い 1 次にファイルがアーカイブされていない場合、この動作は、テークオーバー時のクラッシュを防ぎ、新しい 1 次にファイルをアーカイブできるようにします。受信されたログ・ページは、ローカル・ディスクに書き込まれた後、スタンバイ・データベースで再生できます。ログ・スプーリングが使用不可の場合 (デフォルト)、ログ受信バッファーの読み取りログのみを再生します。

ログ再生に長い時間がかかる場合、受信バッファー・ログがいっぱいになっている可能性があり、スタンバイでの新規ログの受信は停止します。この場合、1 次ログの書き込みがブロックされます。ログ・スプーリングを使用可能にすると、ログ・バッファーの一部が、まだ再生されていない場合でも解放されます。その後、ログ再生でディスクからログ・データがリードバックされます。スプーリング装置がい

っぱいになっている場合、または構成されたスプール制限に達した場合、スタンバイでの受信は停止したままで、1 次はブロックされたままの状態である可能性があります。

1 次データベースとスタンバイ・データベースがピア状態であるときにそれらのデータベースの間の接続が失われ、**hadr_peer_window** データベース構成パラメーターが 0 (デフォルト) に設定されている場合、スタンバイ・データベースはリモート・キャッチアップ・ペンディング状態になります。ただし、1 次データベースとスタンバイ・データベースがピア状態であるときにそれらのデータベースの間の接続が失われ、**hadr_peer_window** パラメーターをゼロ以外の値 (*ピア・ウィンドウ* が構成されていることを意味します) に設定している場合、スタンバイ・データベースは切断済みピア状態になります。

切断済みピア状態

ピア・ウィンドウを構成した後、1 次データベースがピア状態でスタンバイ・データベースとの接続を失うと、構成された時間の間 (*ピア・ウィンドウ* と呼ばれる)、またはスタンバイが再接続されるまでは (いずれか先に発生した方が優先される)、1 次データベースは、1 次データベースとスタンバイ・データベースがピア状態であるかのように作動し続けます。1 次データベースとスタンバイ・データベースは切断されているが、ピア状態であるかのように作動する状態を *切断済みピア* と呼びます。

ピア・ウィンドウを構成する利点は、複数の障害または連鎖する障害の発生時にトランザクションの損失のリスクが低くなることです。ピア・ウィンドウを使用しない場合、1 次データベースは、スタンバイ・データベースとの接続を失うとすぐにピア状態ではなくなり、トランザクション処理を続行します。これらのトランザクションはスタンバイに複製されません。スタンバイとの接続を失った直後に 1 次サーバーで障害が発生した場合、フェイルオーバーでのトランザクションの損失のリスクが高くなります。ピア・ウィンドウを使用可能にすることで、1 次データベースは、ピア状態でスタンバイへの接続を失った後、一定時間の間、トランザクション処理をブロックして、連鎖する障害から保護します。また、スタンバイは、データ損失のリスクを伴うことなく、ピア・ウィンドウ内でテークオーバーすることができます。

ピア・ウィンドウを構成することの欠点は、1 次データベースがピア・ウィンドウにあり、スタンバイ・データベースとの接続が復元されるか、ピア・ウィンドウの有効期限が切れるまで待っている間に、1 次データベース上のトランザクションに長い時間がかかり、タイムアウトになることさえあるという点です。また、断続的なネットワーク障害が 1 次でのトランザクション処理に重大な影響を与える可能性があります。

ピア・ウィンドウのサイズは、**hadr_peer_window** データベース構成パラメーターの値で判別できます。これは、MON_GET_HADR 表関数、または **-hadr** パラメーターを指定した **db2pd** コマンドを使用して行います。

1 次データベースからスタンバイ・データベースへのログ・ファイルの手動によるコピー

1 次データベースおよびスタンバイ・データベースを同期化するための 1 つの方法は、1 次データベースのログ・ファイルをスタンバイ・データベースのログ・パスまたはオーバーフロー・ログ・パス (構成されている場合) に手動でコピーするという方法です。これは、1 次とスタンバイの間に (例えば、スタンバイ・データベースが長時間停止したことが原因で) 大きなログ・ギャップがある場合に特に役立ちます。この方法を使用することで、アーカイブからログをリトリーブする必要があるスタンバイの遅延を減らすことも、これらのログ・ファイル間でシッピングを行う必要がある (アーカイブからリトリーブしなければならない可能性がある) 1 次への影響を減らすこともできます。このステップは、スタンバイ・データベースをアクティブ化する前に実行することが重要です。スタンバイ・データベースはアクティブになると、前述のように、ローカル・ログ・ファイルの検索、アーカイブからのリトリーブの試行、およびログ・シッピングのための 1 次の使用を続行します。アクティブ化された後にスタンバイにログ・ファイルをコピーすると、通常の操作が妨げられます。

HADR スタンバイ・データベースの状態の判別

DB2 高可用性災害時リカバリー (HADR) スタンバイ・データベースで実行可能な操作は、そのデータベースの状態に応じて決まります。スタンバイの状態を判別するためのオプションとして、**db2pd** コマンドおよび MON_GET_HADR 表関数の 2 つが推奨されます。

手順

1 次およびスタンバイ HADR データベースのペアの HADR スタンバイ・データベースの状態を調べるには、以下を実行します。

- 1 次データベースまたはスタンバイ・データベースから、**-hadr** パラメーターを指定した **db2pd** コマンドを発行します。
  - このコマンドを 1 次データベースから発行すると、HADR セットアップ内の各スタンバイのデータ・セットが返されます。
  - このコマンドをスタンバイ・データベースから発行すると、HADR セットアップが複数スタンバイ・モードであっても、スタンバイは他のスタンバイを認識しないため、1 つのデータ・セットのみが返されます。
- 1 次データベースまたはスタンバイ・データベースで、以下のように MON_GET_HADR 表関数を使用して照会を実行します。

  ```
  db2 "select STANDBY_ID, HADR_STATE, from table (mon_get_hadr(NULL))"
  ```

  次の情報が戻されます。

  ```
  STANDBY_ID HADR_STATE
  ---------- ----------------------
           1 PEER
           2 REMOTE_CATCHUP
           3 REMOTE_CATCHUP

   3 record(s) selected.
  ```

  - この照会を 1 次データベースに対して実行すると、表関数は HADR セットアップ内の各スタンバイの 1 行の情報を返します。

– この照会をスタンバイ・データベースに対して実行すると、HADR セットアップが複数スタンバイ・モードであっても、スタンバイは他のスタンバイを認識しないため、表関数は 1 行の情報のみを返します。

HADR スタンバイ上の表スペース・エラーからのリカバリー

HADR スタンバイ・データベースのログ再生中に、特定の表スペースでエラーが発生した場合、スタンバイ・データベースは、他の表スペースに対するログの再生を継続しますが、影響を受けた表スペースに対するログの再生は停止します。

このタスクについて

影響を受けた表スペースの表スペース状態は、「リストア・ペンディング」、「ロールフォワード・ペンディング」、または「オフライン」に変更されます。スタンバイ上で、この表スペースをリカバリーする必要があります。なぜなら、このデータベースが 1 次の役割を引き継いだ場合、この表スペース内のデータが使用できなくなるためです。

手順

1. エラーの根本原因を訂正します。 考えられる原因は以下のとおりです。
   - スペース不足
   - ファイル・システムがマウントされていない
   - ロード・コピーがみつからない
2. 影響を受けた表スペースを修復します。これを行うには、1 次データベースのバックアップ・イメージをリストアして、スタンバイ・データベースを完全に初期化します。

HADR 遅延再生

HADR 遅延再生は、問題のあるトランザクションによるデータ損失を防ぐのに役立ちます。HADR 遅延再生を実装するには、HADR スタンバイ・データベースで **hadr_replay_delay** データベース構成パラメーターを設定します。

遅延再生では、スタンバイ・データベースでログの再生を遅らせることにより、意図的にスタンバイが 1 次データベースより前の時点のものになるようにします。問題のあるトランザクションが 1 次で実行された場合、構成された遅延時間が経過するまで、問題のあるトランザクションがスタンバイで再生されないように処置を行うことができます。データ損失をリカバリーするために、このデータを再び 1 次にコピーすることも、スタンバイを新しい 1 次データベースとしてテークオーバーさせることもできます。

遅延再生は、(1 次で生成される) ログ・ストリームのタイム・スタンプとスタンバイの現行時刻を比較することで機能します。したがって、1 次データベースとスタンバイ・データベースのクロックを同期することが重要です。トランザクション・コミットは以下の式に従ってスタンバイで再生されます。

```
(current time on the standby - value of the hadr_replay_delay configuration parameter) >=
 timestamp of the committed log record
```

hadr_replay_delay データベース構成パラメーターは、1 次の問題のあるトランザクションを検出して対処するのに十分な大きさの値に設定する必要があります。

このフィーチャーは、単一スタンバイ・モードまたは複数スタンバイ・モードで使用できます。複数スタンバイ・モードでは、通常、高可用性または災害時リカバリーのために 1 つ以上のスタンバイを 1 次と同期が取られるようにし、1 つのスタンバイを問題のあるトランザクションから保護するために遅延再生されるように構成します。このフィーチャーを単一スタンバイ・モードで使用する場合は、IBM Tivoli System Automation for Multiplatforms を使用可能にしないでください。テークオーバーが失敗します。

遅延再生に関する重要な制約事項をいくつか以下に示します。

- **hadr_replay_delay** 構成パラメーターはスタンバイ・データベースでのみ設定可能です。
- 遅延再生を使用可能にしたスタンバイでは、**TAKEOVER** コマンドは失敗します。最初に、**hadr_replay_delay** 構成パラメーターを 0 に設定し、スタンバイを非アクティブにしてから再度アクティブにして新しい値を取得した後で、**TAKEOVER** コマンドを発行してください。
- 遅延再生フィーチャーは SUPERASYNC モードの場合のみサポートされています。ログ再生が遅れるため、スタンバイで多くの未再生ログ・データが累積され、受信バッファーとスプール (構成されている場合) がいっぱいになる可能性があります。他の同期モードでは、これにより 1 次がブロックされます。

  このフィーチャーの目的は、アプリケーション・エラーから保護することにあります。このフィーチャーを使用して 1 次で障害が発生したときにデータが失われないようにする場合は、複数スタンバイ・セットアップにおけるプリンシパル・スタンバイの同期設定を増やすことを検討してください。

推奨事項

遅延再生と災害時リカバリー

災害時リカバリーや、問題のあるトランザクションからの保護を目的としてスタンバイ・データベースを使用する場合は、遅延を短くすることを検討してください。

遅延再生と HADR スタンバイ・データベースの読み取りフィーチャー

リーダー・セッションでより最新のデータを確認できるように、スタンバイ・データベースの読み取りのためにスタンバイ・データベースを使用する場合は、遅延を短くすることを検討してください。また、スタンバイ・データベースの読み取りは「非コミット読み取り」分離レベルで実行されるため、適用されたがまだコミットされていない (厳密には、依然として再生より遅延している) 変更を確認できます。これらのコミットされていないトランザクションは、必要な PIT までスタンバイをロールフォワードしてから停止する際に、問題のあるトランザクションのリカバリー手順でロールバックされる可能性があります。

遅延再生とログ・スプーリング

遅延再生を使用可能にする場合は、**hadr_spool_limit** データベース構成パラメーターを設定して、ログ・スプーリングも使用可能にすることをお勧めします。意図的な遅延により、スタンバイでのログの再生位置がログの受信位置よりかなり後になる場合があります。スプーリングを使用しない場合、ログの受信位置と再生位置の差異は、受信バッファーの量を超えて大きくすることはできません。スプーリングを使用すると、スタンバイは再生位置を

超える量のログを受信することができ、1 次で障害が発生した場合のデータ損失に対する保護を強化することができます。いずれの場合も、SUPERASYNC モードが必須であるため、1 次が遅延再生によりブロックされることはありません。

HADR 遅延再生を使用したデータのリカバリー

HADR 遅延再生フィーチャーを使用して、1 次データベースで問題のあるトランザクションにより失われたデータを、そのトランザクションが再生される前にスタンバイの HADR を停止することでリカバリーできます。

始める前に

遅延再生はスタンバイ・データベースで既に使用可能になっている必要があります。

STANDBY_REPLAY_LOG_TIME で指定されたスタンバイでのログ再生時間が、スタンバイで問題のあるトランザクションのコミット時間を過ぎると、以降の手順を使用してデータをリカバリーすることはできません。
STANDBY_REPLAY_LOG_TIME は、**-hadr** パラメーターを指定した **db2pd** コマンド、または MON_GET_HADR 表関数を使用して判別できます。

制約事項: **hadr_replay_delay** 構成パラメーターを設定したスタンバイ・データベースは 1 次としてテークオーバーできません。そのスタンバイで最初に遅延再生を使用不可にする必要があります。

手順

問題のあるトランザクションからリカバリーするには、遅延再生を使用可能にしたスタンバイで以下のステップを実行します。

1. 次の手順でタイミングを確認します。
   a. スタンバイがまだトランザクションを再生していないことを確認します。STANDBY_REPLAY_LOG_TIME 値が問題のあるトランザクションのコミット時間に達していてはなりません。
   b. スタンバイが関係のあるログを受信したことを確認します。ログの受信日時を示す STANDBY_LOG_TIME 値が、問題のあるトランザクションのコミット時間より前 (直前) の PIT に達している必要があります。これがステップ 3 で使用したロールフォワード PIT になります。スタンバイがまだ十分なログ・ファイルを受信していない場合、ログがさらに送信されるまで待機できますが、再生時間が問題のあるトランザクションの時間に達する危険があります。例えば、遅延が 1 時間の場合、ログ・シッピングが希望する PIT にまだ達していない場合であっても、問題のあるトランザクションの時間が経過してから 50 分以内に HADR を停止する必要があります (許容される安全マージンは 10 分です)。

      あるいは、共有ログ・アーカイブが使用可能であり、ログが既にアーカイブされている場合には、待機する必要はありません。ログがまだアーカイブされていない場合は、**ARCHIVE LOG** コマンドを使用して、そのログをアーカイブできます。それ以外の場合、ユーザーは 1 次から遅延時間を指定したスタンバイにすべてのログ・ファイルを手動でコピーすることができます (オー

バーフロー・ログ・パスが推奨されますが、これを使用しない場合はログ・パスを使用してください)。これらの代替方式では、最初にスタンバイを非アクティブにして、スタンバイのログ・シッピングと再生が干渉されないようにします。

これらの時間は、db2pd -db *dbname* -hadr を発行するか、あるいは、スタンバイでスタンバイ・データベースの読み取りフィーチャーを使用可能にしてから、MON_GET_HADR 表関数を使用する以下の照会を実行することで判別できます。

```
DB2 "select HADR_ROLE, STANDBY_ID, STANDBY_LOG_TIME, STANDBY_REPLAY_LOG_TIME,
varchar(PRIMARY_MEMBER_HOST,20) as PRIMARY_MEMBER_HOST,
varchar(STANDBY_MEMBER_HOST,20) as STANDBY_MEMBER_HOST
from table (mon_get_hadr(NULL))"
```

2. スタンバイ・データベースの HADR を停止します。

   ```
   DB2 STOP HADR ON DATABASE dbname
   ```

3. 必要な PIT までスタンバイをロールフォワードしてから停止します。

   ```
   DB2 ROLLFORWARD DB dbname to time-stamp and STOP
   ```

4. 以下のいずれかの方法を使用します。
   - 1 次で失われたデータをリストアします。
     a. スタンバイから該当するデータをコピーして、1 次に送信します。

        問題のあるトランザクションで表がドロップされた場合は、スタンバイでその表をエクスポートしてから 1 次にインポートできます。問題のあるトランザクションで表から行が削除された場合は、スタンバイでその表をエクスポートしてから、1 次でインポート置換操作を使用できます。

     b. 遅延再生を指定したスタンバイを再初期化します。これは、そのログ・ストリームが 1 次のログ・ストリームから分岐しているためです。他のスタンバイは引き続き 1 次のログ・ストリームに従っており、1 次のデータ修復もそれらのスタンバイに複製されるため、アクションは必要ありません。

     c. 1 次で取ったバックアップ・イメージを使用して、データベースをリストアします。このバックアップ・イメージはいつでも取ることができます。

     d. スタンバイ・ログ・パス内のすべてのログ・ファイルを削除します。このステップは重要です。ステップ 3 の **ROLLFORWARD**... STOP コマンドにより、データベース・ログ・ストリームが 1 次から分岐してしまっています。ファイルをそのままにすると、新しくリストアされたデータベースはそのログ・ストリームに従い、1 次からも分岐します。あるいは、最初からやり直すためにリストアする前にデータベースをドロップできますが、そうすると HADR 構成を含む現在の構成も失われます。

     e. データベースで AS STANDBY オプションを指定した **START HADR** コマンドを発行します。これで、データベースはアクティブになり、1 次に接続するはずです。

   - 次のようにして、損傷していないデータを含むスタンバイが 1 次になるようにします。
     a. 分離脳 (1 次が 2 つ) になることを回避するため、古い 1 次をシャットダウンします。

b. 遅延再生を指定したデータベースで、**hadr_replay_delay** 構成パラメーターを 0 に設定します。必要に応じて、**hadr_target_list** などの他のパラメーターを再構成します。次に、そのデータベースで AS PRIMARY BY FORCE オプションを指定した **START HADR** コマンドを実行することで、新しい 1 次に変換します。構成済みのプリンシパル・スタンバイ (古い 1 次) を必ず接続できるとは限らないため、BY FORCE オプションを使用してください。

c. クライアントを新しい 1 次にリダイレクトします。

d. 他のスタンバイは新しい 1 次に自動的にリダイレクトされます。ただし、スタンバイが古い 1 次と新しい 1 次の分岐点 (ステップ 3 で使用した PIT) より後に古い 1 次のログを受け取った場合、新しい 1 次によって拒否されます。このような場合は、古い 1 次の再初期化と同じ手順を使用して、このスタンバイを再初期化してください。

e. 古い 1 次を再初期化します。これは、そのログ・ストリームが新しい 1 次のログ・ストリームから分岐しているためです。

f. 新しい 1 次、またはステップ 3 で使用した PIT の前に古い 1 次で取ったバックアップ・イメージを使用して、データベースをリストアします。

g. ログ・パスにあるすべてのログ・ファイルを削除します。これを行わないと、新しくリストアしたデータベースが古い 1 次のログ・ストリームに従い、新しい 1 次から分岐します。あるいは、最初からやり直すために、リストアする前にデータベースをドロップできますが、そうした場合 HADR 構成を含む現在の構成も失われます。

h. データベースで AS STANDBY オプションを指定した **START HADR** コマンドを発行します。これで、データベースはアクティブになり、1 次に接続するはずです。

DB2 高可用性災害時リカバリー (HADR) 管理

DB2 高可用性災害時リカバリー (HADR) の管理には、 HADR システムの状況を構成することと保守することが含まれます。

HADR の管理には以下のタスクが含まれます。

- 36 ページの『高可用性災害時リカバリーの初期設定 (HADR)』
- 188 ページの『DB2 高可用性災害時リカバリー (HADR) の停止』
- 264 ページの『高可用性災害時リカバリーでのデータベース役割の切り替え (HADR)』
- 261 ページの『HADR フェイルオーバー操作の実行』
- 255 ページの『高可用性災害時リカバリー (HADR) のモニター』
- HADR に関連するデータベース構成パラメーターの検査または変更。
- HADR データベースのカタログ。

以下の方法を用いて HADR を管理できます。

- コマンド行プロセッサー
- DB2 管理 API

- IBM Data Studio バージョン 3.1 以降で HADR を管理するためのタスク・アシスト。

関連情報:

タスク・アシストを使用したデータベースの管理

DB2 高可用性災害時リカバリー (HADR) コマンド

DB2 高可用性災害時リカバリー (HADR) フィーチャーは、DB2 高可用性データベース・ソリューションに対して、複雑なロギング、フェイルオーバー、およびリカバリー機能を提供します。 HADR が提供する機能は複雑なものですが、HADR コマンドを直接実行する必要があるアクションはごくわずかです。HADR を開始すること、HADR を停止すること、およびスタンバイ・データベースに 1 次データベースをテークオーバーさせることだけです。

HADR の管理に使用される高可用性災害時リカバリー (HADR) コマンドは以下の 3 つです。

- **START HADR**
- **STOP HADR**
- **TAKEOVER HADR**

これらのコマンドを呼び出すには、コマンド行プロセッサーまたは管理 API を使用します。

AS PRIMARY または AS STANDBY オプションを指定して **START HADR** コマンドを発行すると、データベースがまだそのデータベース役割になっていない場合、指定された役割に変わります。また、このコマンドを発行すると、データベースがまだアクティブでない場合、アクティブになります。

STOP HADR コマンドは、HADR データベース (1 次またはスタンバイ) を標準データベースに変更します。HADR に関連したデータベース構成パラメーターがあれば、データベースを HADR データベースとして再度アクティブにしやすいように、未変更のまま残されます。

TAKEOVER HADR コマンドはスタンバイ・データベースでのみ発行可能で、スタンバイ・データベースを 1 次データベースに変更します。BY FORCE オプションを指定しないと、1 次データベースとスタンバイ・データベースの役割が切り替わります。BY FORCE オプションを指定すると、一方的にスタンバイ・データベースが 1 次データベースに切り替わります。この場合、スタンバイ・データベースは古い 1 次データベース上でのトランザクション処理を停止しようと試みます。しかし、トランザクション処理が停止するという保証はありません。BY FORCE オプションを使用して強制的にテークオーバー操作を実行するのは、フェイルオーバー状態の場合だけにしてください。BY FORCE オプションを指定して **TAKEOVER HADR** コマンドを発行する前に、可能な範囲で、現在の 1 次が間違いなく障害を起こしているということを確認するか、それを自分でシャットダウンするようにしてください。

HADR データベース役割の切り替え

データベースの 1 次役割と標準役割との間の切り替えは、動的に、かつ繰り返し行うことができます。データベースがオンラインとオフラインのどちらであっても、

START HADR コマンドに AS PRIMARY オプションを指定して発行することと、**STOP HADR** コマンドを発行することの両方が可能です。

データベースのスタンバイ役割と標準役割との間の切り替えは、静的に行うことができます。これを繰り返し行えるのは、データベースがロールフォワード・ペンディング状態に留まる場合だけです。データベースがオフラインで、ロールフォワード・ペンディング状態である場合、**START HADR** コマンドに AS STANDBY オプションを指定して発行し、標準データベースをスタンバイに変更することができます。データベースがオフラインの場合にスタンバイ・データベースを標準データベースに変更するには、**STOP HADR** コマンドを使用します。**STOP HADR** コマンドを発行した後も、データベースはロールフォワード・ペンディング状態のままです。その後、**START HADR** コマンドに AS STANDBY オプションを指定して発行すると、データベースはスタンバイに戻ります。スタンバイ・データベース上で HADR を停止した後、**ROLLFORWARD DATABASE** コマンドに STOP オプションを指定して発行すると、これを再びスタンバイにすることはできなくなります。データベースはロールフォワード・ペンディング状態ではなくなったので、これを標準データベースとして使用できます。これを、スタンバイ・データベースのスナップショットを取ると言います。既存のスタンバイ・データベースを標準データベースに変更した後は、高可用性を目的として新しいスタンバイ・データベースを作成することを考慮してください。

1 次データベースの役割とスタンバイ・データベースの役割を切り替えるには、BY FORCE オプションを指定しないでテークオーバー操作を実行します。

一方的にスタンバイを 1 次に変更する (1 次をスタンバイに変更しない) 場合、強制テークオーバーを使用します。その後、古い 1 次を新しいスタンバイとして再統合できるかもしれません。

HADR 役割は永続的です。HADR 役割は一度確立されると、DB2 インスタンスの停止や開始、または DB2 データベースの非アクティブ化およびアクティブ化が繰り返されても、データベースに残ります。

スタンバイの開始は非同期

START HADR コマンドに AS STANDBY オプションを指定して発行した場合、関連するエンジン・ディスパッチ可能単位 (EDU) が正常に開始されるとすぐに、そのコマンドは戻ります。コマンドはスタンバイが 1 次データベースに接続するのを待機しません。対照的に 1 次データベースは、スタンバイ・データベースに接続するまで、開始されたとは見なされません (**START HADR** コマンドに BY FORCE オプションを指定して、1 次データベースに発行した場合を除く)。スタンバイ・データベースで、1 次データベースによる接続拒否などのエラーが発生した場合、AS STANDBY オプションを指定した **START HADR** コマンドはすでに正常に戻されている可能性があります。その結果、HADR がエラー表示を戻すことのできるユーザー・プロンプトがないということになります。HADR スタンバイは、DB2 診断ログにメッセージを書き込み、シャットダウンします。HADR スタンバイの状況をモニターして、正常に HADR 1 次と接続したことを確認する必要があります。

再生エラー (ログ・レコードの再生中にスタンバイで発生するエラー) も、スタンバイ・データベースをダウンさせる場合があります。これらのエラーは、例えばバッファー・プールを作成するのに十分なメモリーがない場合、または表スペース作成

中にパスが見つからない場合などに発生することがあります。スタンバイ・データベースの状況を、継続してモニターすることが必要です。

クライアント・リルート可能なデータベース別名を使用してクライアントから HADR コマンドを実行しない

自動クライアント・リルートがセットアップされている場合、データベース・サーバーは、事前定義された代替サーバーを持ちます。これは、クライアント・アプリケーションが元のデータベース・サーバーと代替サーバーとの間で作業を切り替えて、作業の中断を最低限で済ませるためです。そのような環境でクライアントが TCP を介してデータベースに接続すると、実際の接続先は、元のデータベースか代替データベースのどちらかになります。HADR コマンドは、通常のクライアント接続ロジックを介してターゲット・データベースを識別するようインプリメントされます。その結果、ターゲット・データベースで代替データベースが定義されている場合、コマンドが実際に操作しているデータベースを判別することは困難です。SQL クライアントは接続先のデータベースを知らなくてもかまいませんが、その一方で HADR コマンドは特定のデータベースに対して適用される必要があります。この制限に対処するには、HADR コマンドをサーバー・マシン上でローカルに発行し、クライアント・リルートを回避する必要があります (クライアント・リルートは TCP/IP 接続にしか影響しません)。

HADR コマンドは有効なライセンスを持つサーバー上で実行しなければならない

START HADR、**STOP HADR**、および **TAKEOVER HADR** コマンドを実行するには、コマンドを実行するサーバー上に有効な HADR ライセンスをインストールしておくことが必要です。ライセンスがないとこれらのコマンドは失敗し、コマンドに特定のエラー・コード (それぞれ SQL1767N、SQL1769N、または SQL1770N) が理由コード 98 と一緒に戻されます。問題を訂正するには、**db2licm** を使用して有効な HADR ライセンスをインストールするか、または有効な HADR ライセンスがディストリビューションの一部として付属しているバージョンのサーバーをインストールします。

HADR 複数スタンバイ・データベース

高可用性災害時リカバリー (HADR) フィーチャーは、複数のスタンバイ・データベースをサポートします。複数スタンバイを使用すると、データを 3 箇所以上の場所に配置でき、単一のテクノロジーでより高いデータ保護を提供します。

複数スタンバイ・モードで HADR フィーチャーをデプロイすると、セットアップにスタンバイ・データベースを 3 つまで含めることができます。これらのデータベースのうちの 1 つを*プリンシパル HADR スタンバイ・データベース* として指定します。他のスタンバイ・データベースはすべて、*補助 HADR スタンバイ・データベース* になります。どちらのタイプの HADR スタンバイも直接 TCP/IP 接続を介して HADR 1 次データベースと同期され、スタンバイ・データベースでの読み取りをサポートし、遅延ログ再生を構成することができます。さらに、どのスタンバイにおいても強制または非強制テークオーバーを実行できます。ただし、プリンシパル・スタンバイと補助スタンバイには次のような重要な違いがいくつかあります。

- IBM Tivoli System Automation for Multiplatforms (SA MP) の自動フェイルオーバーはプリンシパル・スタンバイでのみサポートされます。補助スタンバイの 1 つに対して手動でテークオーバーを実行し、そのスタンバイを 1 次にする必要があります。
- HADR 同期モードはすべてプリンシパル・スタンバイでサポートされていますが、補助スタンバイでは、SUPERASYNC モードにすることのみ可能です。

複数 HADR スタンバイ・セットアップを使用することには、いくつかの利点があります。高可用性の目標を達成するために HADR フィーチャーを使用したり、災害時リカバリーの目標を達成するために別のテクノロジーを使用したりする代わりに、HADR を使用することによってその両方の目標を達成できます。プリンシパル・スタンバイは 1 次と同じ場所にデプロイできます。1 次が停止した場合、リカバリー目標時間内でプリンシパル・スタンバイが 1 次の役割をテークオーバーできます。また、補助スタンバイを遠隔地にデプロイして、1 次とプリンシパル・スタンバイの両方に影響する広範囲に渡る災害から保護することもできます。 1 次と補助の間の距離、およびネットワーク遅延の可能性は 1 次のアクティビティーには影響しません。これは、補助で SUPERASYNC モードが使用されるためです。災害が 1 次とプリンシパル・スタンバイに及んだ場合は、補助のいずれかでテークオーバーを実行できます。 **hadr_target_list** データベース構成パラメーターを使用して、もう 1 つの補助スタンバイ・データベースが新しいプリンシパル・スタンバイになるように構成できます。ただし、補助スタンバイに使用可能なスタンバイがない場合であっても、補助は 1 次としてテークオーバーできます。例えば、1 次とプリンシパル・スタンバイが停止した場合、1 つの補助は、対応するスタンバイがなくても 1 次としてテークオーバーできます。ただし、そのデータベースが新しい 1 次になった後で停止された場合、そのプリンシパル・スタンバイが始動しない限り、HADR 1 次として再始動することはできません。

複数スタンバイ・データベースに関する制約事項

HADR フィーチャーを複数スタンバイ・モードでデプロイする計画において注意すべき制約事項がいくつかあります。

制約事項は以下のとおりです。

- 最大で 3 つのスタンバイ・データベース (プリンシパル・スタンバイを 1 つと、補助スタンバイを 2 つまで) を持つことができます。
- すべての HADR 同期モードをサポートするのはプリンシパル・スタンバイのみです。補助スタンバイはすべて SUPERASYNC モードになります。
- IBM Tivoli System Automation for Multiplatforms (SA MP) サポートは、1 次 HADR データベースとそのプリンシパル・スタンバイ間でのみ適用されます。
- 複数スタンバイ・セットアップ内のすべてのデータベースで、**hadr_target_list** データベース構成パラメーターが設定されている必要があります。各スタンバイのその **hadr_target_list** 設定に 1 次を含める必要があります。

複数スタンバイ・モードでの HADR の初期化

複数スタンバイ・モードでの HADR システムの初期化は、単一スタンバイ・モードの場合と似ています。主な違いは、セットアップ内のすべてのデータベースの **hadr_target_list** データベース構成パラメーターを設定することにより、複数スタンバイ・モードを有効にする必要があることです。

このタスクについて

このタスクでは、複数スタンバイ・モードで HADR を初期化する方法について説明します。単一スタンバイ・セットアップを複数スタンバイ・セットアップに変換する場合は、 223 ページの『既存の HADR セットアップでの複数スタンバイ・モードの使用可能化』を参照してください。

複数スタンバイ・モードでは、関係するすべてのデータベース上に **hadr_target_list** 構成パラメーターを設定する必要があります。このパラメーターは、そのデータベースが 1 次になる場合のシナリオのスタンバイをリストします。これは、スタンバイでも必要になります。また、相互包括が必要です (つまり、A のターゲット・リストに B が含まれる場合、B のターゲット・リストに A が含まれている必要があります)。これにより、スタンバイからのテークオーバー後、新しい 1 次は常に古い 1 次をそのスタンバイとして維持できます。ターゲット・リストで指定した最初のスタンバイが、*プリンシパル HADR スタンバイ・データベース*として指定されます。追加のスタンバイは*補助 HADR スタンバイ・データベース*になります。ターゲット・リストには、必ずしも関係するデータベースをすべて含める必要はありません。また、複数のスタンバイが存在する場合、対称性や相反性に関する要件はありません。つまり、データベース A にそのプリンシパル・スタンバイとしてデータベース B が含まれるように指定されている場合であっても、データベース B のプリンシパル・スタンバイとして A を指定する必要はありません。データベース A のターゲット・リストで指定された各スタンバイのターゲット・リストには、データベース A も含まれている必要があります。データベースごとにターゲット・リストを決定することは重要なステップです。

特例として、複数スタンバイ・モードを 1 つのスタンバイのみで構成できます。例えば、複数スタンバイ・モードで 2 つのデータベースを 1 次およびスタンバイとして構成できます。動作は単一スタンバイ・セットアップの場合とは異なります。複数スタンバイの自動構成などの動作が有効になり、スタンバイ・ターゲットを動的に追加または除去できるためです。

ヒント: ステップ 2 から 4 は、各データベースの単一の更新で実行することができます。

手順

複数スタンバイ・モードで HADR を初期化するには、次のようにします。

1. リストアされたバックアップまたはスプリット・ミラーを使用してスタンバイ・データベースを作成します。この方法については、 57 ページの『スタンバイ・データベースの初期化』、または 36 ページの『高可用性災害時リカバリーの初期設定 (HADR)』のステップ 2 を参照してください。
   - 1 次で以下のコマンドを発行します。

     ```
     BACKUP DB dbname
     ```
   - スタンバイで以下のコマンドを発行します。

     ```
     RESTORE DB dbname
     ```
2. 各データベースで、**hadr_local_host**、**hadr_local_svc**、**hadr_local_svc**、および **hadr_sync_mode** 構成パラメーターを設定します。

```
"UPDATE DB CFG FOR dbname USING
    HADR_LOCAL_HOST   hostname
    HADR_LOCAL_SVC    servicename
    HADR_SYNCMODE     syncmode"
```

3. すべてのスタンバイおよび 1 次で **hadr_target_list** 構成パラメーターを設定します。

```
DB2 UPDATE DB CFG FOR dbname USING
     HADR_TARGET_LIST  principalhostname:principalservicename|
  auxhostname1:auxservicename1|auxhostname2:auxservicename2
```

4. すべてのデータベースで、**hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** 構成パラメーターを設定します。

   複数スタンバイ・モードでは、これらの値が未設定の場合は自動的に設定され、正しく設定されていない場合には自動的に再設定されるため、このステップは不要です。ただし、正しい値に明示的に設定すれば、正しい値をすぐに使用できます。これらの値は、**hadr_remote_inst** 値でリソース名を構成する必要がある可能性のある、IBM Tivoli System Automation for Multiplatforms (SA MP) ソフトウェアで役立ちます。

   - 1 次では、以下のコマンドを発行して、プリンシパル・スタンバイの対応する値にパラメーターを設定します。

     ```
     DB2 "UPDATE DB CFG FOR  dbname USING
          HADR_REMOTE_HOST   principalhostname
          HADR_REMOTE_SVC    principalservicename
          HADR_REMOTE_INST   principalinstname"
     ```

   - 各スタンバイでは、以下のコマンドを発行して、1 次の対応する値にパラメーターを設定します。

     ```
     DB2 "UPDATE DB CFG FOR  dbname USING
          HADR_REMOTE_HOST   primaryhostname
          HADR_REMOTE_SVC    primaryservicename
          HADR_REMOTE_INST   primaryinstname"
     ```

5. 各スタンバイ・インスタンスに接続します。
6. スタンバイ・インスタンスで、以下のように、**AS STANDBY** パラメーターを指定した **START HADR** コマンドを発行します。

   ```
   START HADR ON DB dbname AS STANDBY
   ```

7. 1 次インスタンスに接続します。
8. 1 次インスタンスで、以下のように、**AS PRIMARY** パラメーターを指定した **START HADR** コマンドを発行します。

   ```
   START HADR ON DB dbname AS PRIMARY
   ```

タスクの結果

スタンバイ・データベースは、ローカルで使用可能なログ・ファイルが読み取られて再生されるローカル・キャッチアップ状態で開始されます。すべてのローカル・ログが再生されたら、データベースはリモート・キャッチアップ・ペンディング状態になります。1 次の始動後、スタンバイはリモート・キャッチアップ状態になり、1 次からログ・ページを受け取って再生します。1 次データベースのディスク上にあるすべてのログ・ファイルがスタンバイで再生された後の動作は、同期モードのタイプに応じて次に行われる動作のタイプによって異なります。SUPERASYNC のプリンシパル・スタンバイ、および補助スタンバイはリモート・キャッチアップ・モードのままです。SYNC、NEARSYNC、または ASYNC モードのプリンシパ

ル・スタンバイはピア・モードになります。

既存の HADR セットアップでの複数スタンバイ・モードの使用可能化

複数スタンバイ・モードでの HADR システムの初期化は、単一スタンバイ・モードの場合と似ています。主な違いは、セットアップ内のすべてのデータベースの **hadr_target_list** データベース構成パラメーターを設定することにより、複数スタンバイ・モードを有効にする必要があることです。

始める前に

- 関係するすべてのデータベースのホスト名またはホスト IP アドレス (**hadr_local_host** 設定に使用)、サービス名またはポート番号 (**hadr_local_svc** 設定で使用) を判別します。
- 各データベースのターゲット・リストを判別します。
- データベースが 1 次になる場合は、各データベースのプリンシパル・スタンバイの同期モードおよびピア・ウィンドウを判別します。
- **hadr_timeout** 構成パラメーターの設定を判別します。このパラメーターの設定はすべてのデータベースで同じでなければなりません。
- 1 次と各スタンバイの間に十分なネットワーク帯域幅があるかどうかを判別します。必要に応じてアップグレードします。
- 1 次のネットワーク・インターフェースが追加のスタンバイの出力データ・フローをサポートできるかどうかを判別します。必要に応じてアップグレードします。

このタスクについて

複数スタンバイ・モードでは、関係するすべてのデータベース上に **hadr_target_list** 構成パラメーターを設定する必要があります。このパラメーターは、そのデータベースが 1 次になる場合のシナリオのスタンバイをリストします。これは、スタンバイでも必要になります。また、相互包括が必要です (つまり、A のターゲット・リストに B が含まれる場合、B のターゲット・リストに A が含まれている必要があります)。これにより、スタンバイからのテークオーバー後、新しい 1 次は常に古い 1 次をそのスタンバイとして維持できます。ターゲット・リストで指定した最初のスタンバイが、*プリンシパル HADR スタンバイ・データベース*として指定されます。追加のスタンバイは*補助 HADR スタンバイ・データベース*になります。ターゲット・リストには、必ずしも関係するデータベースをすべて含める必要はありません。また、複数のスタンバイが存在する場合、対称性や相反性に関する要件はありません。つまり、データベース A にそのプリンシパル・スタンバイとしてデータベース B が含まれるように指定されている場合であっても、データベース B のプリンシパル・スタンバイとして A を指定する必要はありません。データベース A のターゲット・リストで指定された各スタンバイのターゲット・リストには、データベース A も含まれている必要があります。データベースごとにターゲット・リストを決定することは重要なステップです。

特例として、複数スタンバイ・モードを 1 つのスタンバイのみで構成できます。例えば、複数スタンバイ・モードで 2 つのデータベースを 1 次およびスタンバイと

して構成できます。動作は単一スタンバイ・セットアップの場合とは異なります。複数スタンバイの自動構成などの動作が有効になり、スタンバイ・ターゲットを動的に追加または除去できるためです。

このタスクでは、最初に新しいスタンバイのみを作成して構成します。最終ステップまで元の構成を維持することで、可能な限り長く 1 次とスタンバイのペアを機能させておくことができます。元のスタンバイの構成の変更が早過ぎると、その新しい構成を取得するためにスタンバイが意図せず非アクティブになってから再びアクティブになった場合に、古い HADR ペアが分裂する可能性があります。

手順

複数スタンバイ・モードで HADR を有効化するには、次のようにします。

1. リストアされたバックアップまたはスプリット・ミラーを使用して、追加のスタンバイ・データベースを作成します。この方法については、57 ページの『スタンバイ・データベースの初期化』、または36 ページの『高可用性災害時リカバリーの初期設定 (HADR)』のステップ 2 を参照してください。
   - 1 次データベースの場合:

     ```
     DB2 BACKUP DB dbname
     ```

   - スタンバイの場合:

     ```
     DB2 RESTORE DB dbname
     ```

2. 以下のように、新しいスタンバイ・データベースをそれぞれ構成します。
   a. **hadr_local_host** および **hadr_local_svc** を、HADR 接続で使用される TCP アドレスに設定します。
   b. **hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** の各構成パラメーターを、1 次データベースを指すように設定します。
   c. すべてのデータベースで同じ設定値を使用して、**hadr_timeout** 構成を設定します。
   d. 以前に計画したように、**hadr_target_list** 構成パラメーターを設定します。
   e. プリンシパル・スタンバイ・データベースが 1 次になる場合は、そのスタンバイの **hadr_syncmode** および **hadr_peer_window** の各構成パラメーターを設定します。
   f. 必要なセットアップに応じて、**hadr_spool_limit** や **hadr_replay_delay** などの HADR 固有の他のパラメーターを設定します。
3. 新しいスタンバイ・インスタンスにそれぞれ接続し、**AS STANDBY** オプションを指定した **START HADR** コマンドを発行します。

   ```
   START HADR ON DB dbname AS STANDBY
   ```

4. ステップ 2 と同じ指示に従って、元のスタンバイを再構成します。
5. 以下のように、1 次を再構成します。
   a. **hadr_local_host** および **hadr_local_svc** を、HADR 接続で使用される TCP アドレスに設定します。より多くのスタンバイに対応する、より広いネットワーク帯域幅をサポートする新しいネットワーク・インターフェース・カード (NIC) を使用する場合は、更新が必要になる場合があります。

b. **hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** の各構成パラメーターを、プリンシパル・スタンバイ・データベースを指すように設定します。
c. すべてのスタンバイ・データベースで同じ設定値を使用して、**hadr_timeout** 構成を設定します。
d. 以前に計画したように、**hadr_target_list** 構成パラメーターを設定します。
e. プリンシパル・スタンバイが使用する、**hadr_syncmode** および **hadr_peer_window** の各構成パラメーターを設定します。
f. 必要なセットアップに応じて、**hadr_spool_limit** や **hadr_replay_delay** などの HADR 固有の他のパラメーターを設定します。

6. 元のスタンバイを非アクティブにしてから再度アクティブにして、新しい構成を取得します。
7. 1 次で HADR を停止してから再始動し、新しい構成を取得します。

タスクの結果

すべてのスタンバイが数秒以内に 1 次に接続するはずです。これらの状態は、-hadr オプションを指定した **db2pd** コマンド、または MON_GET_HADR 表関数を使用してモニターできます。

複数スタンバイ・データベース・セットアップの変更

複数 HADR スタンバイ・セットアップを稼働した後、追加の変更を加えることができます。例えば、補助スタンバイ・データベースを追加、除去したり、プリンシパル・スタンバイ・データベースの指定を変更したりできます。このような変更は 1 次データベースを停止せずに行えます。

補助スタンバイの追加

補助スタンバイを追加する理由はいくつかあります。

- 読み取り専用ワークロードを処理するための追加のスタンバイをデプロイする。
- 遅延再生のための追加のスタンバイをデプロイする。
- 災害時リカバリーのための追加のスタンバイをデプロイする。
- 以前はアクティブな HADR デプロイメントの一部だったが、新しい 1 次の **hadr_target_list** 構成パラメーターでそのスタンバイが指定されていないために *孤立* 状態だったスタンバイを追加する。

補助スタンバイは、HADR デプロイメントが複数スタンバイ・モードである場合にのみ追加できます。つまり、少なくとも 1 つのスタンバイに **hadr_target_list** 構成パラメーターを事前に設定しておく必要があります。

補助スタンバイを HADR デプロイメントに追加するには、スタンバイのホストおよびポート情報で 1 次のターゲット・リストを更新します。この情報は、スタンバイの **hadr_local_host** パラメーターと **hadr_local_svc** パラメーターの設定に対応します。また、1 次のホストおよびポートの情報を新しいスタンバイのターゲット・リストに追加することも必要です。

ヒント: 必須ではありませんが、新しいスタンバイのホストおよびポートの情報をデプロイメント内の他のスタンバイのターゲット・リストにも追加することはベス

ト・プラクティスです。それらのスタンバイのホストおよびポートの情報を新しいスタンバイのターゲット・リストに指定することも必要です。この追加の更新を行わずに他のスタンバイの 1 つが新しい 1 次としてテークオーバーすると、新しいスタンバイはスタンバイのターゲットとして拒否され、シャットダウンされます。

補助スタンバイの除去

動的に除去可能なスタンバイは補助スタンバイのみです。補助スタンバイを複数スタンバイ・デプロイメントから動的に除去しても、1 次およびプリンシパル・スタンバイでの通常の HADR 操作には影響はありません。補助スタンバイを除去するには、対象のスタンバイで **STOP HADR** コマンドを発行します。その後、1 次および他のスタンバイのターゲット・リストからそのスタンバイを除去できます。

プリンシパル・スタンバイの変更

プリンシパル・スタンバイは、最初に 1 次データベースで HADR を停止した場合にのみ変更できます。1 次を非アクティブにする必要はないため、これが停止の原因になることはありません。

プリンシパル・スタンバイを変更するには、1 次データベースで HADR を停止する必要があります。次に、1 次データベースのターゲット・リストを更新して、新しいプリンシパル・スタンバイを最初にリストします。新しいプリンシパル・スタンバイがまだスタンバイでない場合は、1 次データベースのアドレスをそのターゲット・リストに追加し、他の HADR パラメーターを構成して、スタンバイをアクティブにします。既にスタンバイである場合、アクションは不要です。

ヒント: 必須ではありませんが、新しいプリンシパル・スタンバイのホストおよびポートの情報をデプロイメント内の他のスタンバイのターゲット・リストにも追加することはベスト・プラクティスです。そのスタンバイのホストおよびポートの情報を新しいプリンシパル・スタンバイのターゲット・リストに指定することも必要です。この追加の更新を行わずにスタンバイの 1 つが新しい 1 次としてテークオーバーする場合、他方のスタンバイはスタンバイのターゲットとして拒否され、シャットダウンされます。

複数 HADR スタンバイ・データベース用のデータベース構成

複数 HADR スタンバイ・セットアップにはデータベース構成に関するいくつかの考慮事項があります。

HADR パラメーターの自動再構成

HADR 開始後の再構成

複数スタンバイ・モードでは、スタンバイの 1 次データベースを識別する構成パラメーターと 1 次のプリンシパル・スタンバイを識別する構成パラメーターが正しく設定されていない場合、HADR の開始時に自動的にリセットされます。この動作は、以下の構成パラメーターに該当します。

- **hadr_remote_host**
- **hadr_remote_inst**
- **hadr_remote_svc**

ヒント: この再構成は自動で行われますが、常に正しい初期値の設定を試みる必要があります。自動再構成はスタンバイとその 1 次の間の接続が確立されるまで有効にならない可能性があるからです。一部の HADR デプロイメントでは、その初期値が必要になる可能性があります。例えば、IBM Tivoli System Automation for Multiplatforms ソフトウェアを使用する場合、リソース名を構成するために **hadr_remote_inst** 構成パラメーターの値が必要になります。

注: **DB2_HADR_NO_IP_CHECK** レジストリー変数が ON に設定されている場合、**hadr_remote_host** と **hadr_remote_svc** は自動的に更新されません。

再構成では、**hadr_target_list** 構成パラメーターの値が正しいものであることが前提となります。ターゲット・リスト項目に間違いがある場合、手動で訂正する必要があります。

1 次の場合、再構成は次のように行われます。

- **hadr_remote_host** および **hadr_remote_svc** の構成パラメーターの値が **hadr_target_list** 構成パラメーターの最初の項目 (つまり、プリンシパル・スタンバイ) である *host*:*port* ペアと一致しない場合、**hadr_remote_host** と **hadr_remote_svc** の構成パラメーターはターゲット・リストの値で更新されます。
- **hadr_remote_inst** 構成パラメーターの値がプリンシパル・スタンバイのインスタンス名と一致しない場合、プリンシパル・スタンバイが 1 次に接続した後、正しいインスタンス名が 1 次の **hadr_remote_inst** 構成パラメーターにコピーされます。

スタンバイ・データベースの場合、再構成は次のように行われます。

- スタンバイは、始動時に、**hadr_remote_host**、**hadr_remote_inst**、および **hadr_remote_svc** の各構成パラメーターに指定されたデータベースへの接続を試行します。
- スタンバイが 1 次に接続できない場合、1 次がそのスタンバイに接続するのを待機します。
- 1 次は、**hadr_target_list** パラメーターにリストされているアドレスを使用して、そのスタンバイへの接続を試行します。1 次がスタンバイに接続した後、スタンバイの **hadr_remote_host**、**hadr_remote_inst**、および **hadr_remote_svc** の各構成パラメーターが 1 次の正しい値で更新されます。

テークオーバー時、およびその後の再構成

非強制テークオーバーでは、新しい 1 次の **hadr_remote_host**、**hadr_remote_inst**、および **hadr_remote_svc** の各構成パラメーターの値はそのプリンシパル・スタンバイに自動的に更新され、新しい 1 次の **hadr_target_list** にリストされているスタンバイのこれらのパラメーターは新しい 1 次を指すように自動的に更新されます。新しい 1 次の **hadr_target_list** にリストされていないデータベースは更新されません。それらのデータベースは引き続き古い 1 次への接続を試行しますが、古い 1 次はスタンバイになっているため、拒否されます。古い 1 次は、ターゲット・リストで相互包括が必要であるため、新しい 1 次のターゲット・リストに確実に保管されます。

強制テークオーバーでは、新しい 1 次とそのスタンバイ (古い 1 次を除く) の自動更新は非強制テークオーバーと同じように機能します。ただし、古い 1 次の自動更新は、再統合のためにシャットダウンして、スタンバイとして再始動するまでは行われません。

テークオーバーの際にオンラインになっていないデータベースはすべて、その開始後に自動的に再構成されます。自動再構成は、定期的にスタンバイに接続する場合に新しい 1 次に依存するため、開始してすぐには有効にならない可能性があります。スタンバイは開始時に古い 1 次への接続を試行し、古い 1 次のログ・ストリームに従う可能性があります。これにより、スタンバイは新しい 1 次のログ・ストリームから分岐し、そのスタンバイと新しい 1 次をペアにすることができなくなります。したがって、このような*分離脳* のシナリオを回避するために、テークオーバーの前に古い 1 次をシャットダウンする必要があります。

スタンバイが同期モードおよびピア・ウィンドウを制御しない

複数スタンバイ・モードでは、現行の 1 次の **hadr_syncmode** および **hadr_peer_window** 構成パラメーターの設定だけが関係します。スタンバイ・データベースは、1 次 (プリンシパル・スタンバイの場合) またはその補助スタンバイとしての役割によって定義されたそれらのパラメーターの設定を持ちます。

同期モード

複数スタンバイ・モードでは、**hadr_syncmode** 構成パラメーターの設定が 1 次データベースとスタンバイ・データベースで同じである必要はありません。スタンバイで指定される **hadr_syncmode** 構成パラメーターの設定はすべて、*その構成済み同期モード* と見なされます。この設定は、スタンバイが 1 次になる場合にのみ関係します。スタンバイに*有効同期モード* が割り当てられます。どの補助スタンバイでも、有効同期モードは常に SUPERASYNC になります。プリンシパル・スタンバイの場合、有効同期モードは、1 次の **hadr_syncmode** 構成パラメーターの設定です。スタンバイの場合は、モニター・インターフェースに有効同期モードが同期モードとして表示されます。

ピア・ウィンドウ

複数スタンバイ・モードでは、**hadr_peer_window** 構成パラメーターの設定が 1 次データベースとスタンバイ・データベースで同じである必要はありません。実際、ピア・ウィンドウの機能には SUPERASYNC モードとの互換性がないため、補助スタンバイの **hadr_peer_window** 構成パラメーターの設定は無視されます。プリンシパル・スタンバイは 1 次のピア・ウィンドウの設定を使用しますが、これはスタンバイの **hadr_syncmode** 構成パラメーターの値が SYNC または NEARSYNC である場合にのみ当てはまります (単一スタンバイ・モードの場合と同じ)。

HADR 複数スタンバイ・モードでのローリング・アップグレード

HADR 単一スタンバイ・モードと同様に、ローリング・アップグレードを使用できます。決定的な違いは、複数スタンバイの場合、1 次とスタンバイをアクティブのままにして HADR 保護を維持しつつ、この手順を使用できることです。データベース・サービスを提供する 1 次が常にあり、この 1 次には常に高可用性災害時リカバリー保護を提供するスタンバイが少なくとも 1 つは含まれています。

複数スタンバイを使用する場合、更新またはアップグレードを 1 次で実行する前に、すべてのスタンバイで実行しておく必要があります。これは、フィックスパック・レベルを更新する場合に特に重要です。HADR では、1 次のフィックスパック・レベルをスタンバイより高くすることができないためです。

一度に 1 つのデータベースをアップグレードする (補助スタンバイからそのアップグレードを開始する) 必要があることを除くと、手順は単一スタンバイ・モードの場合と基本的には同じです。例えば、以下のような HADR セットアップがあるとします。

- host1 が 1 次である。
- host2 がプリンシパル・スタンバイである。
- host3 が補助スタンバイである。

このセットアップの場合、以下の順序でローリング・アップグレードまたは更新を実行します。

1. host3 を非アクティブにし、必要な変更を加え、host3 をアクティブにしてから host3 (スタンバイ) で HADR を開始します。
2. host3 が 1 次のすべてのログを再生し終えた後で、host2 を非アクティブにして必要な変更を加え、その host2 をアクティブにして host2 (スタンバイ) で HADR を開始します。
3. host2 が host1 のすべてのログを再生し終えて、host 1 とピア状態になったら、host2 でテークオーバーを実行します。
4. host1 を非アクティブにし、必要な変更を加え、host1 をアクティブにしてから host1 (スタンバイ) で HADR を開始します。
5. host1 が host2 とピア状態になったら、host1 でテークオーバーを実行します。これで、host1 は再び 1 次になり、host2 は再びプリンシパル・スタンバイになります。

複数スタンバイ・モードでの高可用性災害時リカバリー (HADR) のモニター

HADR 複数スタンバイ・モードでは単一スタンバイ・モードと同じモニター・インターフェースがサポートされます。ただし、他のモニター・インターフェースではすべてのスタンバイの完全なビューが提供されるわけではないため、**db2pd** コマンドと MON_GET_HADR 表関数のみを使用する必要があります。

モニター・インターフェースが返す情報は、実行場所によって異なります。スタンバイでモニターを実行すると、そのスタンバイと 1 次のみの情報が返され、他のスタンバイの情報は提供されません。1 次でモニターを実行すると、すべてのスタンバイの情報が返されます (**db2pd** コマンドまたは MON_GET_HADR 表関数を使用する場合)。接続されていないスタンバイでも、1 次の **hadr_target_list** 構成パラメーターで構成されていれば表示されます。**GET SNAPSHOT FOR DATABASE** コマンドなどの他のインターフェースでは、1 次とプリンシパル・スタンバイのみが報告されます。

db2pd コマンドと MON_GET_HADR 表関数は基本的には同じ情報を返しますが、**db2pd** コマンドではスタンバイ・データベースの読み取りを (スタンバイからの報告のために) 使用可能にする必要はありません。また、1 次とスタンバイの両方で

クライアント接続が許可されない時間枠が存在する可能性があるため、テークオーバー時は **db2pd** コマンドが推奨されます。

db2pd コマンド

以下の例では、DBA は 3 つのスタンバイを持つ 1 次データベースで **db2pd** コマンドを発行します。3 つのデータ・セットが返され、それぞれ 1 次とスタンバイ間のログ・シッピング・チャネルを示します。HADR_ROLE フィールドには、**db2pd** が発行されたデータベースの役割が示され、すべてのセットで PRIMARY としてリストされます。2 つの補助スタンバイ (hostS2 と hostS3) の HADR_STATE は REMOTE_CATCHUP です。これらのスタンバイは、**hadr_syncmode** に構成された設定に関係なく、SUPERASYNC モード (**db2pd** 出力にも反映されます) で自動的に実行されるためです。スタンバイは STANDBY_ID で区別します。この ID はシステムによって生成され、ID とスタンバイ間のマッピングは照会に応じて変更できます。ただし、ID「1」は常にプリンシパル・スタンバイに割り当てられます。

注: 現在の状況に関係しないフィールドは出力で省略される場合があります。例えば、以下の出力では、再生専用時間枠がアクティブではないため、その情報 (開始時刻やトランザクション数など) は含まれません。

```
db2pd -db hadr_db -hadr

  Database Member 0 -- Database hadr_db -- Active -- Up 0 days 00:23:17 -- Date 06/08/2011 13:57:23

                             HADR_ROLE = PRIMARY
                           REPLAY_TYPE = PHYSICAL
                         HADR_SYNCMODE = SYNC
                            STANDBY_ID = 1
                         LOG_STREAM_ID = 0
                            HADR_STATE = PEER
                    PRIMARY_MEMBER_HOST = hostP.ibm.com
                       PRIMARY_INSTANCE = db2inst1
                         PRIMARY_MEMBER = 0
                    STANDBY_MEMBER_HOST = hostS1.ibm.com
                       STANDBY_INSTANCE = db2inst2
                         STANDBY_MEMBER = 0
                    HADR_CONNECT_STATUS = CONNECTED
               HADR_CONNECT_STATUS_TIME = 06/08/2011 13:38:10.199479 (1307565490)
            HEARTBEAT_INTERVAL(seconds) = 30
                  HADR_TIMEOUT(seconds) = 120
          TIME_SINCE_LAST_RECV(seconds) = 3
               PEER_WAIT_LIMIT(seconds) = 0
             LOG_HADR_WAIT_CUR(seconds) = 0.000
      LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
     LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                    LOG_HADR_WAIT_COUNT = 82
  SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 50772
  SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87616
              PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
              STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                    HADR_LOG_GAP(bytes) = 0
       STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
         STANDBY_RECV_REPLAY_GAP(bytes) = 0
                    PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                    STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
           STANDBY_RECV_BUF_SIZE(pages) = 16
               STANDBY_RECV_BUF_PERCENT = 0
             STANDBY_SPOOL_LIMIT(pages) = 0
                  PEER_WINDOW(seconds) = 0
              READS_ON_STANDBY_ENABLED = Y
      STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N
```

```
                            HADR_ROLE = PRIMARY
                          REPLAY_TYPE = PHYSICAL
                        HADR_SYNCMODE = SUPERASYNC
                           STANDBY_ID = 2
                        LOG_STREAM_ID = 0
                           HADR_STATE = REMOTE_CATCHUP
                    PRIMARY_MEMBER_HOST = hostP.ibm.com
                       PRIMARY_INSTANCE = db2inst1
                         PRIMARY_MEMBER = 0
                    STANDBY_MEMBER_HOST = hostS2.ibm.com
                       STANDBY_INSTANCE = db2ins3t
                         STANDBY_MEMBER = 0
                    HADR_CONNECT_STATUS = CONNECTED
               HADR_CONNECT_STATUS_TIME = 06/08/2011 13:35:51.724447 (1307565351)
            HEARTBEAT_INTERVAL(seconds) = 30
                  HADR_TIMEOUT(seconds) = 120
          TIME_SINCE_LAST_RECV(seconds) = 16
               PEER_WAIT_LIMIT(seconds) = 0
             LOG_HADR_WAIT_CUR(seconds) = 0.000
      LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
     LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                    LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
              PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
              STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                    HADR_LOG_GAP(bytes) = 0
       STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
         STANDBY_RECV_REPLAY_GAP(bytes) = 0
                      PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                      STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
          STANDBY_RECV_BUF_SIZE(pages) = 16
              STANDBY_RECV_BUF_PERCENT = 0
            STANDBY_SPOOL_LIMIT(pages) = 0
                 PEER_WINDOW(seconds) = 0
              READS_ON_STANDBY_ENABLED = Y

                            HADR_ROLE = PRIMARY
                          REPLAY_TYPE = PHYSICAL
                        HADR_SYNCMODE = SUPERASYNC
                           STANDBY_ID = 3
                        LOG_STREAM_ID = 0
                           HADR_STATE = REMOTE_CATCHUP
                    PRIMARY_MEMBER_HOST = hostP.ibm.com
                       PRIMARY_INSTANCE = db2inst1
                         PRIMARY_MEMBER = 0
                    STANDBY_MEMBER_HOST = hostS3.ibm.com
                       STANDBY_INSTANCE = db2inst3
                         STANDBY_MEMBER = 0
                    HADR_CONNECT_STATUS = CONNECTED
               HADR_CONNECT_STATUS_TIME = 06/08/2011 13:46:51.561873 (1307566011)
            HEARTBEAT_INTERVAL(seconds) = 30
                  HADR_TIMEOUT(seconds) = 120
          TIME_SINCE_LAST_RECV(seconds) = 6
               PEER_WAIT_LIMIT(seconds) = 0
             LOG_HADR_WAIT_CUR(seconds) = 0.000
      LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
     LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                    LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
              PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
              STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                    HADR_LOG_GAP(bytes) = 0
       STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
         STANDBY_RECV_REPLAY_GAP(bytes) = 0
                      PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                      STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
```

```
STANDBY_RECV_BUF_SIZE(pages) = 16
    STANDBY_RECV_BUF_PERCENT = 0
  STANDBY_SPOOL_LIMIT(pages) = 0
        PEER_WINDOW(seconds) = 0
   READS_ON_STANDBY_ENABLED = N
```

MON_GET_HADR 表関数

以下の例では、DBA は 3 つのスタンバイを持つ 1 次データベースで **MON_GET_HADR** 表関数を呼び出します。返される行は 3 つです。各行に 1 次とスタンバイ間のログ・シッピング・チャネルが示されます。HADR_ROLE 列には、照会が実行されたデータベースの役割が示されます。したがって、すべての行には PRIMARY が示されています。2 つの補助スタンバイ (hostS2 と hostS3) の HADR_STATE は REMOTE_CATCHUP です。これらのスタンバイは、**hadr_syncmode** に構成された設定に関係なく、SUPERASYNC モードで自動的に実行されるためです。

```
db2 "select HADR_ROLE, STANDBY_ID, HADR_STATE, varchar(PRIMARY_MEMBER_HOST,20)
 as PRIMARY_MEMBER_HOST, varchar(STANDBY_MEMBER_HOST,20)
 as STANDBY_MEMBER_HOST from table (mon_get_hadr(NULL))"

  HADR_ROLE     STANDBY_ID HADR_STATE              PRIMARY_MEMBER_HOST  STANDBY_MEMBER_HOST
  ------------ ---------- ----------------------- -------------------- --------------------
  PRIMARY                1 PEER                    hostP.ibm.com        hostS1.ibm.com
  PRIMARY                2 REMOTE_CATCHUP          hostP.ibm.com        hostS2.ibm.com
  PRIMARY                3 REMOTE_CATCHUP          hostP.ibm.com        hostS3.ibm.com

 3 record(s) selected.
```

HADR 複数スタンバイ・モードでのテークオーバー

HADR スタンバイ・データベースが複数スタンバイ環境で 1 次データベースとしてテークオーバーする際に、単一スタンバイ・モードの場合と比べて重要な相違点がいくつかあります。

HADR でのテークオーバーには、*役割の切り替え* および*フェイルオーバー* という 2 つのタイプがあります。役割の切り替え (正常なテークオーバーまたは非強制テークオーバーと呼ばれる場合があります) は、1 次が使用可能であり、1 次とスタンバイの役割を切り替える場合にのみ実行できます。フェイルオーバー (つまり、強制テークオーバー) は、1 次が使用不可の場合に実行できます。一般的には、1 次で障害が発生した場合にスタンバイを新しい 1 次にするために使用されます。強制テークオーバーでは古い 1 次の役割は 1 次のままです。複数スタンバイ・モードでは両方のタイプのテークオーバーがサポートされ、スタンバイ・データベースのいずれかが 1 次としてテークオーバーできます。ただし、覚えておくべき重要な点があります。それは、スタンバイが新しい 1 次のターゲット・リストに含まれていない場合は、孤立しているものと見なされ、新しい 1 次に接続できないということです。

テークオーバーでは、DB2 はいくつかの構成変更を自動的に行うため、新しい 1 次のターゲット・リストにリストされるスタンバイは新しい 1 次に接続できます。**hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** の各構成パラメーターは、以下の方法で、新しい 1 次とリストされたスタンバイで更新されます。

- **新しい 1 次の場合:** プリンシパル・スタンバイ (新しい 1 次のターゲット・リストにリストされる最初のデータベース) を参照します。

- **スタンバイの場合:** 新しい 1 次を参照します。古い 1 次がスタンバイになるように再統合されている場合、**START HADR** AS STANDBY コマンドは最初にそれをスタンバイに変換します。したがって、新しい 1 次のターゲット・リストにリストされている場合は、新しい 1 次に自動的にリダイレクトすることもできます。

  注: 孤立スタンバイはこの方法では自動的に更新されません。スタンバイとして結合する場合は、その孤立スタンバイが新しい 1 次のターゲット・リストにあり、それ自体のターゲット・リストに新しい 1 次が含まれていることを確認する必要があります。

役割の切り替え

単一スタンバイ・モードと同様に、複数スタンバイ・モードでの役割の切り替えでは、古い 1 次と新しい 1 次の間でデータが失われることはありません。新しい 1 次の **hadr_target_list** 構成パラメーターで構成されている他のスタンバイは新しい 1 次に自動的にリダイレクトされ、引き続きログを受信します。

フェイルオーバー

単一スタンバイ・モードと同様に、複数スタンバイ・モードでのフェイルオーバーによりデータが失われた場合 (新しい 1 次に古い 1 次のデータがすべて含まれているわけではありません)、古い 1 次と新しい 1 次のログ・ストリームは分岐し、古い 1 次を再初期化する必要があります。他のスタンバイでは、スタンバイが分岐点を超えて古い 1 次のログを受け取った場合に、そのスタンバイを再初期化する必要があります。それ以外の場合、スタンバイは新しい 1 次に接続可能であり、ログ・シッピングと再生は引き続き行われます。したがって、すべてのスタンバイのログ位置を確認して、最大量のデータを含むスタンバイをフェイルオーバー・ターゲットとして選択することが重要です。この情報は、**db2pd** コマンドまたは MON_GET_HADR 表関数を使用して照会できます。

注: 新しい 1 次を指すようにスタンバイの **hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** の各構成パラメーターが正しく自動的に再構成されても、スタンバイを新しい 1 次とペアにできるとは限りません。このスタンバイでは 1 次への TCP 接続のみが可能です。接続時に、DB2 が 2 つのデータベースのログ・ストリームが分岐していると判断した場合、ペア要求は拒否され、接続はクローズします。

シナリオ: HADR 複数スタンバイ・データベース・セットアップのデプロイ

このシナリオでは、ExampleBANK という銀行用に HADR セットアップを計画、構成、およびデプロイする方法を説明します。セットアップには 3 つのスタンバイ・データベース (1 つのプリンシパル・スタンバイと 2 つの補助スタンバイ) が含まれます。

背景

銀行業務は 1 日 24 時間週 7 日体制であるため、高可用性は ExampleBANK のテクノロジー戦略には不可欠です。また、ExampleBANK は都市 A を襲ったハリケーンにより危機一髪の事態を経験しました。その都市には本社があるため、銀行は災

害時リカバリー戦略も必要とします。高可用性災害時リカバリー (HADR) は、銀行が 1 つのテクノロジーで両方の目的を達成するために役立つソリューション (HADR 複数スタンバイ・データベース) を提供します。

ExampleBANK は HADR ソリューションに不可欠な以下の要件を考慮します。

積極的なリカバリー時間目標
: 24 時間のオンライン・サービスを提供する銀行として、ExampleBANK は、アプリケーションがデータベースに接続できない時間を最小限に抑える必要があります。

積極的なリカバリー・ポイント目標
: ExampleBANK はデータ損失を許容できないため、RPO をできるだけ 0 に近付ける必要があります。

ゼロに近い計画的ダウン時間
: ExampleBANK のデータベースは、アップグレードやメンテナンスなどのアクティビティーを計画した場合でも、できる限り使用できるようになっている必要があります。

地理的分散によるデータ保護
: 承諾基準の一部として、ExampleBANK は遠隔地で操作をリカバリーする機能を必要とします。

簡単なデプロイメントと管理
: ExampleBANK の多忙を極める IT 部門には、比較的簡単に構成でき、自動機能を持つソリューションが必要です。

以下のシナリオで示すように、複数スタンバイ・モードでの HADR フィーチャーの使用は、ExampleBANK がこれらすべての要件を満たすために役立ちます。

複数スタンバイ・セットアップの計画

ExampleBANK は HADR セットアップの高可用性と災害時リカバリーの両方の保護を受ける必要があるため、最大数 (3 つ) のスタンバイの使用を決定しました。この銀行は、RTO を達成するために、1 次と密接に同期している (SYNC モードまたは NEARSYNC モードを使用している) スタンバイを必要とし、このスタンバイは 1 次と連結されている必要があります。すべての同期モードをサポートするのはこのスタンバイのみであるため、これをプリンシパル・スタンバイとして使用するのが妥当です。1 次とプリンシパル・スタンバイは両方とも都市 A にある ExampleBANK の本社にあり、LAN で接続されています。

また、銀行のデータを災害による消失から保護するために、ExampleBANK の DBA は都市 B にある銀行の支社に 2 つのスタンバイをセットアップすることにします。この支社は、都市 A にある本社に WAN で接続されています。2 つの都市の間の距離は、スタンバイが (SUPERASYNC モードで自動的に稼働する) 補助スタンバイであるため、1 次には影響しません。DBA は、追加データベースのうちの 1 つをスタンバイ・データベースの読み取りフィーチャーを使用するように設定し、その他のデータベースを遅延再生フィーチャーを使用するように設定することで、それらのデータベースのコストをより正当化できます。また、これらのスタンバイはローリング更新またはメンテナンス・シナリオにおいて、HADR 保護を失うことなく、データベースの可用性を維持するのに役立ちます。

複数スタンバイ・セットアップの構成

ExampleBANK の DBA は、以下のようにして、対象の 1 次データベース HADR_DB のバックアップを取ります。

```
DB2 BACKUP DB hadr_db TO backup_dir
```

次に、DBA は、以下のコマンドを発行して、対象のスタンバイ・ホストにそれぞれバックアップをリストアします。

```
DB2 RESTORE DB hadr_db FROM backup_dir
```

ヒント: スタンバイを作成するためのオプションについて詳しくは、57 ページの『スタンバイ・データベースの初期化』を参照してください。

初期セットアップについて、ExampleBANK の DBA は、ほとんどの構成がデフォルト設定で十分であると判断します。ただし、通常の HADR セットアップの場合は、以下のデータベース構成パラメーターを明示的に設定する必要があります。

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

これらの構成パラメーターの正しい値を取得するために、DBA は HADR セットアップに含める 4 つのデータベースのホスト名、ポート番号、およびインスタンス名を次のように判別します。

表 9. データベースのホスト名、ポート番号、およびインスタンス名

対象となる役割	ホスト名	ポート番号	インスタンス名
1 次	host1	10	dbinst1
プリンシパル・スタンバイ	host2	40	dbinst2
補助スタンバイ	host3	41	dbinst3
補助スタンバイ	host4.ibm.com	42	dbinst4

1 次では、**hadr_remote_host**、**hadr_remote_inst**、および **hadr_remote_svc** 構成パラメーターの設定値がプリンシパル・スタンバイのホスト名、インスタンス名、およびポート番号にそれぞれ対応します。スタンバイでは、これらの構成パラメーターの値は、1 次のホスト名、ポート番号、およびインスタンス名にそれぞれ対応します。さらに、DBA はホスト名とポートの値を使用して、すべてのデータベースで **hadr_target_list** 構成パラメーターを設定します。また、必須ではありませんが、DBA はセットアップに含まれるすべてのスタンバイに関する情報を、他のスタンバイのそれぞれのターゲット・リストに追加します。このトピックについて詳しくは、41 ページの『高可用性災害時リカバリー用のデータベース構成 (HADR)』を参照してください。

前述のように、銀行は 1 次とプリンシパル・スタンバイの間の同期をできるだけ最密にする必要があるため、DBA は 1 次の **hadr_syncmode** パラメーターを SYNC に設定します。プリンシパル・スタンバイは、1 次への接続後、有効同期モードを自動的に SYNC に設定しますが、DBA は引き続き、プリンシパル・スタンバイで **hadr_syncmode** パラメーターを SYNC に設定します。理由は、プリンシパル・スタ

ンバイが 1 次と役割を交換する場合、新しい 1 次とプリンシパル・スタンバイのペアの同期モードも SYNC になるためです。

DBA は、host2 (補助スタンバイとは異なる都市にあります) を補助スタンバイのプリンシパル・スタンバイとして指定することを決定します。補助の 1 つが 1 次になる場合は、1 次とリモートで配置された host2 との間の同期モードは SUPERASYNC が適しています。したがって、DBA は補助スタンバイの **hadr_syncmode** パラメーターを SUPERASYNC に設定します。ただし、補助スタンバイでは 1 次への接続後、有効同期モードが自動的に SUPERASYNC に設定されます。このトピックについて詳しくは、 64 ページの『高可用性災害時リカバリー (HADR) 同期モード』を参照してください。

最後に、DBA は新しい HADR 遅延再生フィーチャーを確認しました。このフィーチャーを使用して、ログの再生を遅らせることにより、意図的にスタンバイ・データベースを 1 次より前の時点のものにしておくことができます。DBA は、このフィーチャーを使用可能にすることで、1 次の問題のあるトランザクションからの ExampleBANK のデータの保護が改善されると判断します。DBA は、このフィーチャーでは補助スタンバイとして host4 を選択し、このフィーチャーを使用不可にしてからでないと、host4 が 1 次データベースとしてテークオーバーできないことを注記します。このトピックについて詳しくは、 212 ページの『HADR 遅延再生』を参照してください。

DBA は以下のコマンドを発行して、各データベースの構成パラメーターを更新します。

- host1 (1 次) の場合:

```
DB2 "UPDATE DB CFG FOR hadr_db USING
     HADR_TARGET_LIST  host2:40|host3:41|host4:42
     HADR_REMOTE_HOST  host2
     HADR_REMOTE_SVC   40
     HADR_LOCAL_HOST   host1
     HADR_LOCAL_SVC    10
     HADR_SYNCMODE     sync
     HADR_REMOTE_INST  db2inst2"
```

- host2 (プリンシパル・スタンバイ) の場合:

```
DB2 "UPDATE DB CFG FOR hadr_db USING
     HADR_TARGET_LIST  host1:10|host3:41|host4:42
     HADR_REMOTE_HOST  host1
     HADR_REMOTE_SVC   10
     HADR_LOCAL_HOST   host2
     HADR_LOCAL_SVC    40
     HADR_SYNCMODE     sync
     HADR_REMOTE_INST  db2inst1"
```

- host3 (補助スタンバイ) の場合:

```
DB2 "UPDATE DB CFG FOR hadr_db USING
     HADR_TARGET_LIST  host2:40|host1:10|host4:42
     HADR_REMOTE_HOST  host1
     HADR_REMOTE_SVC   10
     HADR_LOCAL_HOST   host3
     HADR_LOCAL_SVC    41
     HADR_SYNCMODE     superasync
     HADR_REMOTE_INST  db2inst1"
```

- host4 (補助スタンバイ) の場合:

```
DB2 "UPDATE DB CFG FOR hadr_db USING
     HADR_TARGET_LIST  host2.:40|host1:10|host3:41
     HADR_REMOTE_HOST  host2
     HADR_REMOTE_SVC   10
     HADR_LOCAL_HOST   host4
     HADR_LOCAL_SVC    42
     HADR_SYNCMODE     superasync
     HADR_REMOTE_INST  db2inst1
     HADR_REPLAY_DELAY 86400"
```

最後に、ExampleBANK の DBA は、以下の理由で HADR スタンバイ・データベースの読み取りフィーチャーを使用可能にする必要があります。

- スタンバイの HADR 構成パラメーターのいくつかをオンラインで変更する。
- スタンバイで MON_GET_HADR 表関数を呼び出す。
- 1 次の読み取り専用ワークロードを再割り当てする。

DBA は、host2、host3、および host4 のそれぞれで以下のコマンドを発行して、スタンバイ・データベースのレジストリー変数を更新します。

```
DB2SET DB2_HADR_ROS=ON
DB2SET DB2_STANDBY_ISO=UR
```

HADR データベースの開始

DBA は、host2、host3、および host4 のそれぞれで以下のコマンドを発行して、最初にスタンバイ・データベースを開始します。

```
DB2 START HADR ON DB hadr_db AS STANDBY
```

次に、DBA は、以下のように host1 の 1 次データベースで HADR を開始します。

```
DB2 START HADR ON DB hadr_db AS PRIMARY
```

HADR が稼働していることを確認するために、DBA は、以下のように **db2pd** コマンド (すべてのスタンバイに関する情報を返します) を発行して、host1 における 1 次のデータベースの状況を照会します。

```
db2pd -db hadr_db -hadr

  Database Member 0 -- Database hadr_db -- Active -- Up 0 days 00:23:17 -- Date 06/08/2011 13:57:23

                             HADR_ROLE = PRIMARY
                           REPLAY_TYPE = PHYSICAL
                         HADR_SYNCMODE = SYNC
                            STANDBY_ID = 1
                         LOG_STREAM_ID = 0
                            HADR_STATE = PEER
                    PRIMARY_MEMBER_HOST = host1
                       PRIMARY_INSTANCE = db2inst1
                         PRIMARY_MEMBER = 0
                    STANDBY_MEMBER_HOST = host2
                       STANDBY_INSTANCE = db2inst2
                         STANDBY_MEMBER = 0
                    HADR_CONNECT_STATUS = CONNECTED
               HADR_CONNECT_STATUS_TIME = 06/08/2011 13:38:10.199479 (1307565490)
            HEARTBEAT_INTERVAL(seconds) = 30
                  HADR_TIMEOUT(seconds) = 120
          TIME_SINCE_LAST_RECV(seconds) = 3
               PEER_WAIT_LIMIT(seconds) = 0
             LOG_HADR_WAIT_CUR(seconds) = 0.000
      LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
    LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
```

```
                 LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 50772
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87616
            PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
            STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                 HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                   PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                   STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
             STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
         STANDBY_RECV_BUF_SIZE(pages) = 16
             STANDBY_RECV_BUF_PERCENT = 0
           STANDBY_SPOOL_LIMIT(pages) = 0
                PEER_WINDOW(seconds) = 0
            READS_ON_STANDBY_ENABLED = Y
    STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N

                          HADR_ROLE = PRIMARY
                      REPLAY_TYPE = PHYSICAL
                    HADR_SYNCMODE = SUPERASYNC
                       STANDBY_ID = 2
                    LOG_STREAM_ID = 0
                       HADR_STATE = REMOTE_CATCHUP
                PRIMARY_MEMBER_HOST = host1
                   PRIMARY_INSTANCE = db2inst1
                     PRIMARY_MEMBER = 0
                STANDBY_MEMBER_HOST = host3
                   STANDBY_INSTANCE = db2inst3
                     STANDBY_MEMBER = 0
                HADR_CONNECT_STATUS = CONNECTED
           HADR_CONNECT_STATUS_TIME = 06/08/2011 13:35:51.724447 (1307565351)
         HEARTBEAT_INTERVAL(seconds) = 30
               HADR_TIMEOUT(seconds) = 120
       TIME_SINCE_LAST_RECV(seconds) = 16
            PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
    LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
   LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                 LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
            PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
            STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                 HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                   PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                   STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
             STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
         STANDBY_RECV_BUF_SIZE(pages) = 16
             STANDBY_RECV_BUF_PERCENT = 0
           STANDBY_SPOOL_LIMIT(pages) = 0
                PEER_WINDOW(seconds) = 0
            READS_ON_STANDBY_ENABLED = Y
    STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N

                          HADR_ROLE = PRIMARY
                      REPLAY_TYPE = PHYSICAL
                    HADR_SYNCMODE = SUPERASYNC
                       STANDBY_ID = 3
                    LOG_STREAM_ID = 0
                       HADR_STATE = REMOTE_CATCHUP
                PRIMARY_MEMBER_HOST = host1
                   PRIMARY_INSTANCE = db2inst1
                     PRIMARY_MEMBER = 0
                STANDBY_MEMBER_HOST = host4
                   STANDBY_INSTANCE = db2inst4
                     STANDBY_MEMBER = 0
                HADR_CONNECT_STATUS = CONNECTED
```

```
            HADR_CONNECT_STATUS_TIME = 06/08/2011 13:46:51.561873 (1307566011)
          HEARTBEAT_INTERVAL(seconds) = 30
                HADR_TIMEOUT(seconds) = 120
        TIME_SINCE_LAST_RECV(seconds) = 6
             PEER_WAIT_LIMIT(seconds) = 0
           LOG_HADR_WAIT_CUR(seconds) = 0.000
    LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
   LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                  LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
            PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
            STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                  HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                    PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                    STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
              STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
         STANDBY_RECV_BUF_SIZE(pages) = 16
             STANDBY_RECV_BUF_PERCENT = 0
           STANDBY_SPOOL_LIMIT(pages) = 0
                PEER_WINDOW(seconds) = 0
            READS_ON_STANDBY_ENABLED = Y
    STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N
```

例: HADR 複数スタンバイ・モードでのテークオーバー

この HADR 複数スタンバイ・モードでの一連のテークオーバー (強制と非強制の両方) の例は、3 つのスタンバイ・セットアップに基づいています。これらの例は、複数スタンバイ自動再構成がテークオーバー状態でどのように機能するかを示すためのものです。

- 240 ページの『プリンシパル・スタンバイの正常なテークオーバー (役割切り替え)』
- 241 ページの『補助スタンバイの強制テークオーバー (フェイルオーバー)』
- 243 ページの『SA MP 環境での補助スタンバイの強制テークオーバー (フェイルオーバー)』

各例の初期セットアップは以下のとおりです。

- 1 次データベース (host1)
- プリンシパル・スタンバイ (host2)
- 2 つの補助スタンバイ (host3 および host4)

データベースはすべて hadr_db と呼ばれます。1 次とプリンシパル・スタンバイではその同期モードが SYNC に設定されており、補助スタンバイでは SUPERASYNC に設定されています。

各データベースの構成を表 10 に示します。

表 10. 各 HADR データベースの構成値

構成パラメーター	Host1	Host2	Host3	Host4
hadr_target_list	host2:40|host3:41| host4:42	host1:10|host3:41| host4:42	host2:40|host1:10| host4:42	host2:40|host1:10| host3:41
hadr_remote_host	host2	host1	host1	host1
hadr_remote_svc	40	10	10	10
hadr_remote_inst	dbinst2	dbinst1	dbinst1	dbinst1

表 10. 各 HADR データベースの構成値 (続き)

構成パラメーター	Host1	Host2	Host3	Host4
hadr_local_host	host1	host2	host3	host4
hadr_local_svc	10	40	41	42
構成済み **hadr_syncmode** (データベースが 1 次になる場合に使用される、明示的に設定された同期モードを指します)	SYNC	SYNC	SUPERASYNC	SUPERASYNC
有効な **hadr_syncmode** (データベースが現在スタンバイである場合に使用される同期モードを指します)	なし	SYNC	SUPERASYNC	SUPERASYNC

プリンシパル・スタンバイの正常なテークオーバー (役割切り替え)

DBA は、host2 で以下のコマンドを発行して、プリンシパル・スタンバイに対してテークオーバーを実行します。

```
DB2 TAKEOVER HADR ON DB hadr_db
```

テークオーバーが正常に完了すると、host2 は新しい 1 次になり、host2 の **hadr_target_list** の最初の項目である host1 (239 ページの表 10 を参照) はそのプリンシパル・スタンバイになります。それらの同期モードは、**hadr_syncmode** として SYNC を使用して host2 が構成されているため、SYNC モードになります。補助スタンバイのターゲットである host3 および host4 の **hadr_remote_host** および **hadr_remote_svc** は古い 1 次である host1 を指していますが、新しい 1 次である host2 に自動的にリダイレクトされます。このリダイレクトでは、host3 と host4 は (永続的に) それぞれの **hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** 構成パラメーターを更新します。これらは補助スタンバイとして host2 に再接続し、(ローカルで **hadr_syncmode** に構成された内容に関係なく) 有効同期モードとして SUPERASYNC を使用するように host2 によって指示されます。**hadr_syncmode** の設定は永続的には更新されません。各データベースの構成を表 11 に示します。

表 11. 役割の切り替え後の各 HADR データベースの構成値： 列 4 と 5 の行 3 から 5 までが太字になっており、自動再構成されたことを示しています。

構成パラメーター	Host1	Host2	Host3	Host4
hadr_target_list	host2:40|host3:41| host4:42	host1:10|host3:41| host4:42	host2:40|host1:10| host4:42	host2:40|host1:10| host3:41
hadr_remote_host	host2	host1	**host2**	**host2**
hadr_remote_svc	40	10	**40**	**40**
hadr_remote_inst	dbinst2	dbinst1	**dbinst2**	**dbinst2**
hadr_local_host	host1	host2	host3	host4
hadr_local_svc	10	40	41	42

表 11. 役割の切り替え後の各 HADR データベースの構成値 (続き): 列 4 と 5 の行 3 から 5 までが太字になっており、自動再構成されたことを示しています。

構成パラメーター	Host1	Host2	Host3	Host4
構成済み **hadr_syncmode**	SYNC	SYNC	SUPERASYNC	SUPERASYNC
有効な **hadr_syncmode**	SYNC	なし	SUPERASYNC	SUPERASYNC

注: いくつかの値は以下の理由により更新されません。

- host2 の **hadr_remote_host** および **hadr_remote_svc** 構成パラメーターが既にそのプリンシパル・スタンバイである host1 を指しているため、これらの値は host2 で更新されません。
- host1 の **hadr_remote_host** および **hadr_remote_svc** 構成パラメーターが既に新しい 1 次を指しているため、これらの値は host1 で更新されません。
- host1 の運用同期モードが SYNC であり、host3 と host4 の各運用同期モードが SUPERASYNC であるため、有効同期モードは変更されません。

補助スタンバイの強制テークオーバー (フェイルオーバー)

都市 A の広範囲で停電が発生した場合、1 次 (host1) は使用不可になります。通常は、SYNC モードのプリンシパル・スタンバイ (host2) がテークオーバーを行って新しい 1 次になる対象としては最適ですが、停電が発生すると host2 も一時的に使用不可になります。DBA は、次のように、2 つの補助スタンバイを照会して、最も多くのログ・データを含む補助スタンバイを判別します。

```
db2pd -hadr -db hadr_db | grep 'PRIMARY_LOG_FILE,PAGE,POS|STANDBY_LOG_FILE,PAGE,POS'
```

DBA は host3 が最新 (ただし、ログ再生は依然として少し遅れる) であると判断し、そのホストを新しい 1 次として選出します。

```
DB2 TAKEOVER HADR ON DB hadr_db BY FORCE
```

テークオーバーが正常に完了すると、host3 は新しい 1 次になります。一方、host2 は再び使用可能になります。host3 は 1 次になったことを host2 と host4 に知らせます。host3 では、**hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** の各値が host2 (host3 の **hadr_target_list** の最初の項目であるため、このホストがプリンシパル・スタンバイになります) を指すように再構成されます。host2 では、同期モードが SUPERASYNC (host3 の **hadr_syncmode** に設定されているため) になるように再構成され、さらに、**hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** が (永続的に) 更新されます。 host4 は新しい 1 次である host3 に自動的にリダイレクトされます。このリダイレクトでは、host4 は (永続的に) その **hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** の各構成パラメーターを更新します。 host1 は再度使用可能になるまで、自動的に再構成されません。各データベースの構成を表 12 に示します。

表 12. フェイルオーバー後の各 HADR データベースの構成値: 列 3 から 5 までの行 3 から 5 までが太字になっており、自動再構成されたことを示しています。

構成パラメーター	Host1 (使用不可)	Host2	Host3	Host4
hadr_target_list	host2:40|host3:41| host4:42	host1:10|host3:41| host4:42	host2:40|host1:10| host4:42	host2:40|host1:10| host3:41

表 12. フェイルオーバー後の各 HADR データベースの構成値 (続き): 列 3 から 5 までの行 3 から 5 までが太字になっており、自動再構成されたことを示しています。

構成パラメーター	Host1 (使用不可)	Host2	Host3	Host4
hadr_remote_host	host2	**host3**	**host2**	**host3**
hadr_remote_svc	40	**41**	**40**	**41**
hadr_remote_inst	dbinst2	**dbinst3**	**dbinst2**	**dbinst3**
hadr_local_host	host1	host2	host3	host4
hadr_local_svc	10	40	41	42
構成済み **hadr_syncmode**	SYNC	SYNC	SUPERASYNC	SUPERASYNC
有効な **hadr_syncmode**	なし	**SUPERASYNC**	なし	SUPERASYNC

host1 はしばらく経ってから使用可能になります。 DBA は host1 をスタンバイとして開始しようとしますが、host1 には host3 に伝搬されたログより多くのログがあるため、host1 は新しい 1 次との最初のハンドシェークの一工程で拒否されます。DBA は、以下のように新しい 1 次のバックアップを取り、host1 にリストアしてから HADR をそのホストで開始します。

```
DB2 BACKUP DB hadr_db

DB2 RESTORE DB hadr_db

DB2 START HADR ON DB hadr_db AS STANDBY
```

表 13 を見るとわかるように、host1 は再構成されます。

表 13. 再統合されたスタンバイの構成値: 列 2 のいくつかの行が太字になっており、自動再構成されたことを示しています。

構成パラメーター	Host1	Host2	Host3	Host4
hadr_target_list	host2:40|host3:41| host4:42	host1:10|host3:41| host4:42	host2:40|host1:10| host4:42	host2:40|host1:10| host3:41
hadr_remote_host	**host3**	host3	host2	host3
hadr_remote_svc	**41**	41	40	41
hadr_remote_inst	**dbinst3**	dbinst3	dbinst2	dbinst3
hadr_local_host	host1	host2	host3	host4
hadr_local_svc	10	40	41	42
構成済み **hadr_syncmode**	SYNC	SYNC	SUPERASYNC	SUPERASYNC
有効な **hadr_syncmode**	**SUPERASYNC**	SUPERASYNC	なし	SUPERASYNC

DBA が host1 を再度 1 次にする場合に必要なのはフェイルバックのみです。これにより、 239 ページの表 10 に示されている元の構成がリストアされます。

SA MP 環境での補助スタンバイの強制テークオーバー (フェイルオーバー)

この例は前の例と似ていますが、HADR は、フェイルオーバーを自動化するための IBM Tivoli System Automation for Multiplatforms (SA MP) とともにデプロイされています。

都市 A で停電が発生した場合、プリンシパル・スタンバイ (host2) は使用不可になります。その後、1 次 (host1) は停止します。通常は、SA MP (クラスター・マネージャー) が自動的にプリンシパル・スタンバイ (host2) へのフェイルオーバーを行いますが、停電が発生した場合は補助スタンバイの 1 つをテークオーバーのターゲットにする必要があります。補助スタンバイへのフェイルオーバーは自動化できないため、DBA が手動で行う必要があります。ただし、これを行う前に、host1 または host2 が使用可能になった場合に*分離脳* 状態 (複数のデータベースが 1 次として独立して動作する状態) が発生することのないように、DBA は TSA が使用不可になっていることを確認する必要があります。そのため、DBA は host1 および host2 が使用可能になる場合は必ず、これらのホストで次のコマンドを発行します。

```
db2haicu -disable
```

さらに、DBA は、host1 をオフライン状態に保ち、クライアントが古い 1 次に接続した際に、その古い 1 次が再始動しないようにする必要があります。

DBA は、次のように、2 つの補助スタンバイを照会して、最も多くのログ・データを含む補助スタンバイを判別します。

```
db2pd -hadr -db hadr_db | grep 'STANDBY_LOG_FILE,PAGE,POS'
```

DBA は host3 が最新であると判断し、そのホストを新しい 1 次として選択します。

次に、DBA は以下のように host3 で強制テークオーバーを実行します。

```
DB2 TAKEOVER HADR ON DB hadr_db BY FORCE
```

テークオーバーが正常に完了すると、host3 は新しい 1 次になります。一方、host2 は再び使用可能になります。host3 は 1 次になったことを host2 と host4 に知らせます。host3 では、**hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** の各値が host2 (host3 の **hadr_target_list** の最初の項目であるため、このホストがプリンシパル・スタンバイになります) を指すように再構成されます。host2 では、同期モードが SUPERASYNC (host3 の **hadr_syncmode** に設定されているため) になるように再構成され、さらに、**hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** が (永続的に) 更新されます。 host4 は新しい 1 次である host3 に自動的にリダイレクトされます。このリダイレクトでは、host4 は (永続的に) その **hadr_remote_host**、**hadr_remote_svc**、および **hadr_remote_inst** の各構成パラメーターを更新します。 host1 では自動再構成は行われません。各データベースの構成を 244 ページの表 14 に示します。

表 14. フェイルオーバー後の各 HADR データベースの構成値： 列 3 から 5 までの行 3 から 5 までが太字になっており、自動再構成されたことを示しています。

構成パラメーター	Host1 (使用不可)	Host2	Host3	Host4
hadr_target_list	host2:40|host3:41| host4:42	host1:10|host3:41| host4:42	host2:40|host1:10| host4:42	host2:40|host1:10| host3:41
hadr_remote_host	host2	**host3**	**host2**	**host3**
hadr_remote_svc	40	**41**	**40**	**41**
hadr_remote_inst	dbinst2	**dbinst3**	**dbinst2**	**dbinst3**
hadr_local_host	host1	host2	host3	host4
hadr_local_svc	10	40	41	42
構成済み **hadr_syncmode**	SYNC	SYNC	SUPERASYNC	SUPERASYNC
有効な **hadr_syncmode**	なし	**SUPERASYNC**	なし	SUPERASYNC

HADR スタンバイ・データベースの読み取りフィーチャー

スタンバイ・データベースの読み取り機能を使用して、高可用性災害時リカバリー (HADR)・ソリューションのスタンバイ・データベース上で、読み取り専用操作を実行できます。スタンバイ・データベースで実行される読み取り操作は、スタンバイ・データベースの主な役割 (1 次データベースから送られるログの適用) には影響を与えません。

スタンバイ・データベースの読み取りフィーチャーによって、HADR セットアップの総所有コストを削減できます。スタンバイ・データベースのこの拡張された役割によって、 1 次データベースで実行する予定であったワークロードをスタンバイ・データベースで実行するなど、スタンバイ・データベースを新しい方法で使用できます。つまり、1 次データベースの余分なワークロードを解消できます。

読み取りおよび書き込みクライアントは、引き続き 1 次データベースに接続します。ただし、読み取りクライアントは、ローカル・キャッチアップ状態にも適用専用時間枠にもない限り、読み取り可能なスタンバイ・データベース、つまりアクティブ・スタンバイ・データベース にも接続できます。アクティブ・スタンバイ・データベースの主な役割は、1 次データベースから送られたログの再生であることに変わりはありません。その結果、スタンバイ・データベースのデータは、1 次データベースのデータと実際に同じものになります。フェイルオーバーの際に、スタンバイ・データベースが新しい 1 次データベースとしてテークオーバーする際は、スタンバイ・データベースに接続していたユーザーは切断されます。

両方向スクロール・カーソルおよびスクロール不能カーソルを含むすべてのタイプの読み取り照会が、スタンバイ・データベースでサポートされます。読み取り機能は、 HADR の 3 つの同期モード (SYNC、NEARSYNC、および ASYNC) のすべてにおいて、ローカル・キャッチアップを除くすべての HADR 状態でサポートされます。

スタンバイ・データベースの読み取りの使用可能化

高可用性災害時リカバリー (HADR) スタンバイ・データベースの読み取りフィーチャーは、 **DB2_HADR_ROS** レジストリー変数を使用して有効にします。

始める前に

データベース構成パラメーター **logindexbuild** を ON に設定することをお勧めします。これによって、無効な索引を使用しない照会アクセス・プランが、パフォーマンスに影響を与えることを防止します。

また、スタンバイ・データベースの読み取りを使用可能にする場合、仮想 IP の使用をお勧めします。クライアント・リルートは、書き込み可能データベース (1 次および標準データベース) と読み取り専用データベース (スタンバイ・データベース) を区別しません。 1 次データベースとスタンバイ・データベースの間にクライアント・リルートを構成すると、本来の実行場所ではないデータベースにアプリケーションをリルートしてしまう可能性があります。

手順

1. **DB2_HADR_ROS** レジストリー変数を ON にします。
2. HADR 用の 1 次データベースおよびスタンバイ・データベースをセットアップして初期設定します。 36 ページの『高可用性災害時リカバリーの初期設定 (HADR)』 を参照してください。

タスクの結果

以上で、スタンバイ・データベースはアクティブ・スタンバイ と見なされ、読み取り専用ワークロードを受け入れることが可能になります。

次のタスク

こうして、スタンバイ・データベースを要件に合わせて利用できるようになり、例えば読み取り専用ワークロードの一部をスタンバイ・データベースで実行するように構成することができます。

アプリケーションがスタンバイ・データベースにアクセスできるようにしておくには、ホワイト・ペーパー「Continuous access to Read on Standby databases using Virtual IP addresses」に記載されているステップに従ってください。

アクティブ・スタンバイ・データベースのデータの並行性

HADR 1 次データベースの変更は、必ず HADR アクティブ・スタンバイ・データベースに反映されるわけではありません。1 次データベースでコミットされていない変更は、 1 次データベースがそのログをディスクにフラッシュまたは出力するまで、スタンバイ・データベースには複製されない可能性があります。

ログは、必ずディスクにフラッシュされます。それゆえ、コミットされた後、ログはスタンバイ・データベースに送信されます。ログのフラッシュは、ログ・バッファー・フルのような不確定な状態によってもトリガーされます。その結果として、1 次データベースでコミットされていない変更が、1 次データベースのログ・バッフ

ァー上に長い間残る可能性があります。ロガーは部分ページのフラッシュを避けるので、このような状態は、特に 1 次データベース上でコミットされていない小さい変更に影響を与える可能性があります。

スタンバイ・データベースで実行するワークロードが、1 次データベースと実際に同一のデータを必要とする場合は、トランザクションをより頻繁にコミットすることを検討する必要があります。

アクティブ・スタンバイ・データベースの分離レベル

アクティブ・スタンバイ・データベース (読み取り可能な HADR スタンバイ・データベース) でサポートされる唯一の分離レベルは、非コミット読み取り (UR) です。アプリケーション、ステートメント、またはサブステートメントの要求する分離レベルが UR よりも高い場合、エラーが戻されます (SQL1773N 理由コード 1)。

UR 以外の分離レベルが必要な場合は、スタンバイ・データベースではなく HADR 1 次データベースをアプリケーションに使用するように検討してください。単にこのメッセージを受け取らないようにするには、**DB2_STANDBY_ISO** レジストリー変数を UR に設定します。**DB2_STANDBY_ISO** を UR に設定すると、分離レベルは自動的に UR に強制変更されます。この設定は、他のすべての分離設定 (例えば、ステートメント分離およびパッケージ分離) よりも優先されます。

アクティブ・スタンバイ・データベースの適用専用時間枠

HADR アクティブ・スタンバイ・データベースが DDL ログ・レコードまたは保守操作を適用している間、スタンバイ・データベースは*適用専用時間枠*に入ります。スタンバイ・データベースが適用専用時間枠にあると、スタンバイ・データベースへの既存の接続は強制終了され、新規の接続はブロックされます (SQL1776N 理由コード 4)。アクティブな DDL または保守操作の適用がすべて完了すると、スタンバイ・データベースへの新規接続が許可されます。

適用専用時間枠にあるスタンバイ・データベース上にアクティブなまま保持されるユーザー接続は、**DEACTIVATE DATABASE** または **TAKEOVER** コマンドを実行している接続だけです。適用専用時間枠に入ると、アプリケーションは強制終了され、エラーが戻されます (SQL1224N)。アクティブ・スタンバイ・データベースに接続していた読み取りプログラムの数によっては、 DDL ログ・レコードまたは保守操作のスタンバイ・データベースへの適用に、若干の遅延が生じる可能性があります。

HADR 1 次データベースで実行すると、スタンバイ・データベースの適用専用時間枠をトリガーする、いくつかの DDL ステートメントおよび保守操作があります。以下は、すべてをリストしたものではありません。

DDL ステートメント

- CREATE、ALTER、または DROP TABLE (DGTT に対する DROP TABLE は除く)
- CREATE GLOBAL TEMP TABLE
- TRUNCATE TABLE
- RENAME TABLE
- RENAME TABLESPACE
- CREATE、DROP、または ALTER INDEX

- CREATE または DROP VIEW
- CREATE、ALTER、または DROP TABLESPACE
- CREATE、ALTER、または DROP BUFFER POOL
- CREATE、ALTER、または DROP FUNCTION
- CREATE、ALTER、または DROP PROCEDURE
- CREATE または DROP TRIGGER
- CREATE、ALTER、または DROP TYPE
- CREATE、ALTER、または DROP ALIAS
- CREATE または DROP SCHEMA
- CREATE、ALTER、または DROP METHOD
- CREATE、ALTER、または DROP MODULE
- CREATE、ALTER、または DROP NICKNAME
- CREATE、ALTER、または DROP SEQUENCE
- CREATE、ALTER、または DROP WRAPPER
- CREATE、ALTER、または DROP FUNCTION MAPPING
- CREATE または DROP INDEX EXTENSION
- CREATE または DROP INDEX FOR TEXT
- CREATE または DROP EVENT MONITOR
- CREATE、ALTER、または DROP SECURITY LABEL
- CREATE、ALTER、または DROP SECURITY LABEL COMPONENT
- CREATE、ALTER、または DROP SECURITY POLICY
- CREATE または DROP TRANSFORM
- CREATE、ALTER、または DROP TYPE MAPPING
- CREATE、ALTER、または DROP USER MAPPING
- CREATE または DROP VARIABLE
- CREATE、ALTER、または DROP WORKLOAD
- GRANT USAGE ON WORKLOAD
- REVOKE USAGE ON WORKLOAD
- CREATE、ALTER、または DROP SERVICE CLASS
- CREATE、ALTER、または DROP WORK CLASS SET
- CREATE、ALTER、または DROP WORK ACTION SET
- CREATE、ALTER、または DROP THRESHOLD
- CREATE、ALTER、または DROP HISTOGRAM TEMPLATE
- AUDIT
- CREATE、ALTER、または DROP AUDIT POLICY
- CREATE または DROP ROLE
- CREATE、ALTER、または DROP TRUSTED CONTEXT
- REFRESH TABLE
- SET INTEGRITY

保守操作

- 従来の、つまりオフライン再編成
- インプレースまたはオンライン再編成
- 索引再編成 (すべての索引、個別索引)
- MDC および ITC の再利用のための再編成
- ロード
- バインドまたは再バインド
- db2rbind
- Runstats
- 表移動
- 自動統計
- 自動再編成
- リアルタイム統計

その他の操作またはアクション

- COMPRESS YES 属性を持つ表のディクショナリー自動作成
- デタッチされた表パーティションの非同期索引クリーンアップ
- 暗黙的な再バインド
- 暗黙的な索引再作成
- 統計の手動更新
- 据え置き MDC ロールアウト
- MDC ロールアウト後の非同期索引クリーンアップ
- MDC または ITC 表への INSERT 時に削除済みの MDC または ITC ブロックを再利用
- タスクの挿入、更新、および削除に関してカタログ表 SYSJOBS および SYSTASKS を更新する非同期のバックグラウンド・プロセス

適用専用時間枠のモニター

アクティブ・スタンバイ・データベースの適用専用時間枠をモニターするには、**-hadr** オプションを指定して **db2pd** コマンドを使用します。次の例では、以下の 3 つの関連するエレメントがあります。

- **ReplayOnlyWindowStatus**: スタンバイ・データベースで、DDL または保守操作の適用が進行中かどうかを示します。通常この値は「Inactive」ですが、適用専用時間枠がアクティブな場合は、「Active」です。
- **ReplayWindowStartTime**: 現行の適用専用時間枠 (ある場合) がアクティブになった時刻を示します。
- **MaintenanceTxCount** または **DDLTxCount**: 現行の適用専用時間枠 (ある場合) で、これまでに実行されたコミットされていない既存の DDL ステートメントまたは保守トランザクションの総数。

```
db2pd -db hadrdb -hadr
Database Partition 0 -- Database HADRDB -- Active -- Up 0 days 00:00:06

HADR Information:
Role    State  SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
```

```
Standby Peer  Nearsync 0                  0

ConnectStatus ConnectTime                            Timeout
Connected     Sat Jun 15 03:09:35 2008          120

ReplayOnlyWindowStatus       ReplayOnlyWindowStartTime          MaintenanceTxCount
Active                       Sun Jun 16 08:09:35 2008           5

LocalHost  LocalService
skua       52601

RemoteHost  RemoteService  RemoteInstance
gull        52600          vinci

PrimaryFile  PrimaryPg  PrimaryLSN
S0000000.LOG 1          0x000000000137126F

StandByFile  StandByPg  StandByLSN
S0000000.LOG 0          0x000000000137092E
```

適用専用時間枠の影響を最小化するための推奨事項

HADR スタンバイ・データベースの適用操作は、読み取りプログラムよりも優先順位が高いため、読み取り専用時間枠が頻繁に発生すると、スタンバイ・データベースに接続していた、または接続しようとする読み取りプログラムは、何度も中断される可能性があります。この影響を回避または最小化するには、以下の推奨事項を検討してください。

- DDL および保守操作は、定期保守時間枠内 (オフピーク時が望ましい) に実行する。
- DDL 操作は、複数のグループに分けて実行するのではなく、集中的に実行する。
- **REORG** または **RUNSTATS** は、すべての表にではなく、必要な表に対してのみ実行する。
- 1 次データベースで DDL または保守操作を実行する前に、**ALL** オプションを指定して **FORCE APPLICATION** コマンドを使用することにより、アクティブ・スタンバイ・データベースのアプリケーションを強制終了する。適用専用時間枠をモニターして、時間枠が非アクティブであるのを確認してから、アプリケーションをスタンバイ・データベースに再配備する。

アクティブ・スタンバイ・データベースの読み取りアプリケーションの一時的な強制終了

HADR アクティブ・スタンバイ・データベースは読み取り専用ワークロードの実行に使用できますが、主な役割は、HADR 1 次データベースとの同期を維持するためにログ・レコードを適用して、 1 次データベースの役割を引き継がなければいけない場合に備えることです。読み取り専用ワークロードが原因で、スタンバイ・データベースのログの適用が遅延した場合は、キャッチアップできるように、スタンバイ・データベースへのすべての接続を一時的に強制終了することがあります。

このタスクについて

一時的に、アクティブ・スタンバイを読み取りプログラムからアクセス不能にするには、以下の手順を使用します。

手順

1. **FORCE APPLICATION** コマンドを実行します。これによって、スタンバイ・データベースへの既存の接続が強制終了されます。
2. 仮想 IP の構成を変更します。これによって、スタンバイ・データベースへの新規接続を防止します。

次のタスク

スタンバイ・データベースがログを適用して 1 次データベースにキャッチアップしたら、仮想 IP の構成を元の設定に戻して、アクティブ・スタンバイ・データベースへの接続を再開できるようにします。

スタンバイ・データベースの読み取りに関する制約事項

高可用性災害時リカバリー (HADR) スタンバイ・データベースの読み取りフィーチャーを使用すると、読み取り専用ワークロードを HADR アクティブ・スタンバイ・データベースで実行できます。読み取り専用という制約事項の他にも、この機能には注意すべき以下の制約事項があります。

- スタンバイ・データベースでの書き込み操作はサポートされていません。この場合の書き込み操作とは、カタログ、表、および索引のような永続的なデータベース・オブジェクトを変更する操作です。スタンバイ・データベースで書き込み操作を実行すると、エラー (SQL1773N 理由コード 5) が戻されます。具体的には、ログ・レコードを生成することになる操作は、スタンバイ・データベースで実行できません。
- DDL ログ・レコードまたは保守操作の適用中 (*適用専用時間枠*) は、スタンバイ・データベースにユーザー接続によってアクセスすることはできません。詳しくは、 246 ページの『アクティブ・スタンバイ・データベースの適用専用時間枠』を参照してください。
- スタンバイ・データベースがローカル・キャッチアップ状態である間は、スタンバイ・データベースにユーザー接続によってアクセスすることはできません。この状態の時に接続しようとしたクライアントは、エラー (SQL1776N 理由コード 1) を受け取ります。
- スタンバイ・データベースでは、非コミット読み取り (UR) の分離レベルのみサポートされます。より高い分離レベルを要求するアプリケーション、ステートメント、またはサブステートメントは、エラー (SQL1773N 理由コード 1) を受け取ります。詳しくは、 246 ページの『アクティブ・スタンバイ・データベースの分離レベル』を参照してください。
- インスタンス・レベルの監査の構成は、スタンバイ・データベースに複製されません。 db2audit ツールを使用して、 1 次データベースとスタンバイ・データベースで、インスタンス・レベルの監査設定を等しくする必要があります。
- 宣言済み一時表 (DGTT) は、スタンバイ・データベースではサポートされません。スタンバイ・データベース上でこれらを作成またはアクセスしようとすると、エラー (SQL1773N 理由コード 4) を受け取ります。
- 作成済み一時表 (CGTT) は 1 次データベースでのみ作成可能であり、その定義がスタンバイ・データベースに複製されます。ただし、スタンバイ・データベースでは CGTT へのアクセスがサポートされません。これらを作成またはアクセスしようとすると、エラー (SQL1773N 理由コード 4) を受け取ります。

- 作成済みの一時表 (CGTT) を 1 次データベースで作成すると、スタンバイ・データベースで適用専用時間枠がトリガーされます。
- NOT LOGGED INITIALLY (NLI) 表は、スタンバイ・データベースではアクセスできません。スタンバイ・データベースの NLI 表を読み取ろうとしたアプリケーションは、エラー (SQL1477N) を受け取ります。
- スタンバイ・データベースの照会では、SMS SYSTEM TEMPORARY 表スペースだけを使用できます。 DMS SYSTEM TEMPORARY 表スペースを使用するスタンバイ・データベース照会を実行した場合、結果としてエラー (SQL1773N 理由コード 5) が発生することがあります。
- XML およびラージ・オブジェクト (LOB) データは、正常に照会するためにはインラインでなければなりません。そうしないと、エラーが戻されます (SQL1773N 理由コード 3)。
- ロング・フィールド (LF)、これらのいずれかのデータ・タイプに基づく特殊タイプ、および構造化タイプの列のデータは、照会できません。これらのデータ・タイプを照会しようとすると、エラー (SQL1773N 理由コード 3) を受け取ります。
- Explain ツール (**db2exfmt**と **db2expln**) と **db2batch** ツールは、スタンバイ・データベースではサポートされません (SQL1773N 理由コード 5)。読み取り専用ワークロードのパフォーマンスを分析する場合は、まず 1 次データベースでこれらのツールを実行し、 1 次データベースのワークロードに必要な最適化を行ってから、その最適化したワークロードをスタンバイ・データベースに移動します。
- パッケージの明示的なバインドと再バインド、および暗黙的な再バインドは、スタンバイ・データベースではサポートされません。無効なオブジェクトを参照する静的なパッケージを実行、またはこのようなパッケージを暗黙的に再バインドすると、エラー (それぞれ SQL1773N の理由コード 5 と 6) となります。これを回避するには、1 次データベースにパッケージをバインドし、その変更がスタンバイ・データベースに複製された後に、スタンバイ・データベースでパッケージを実行します。
- セルフチューニング・メモリー・マネージャー (STMM) は、スタンバイ・データベースではサポートされません。スタンバイ・データベースをチューニングする場合 (読み取り専用ワークロードの実行調整、またはテークオーバー後のパフォーマンスの改善が目的) は、手動で行う必要があります。
- 1 次データベースに対するワークロード・マネージャー (WLM) の DDL ステートメントは、スタンバイ・データベースに適用されますが、スタンバイ・データベースでは使用可能になりません。ただし、スタンバイ・データベースのセットアップに使用したデータベース・バックアップ内に含まれていた定義はすべて、読み取り可能スタンバイ・データベースでアクティブになります。
- シーケンスの作成および変更は、スタンバイ・データベースではサポートされません。また、シーケンスの次の値を生成する NEXT VALUE 式も使用できません。
- 無効なオブジェクトに関する実行時の再妥当性検査は、スタンバイ・データベースではサポートされません。
- スタンバイ・データベースをフェデレーション・サーバーとして構成することはできません。

- スタンバイ・データベースでのバックアップおよびアーカイブ操作はサポートされていません。
- スタンバイ・データベースでの静止操作はサポートされていません。

高可用性ソリューションにおけるシステム停止の検出と応答

高可用性ソリューションをインプリメントしても、ハードウェアまたはソフトウェアで障害が起きないわけではありません。ただし、予備システムおよびフェイルオーバー・メカニズムを整えておくと、そのソリューションによって、障害を検出して応答し、ワークロードを転送してユーザー・アプリケーションが引き続き処理を行えるようにできます。

手順

障害が生じると、データベース・ソリューションによって以下を行う必要があります。

1. 障害を検出する。

   フェイルオーバー・ソフトウェアは、ハートビート・モニターを使用して、システム・コンポーネントが稼働していることを確認することができます。ハートビート・モニターは、システムのすべてのコンポーネントから通常の通信を listen します。ハートビート・モニターがコンポーネントからの聴取を停止すると、ハートビート・モニターはそのコンポーネントで障害が発生したことをシステムにシグナル通知します。
2. 障害に応答する。フェイルオーバー
   a. 障害の発生したコンポーネントの操作をテークオーバーする 2 次コンポーネントを識別し、オンラインにして、初期化します。
   b. その 2 次コンポーネントにワークロードを転送します。
   c. 障害の発生したコンポーネントをシステムから除去します。
3. 障害からリカバリーする。

   1 次データベース・サーバーが失敗する際の最優先事項は、クライアントを代替サーバーにリダイレクトするか、スタンバイ・データベースにフェイルオーバーして、クライアント・アプリケーションが可能な限り中断することなく作業を行えるようにすることです。フェイルオーバーが成功したなら、障害の発生したデータベース・サーバーでの故障箇所を修理して、そのソリューションに再び組み込むことができるようにする必要があります。障害の発生したデータベース・サーバーを修理するとは、つまりそれを再始動できるようにするという意味です。
4. 正常操作に戻る。

   障害の発生したデータベース・システムを修理したなら、再びそれをデータベース・ソリューションに組み込む必要があります。障害の発生した際に 1 次データベースとしてテークオーバーしたデータベースに対して、障害の発生した 1 次データベースをスタンバイ・データベースとして再統合できます。また、修理したデータベース・サーバーを、再び 1 次データベース・サーバーとしてテークオーバーするようにもできます。

次のタスク

DB2 データベースは、こうした一部のステップを実行できます。以下に例を示します。

- DB2 高可用性災害時リカバリー (HADR) ハートビート・モニター・エレメント **hadr_heartbeat** は、1 次データベースで発生した障害を検出できます。
- DB2 クライアント・リルートは、障害の発生したデータベース・サーバーから 2 次データベース・サーバーにワークロードを転送できます。
- DB2 障害モニターは、予期せずに終了したデータベース・インスタンスを再始動できます。

管理通知ログ

管理通知ログ (*instance_name*.nfy) とは、多数のデータベース管理および保守アクティビティーについての情報が取得できるリポジトリーのことです。データベース管理者はこの情報を使用することにより、問題の診断やデータベースの調整を行ったり、単にデータベースのモニターを行ったりすることもできます。

DB2 データベース・マネージャーは、UNIX および Linux オペレーティング・システム・プラットフォーム上の管理通知ログに以下のような情報を書き込みます (Windows オペレーティング・システム・プラットフォームでは、イベント・ログが管理通知イベントの記録に使用されます)。

- DB2 ユーティリティーの状況 (**REORG** および **BACKUP** など)
- クライアント・アプリケーション・エラー
- サービス・クラスの変更
- ライセンス交付アクティビティー
- ファイル・パス
- ストレージの問題
- モニター・アクティビティー
- 索引付けアクティビティー
- 表スペースの問題

管理通知ログ・メッセージは、標準化されたメッセージ・フォーマットを使用して **db2diag** ログ・ファイルにも記録されます。

通知メッセージには、提供されている SQLCODE を補足する追加情報が備えられています。

管理通知ログ・ファイルは、以下の 2 つの異なる形式で存在できます。

単一の管理通知ログ・ファイル
1 つのアクティブ管理通知ログ・ファイル。名前は *instance_name*.nfy で、サイズは無制限に大きくなります。これがデフォルト形式で、このファイルは **diagsize** データベース・マネージャー構成パラメーターの値が 0 (このパラメーターのデフォルト値は 0) であるときにはいつでも存在しています。

循環管理通知ログ・ファイル
単一のアクティブ・ログ・ファイル (*instance_name.N*.nfy という名前、ここ

で *N* は 0 から始まる連続して増える数であるファイル名索引)。ただし、**diagpath** 構成パラメーターが定義する場所に一連の管理通知ログ・ファイルがあり、それぞれは限度のサイズに達するまで大きくなります。限度に達したとき、ログ・ファイルは閉じられ、1 つ大きいファイル名索引を使って (*instance_name*.*N+1*.nfy) 新しいログ・ファイルが作成され、ロギング用に開かれます。このログ・ファイルは、**diagsize** データベース・マネージャー構成パラメーターの値がゼロ以外であるときはいつでも、存在しています。

注: Windows オペレーティング・システム・プラットフォームでは、単一の管理通知ログ・ファイルと循環管理通知ログ・ファイルのどちらも使用できません。

diagsize データベース・マネージャー構成パラメーターを適切に設定することにより、これら 2 つの形式のどちらがご使用のシステムに存在するかを選択することができます。

構成

以下のデータベース・マネージャー構成パラメーターを設定することにより、管理通知ログ・ファイルのサイズ、場所、および記録されるイベントのタイプと詳細のレベルを構成できます。

diagsize

diagsize の値は、どの形式の管理通知ログ・ファイルが採用されるかを決定します。値が 0 の場合、単一の管理通知ログ・ファイルが採用されます。値が 0 でない場合、循環管理通知ログ・ファイルが採用され、このゼロ以外の値はすべての循環診断ログ・ファイルとすべての循環管理通知ログ・ファイルの合計サイズも指定します。**diagsize** パラメーターの新規値を有効にするには、インスタンスを再始動する必要があります。詳細については、『diagsize - 診断ログ・ファイル・サイズ構成パラメーター』のトピックを参照してください。

diagpath

診断情報は、**diagpath** 構成パラメーターによって定義される場所にある管理通知ログ・ファイルに、書き込まれるように指定できます。詳細については、『diagpath - 診断データ・ディレクトリー・パス構成パラメーター』のトピックを参照してください。

notifylevel

管理通知ログ・ファイルに書き込まれるイベントのタイプおよび詳細のレベルは、**notifylevel** 構成パラメーターで指定できます。詳細については、『notifylevel - 通知レベル構成パラメーター』のトピックを参照してください。

注: diagsize 構成パラメーターがゼロ以外の値に設定される場合、その値によって指定されるのは、すべての循環管理通知ログ・ファイルと、診断データ・ディレクトリー内に含まれるすべての循環診断ログ・ファイルとを組み合わせた合計サイズです。例えば、4 つのデータベース・パーティションがあるシステムで **diagsize** が 1 GB に設定されている場合は、通知ログと診断ログを組み合わせた最大の合計サイズは 4 GB (4 x 1 GB) まで許容されます。

計画外の停止の検出

コンポーネントの障害に対応するにはまず、コンポーネントで障害が発生したことを検出しなければなりません。 DB2 Data Server にはデータベースの正常性をモニターするため、またはデータベースの障害を検出するためのツールがいくつか備えられています。それらのツールは、障害を検出したときに通知したり事前定義処置を取ったりするように構成できます。

手順

DB2 データベース・ソリューションの一部で障害が発生したときは、次のツールを使用して検出することができます。

DB2 障害モニター機能

DB2 障害モニター機能は DB2 データベース・インスタンスを常に稼働させておきます。 DB2 障害モニターが接続されている DB2 データベース・インスタンスが予期せずに終了すると、DB2 障害モニターはインスタンスを再始動します。データベース・ソリューションがクラスターにインプリメントされている場合は、DB2 障害モニターの代わりに、障害の発生したデータベース・インスタンスを再始動するようにクラスター管理ソフトウェアを構成してください。

クラスター環境でのハートビート・モニター

クラスター管理ソフトウェアは、クラスターのノード間のハートビート・メッセージを使用して、ノードの正常性をモニターします。クラスター・マネージャーは、ノードでのメッセージの応答または送信が停止すると、ノードで障害が発生したことを検出します。

DB2 高可用性災害時リカバリー (HADR) データベースのモニター

HADR フィーチャーには独自のハートビート・モニターがあります。 1 次データベースとスタンバイ・データベースはそれぞれ、一定のインターバルで他方からハートビート・メッセージを受け取ることを予期します。

高可用性災害時リカバリー (HADR) のモニター

モニターは、HADR セットアップの設定および保守に不可欠な部分です。DB2 モニター・インターフェースでは、ご使用の環境の構成とヘルスに関する詳細な状況を知ることができます。

HADR データベースの状況をモニターする方法は複数あります。 HADR のモニター方法として推奨されるのは、次の 2 つです。

- db2pd コマンド
- MON_GET_HADR 表関数

また、以下の方法を使用することもできますが、これらの方法は バージョン 10.1 から推奨されなくなり、今後のリリースで除去される可能性があります。

- **GET SNAPSHOT FOR DATABASE** コマンド
- db2GetSnapshot API
- SNAPHADR 管理ビュー
- SNAP_GET_HADR 表関数

- その他のスナップショット管理ビューおよび表関数

db2pd コマンド

このコマンドは、DB2 メモリー・セットから情報を検索します。このコマンドは、1 次データベースまたはスタンバイ・データベースのいずれかから発行できます。複数スタンバイ・モードを使用しており、スタンバイからこのコマンドを発行する場合、他のスタンバイについての情報は返しません。このコマンドを 1 次データベースから発行すると、すべてのスタンバイに関する情報が返されます。

データベース HADRDB の高可用性災害時リカバリーについての情報を確認するには、以下のコマンドを発行します。

```
db2pd -db HADRDB -hadr
```

1 次からそのコマンドを発行したとすると、次のサンプル出力のような出力を受け取ります。

```
Database Member 0 -- Database HADRDB -- Active -- Up 0 days 00:23:17 -- Date 06/08/2011 13:57:23

                           HADR_ROLE = PRIMARY
                      REPLAY_TYPE = PHYSICAL
                    HADR_SYNCMODE = SYNC
                       STANDBY_ID = 1
                    LOG_STREAM_ID = 0
                       HADR_STATE = PEER
                PRIMARY_MEMBER_HOST = hostP.ibm.com
                   PRIMARY_INSTANCE = db2inst
                     PRIMARY_MEMBER = 0
                STANDBY_MEMBER_HOST = hostS1.ibm.com
                   STANDBY_INSTANCE = db2inst
                     STANDBY_MEMBER = 0
                HADR_CONNECT_STATUS = CONNECTED
           HADR_CONNECT_STATUS_TIME = 06/08/2011 13:38:10.199479 (1307565490)
         HEARTBEAT_INTERVAL(seconds) = 25
                HADR_TIMEOUT(seconds) = 100
       TIME_SINCE_LAST_RECV(seconds) = 3
            PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
    LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
   LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
                  LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 50772
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87616
            PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
            STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                  HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                   PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
                   STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
             STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
        STANDBY_RECV_BUF_SIZE(pages) = 16
            STANDBY_RECV_BUF_PERCENT = 0
          STANDBY_SPOOL_LIMIT(pages) = 0
                PEER_WINDOW(seconds) = 0
           READS_ON_STANDBY_ENABLED = Y
    STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N
```

MON_GET_HADR 表関数

1 次データベースにおいてこの照会を実行すると、すべてのスタンバイに関する情報が返されます。スタンバイ・データベースに対して MON_GET_HADR 関数を発行する場合は、以下の点に注意してください。

- スタンバイで、スタンバイ・データベースでの読み取りを有効にする必要があります。
- HADR セットアップが複数スタンバイ・モードになっている場合でも、表関数は、他のスタンバイについての情報を返しません。

例えば、1 次データベースにおいて以下の照会を実行できます。

```
db2 "select HADR_ROLE, STANDBY_ID, HADR_STATE, varchar(PRIMARY_MEMBER_HOST,20)
 as PRIMARY_MEMBER_HOST, varchar(STANDBY_MEMBER_HOST,20)
 as STANDBY_MEMBER_HOST from table (mon_get_hadr(NULL))"
```

出力例は以下のとおりです。

```
 HADR_ROLE     STANDBY_ID HADR_STATE       PRIMARY_MEMBER_HOST  STANDBY_MEMBER_HOST
 ------------ ---------- ---------------- -------------------- --------------------
 PRIMARY                1 PEER             hostP.ibm.com        hostS1.ibm.com

 1 record(s) selected.
```

GET SNAPSHOT FOR DATABASE コマンド

このコマンドは、状況情報を収集し、出力をフォーマットします。返される情報は、コマンド発行時のデータベース・マネージャーの運用状況のスナップショットです。HADR 情報は、HADR status という見出しの下の出力に表示されます。

db2GetSnapshot API

この API は、データベース・マネージャーのモニター情報を収集し、ユーザーが割り振るデータ・バッファーに書き込みます。返される情報は、API が呼び出された時点のデータベース・マネージャーの運用状況のスナップショットです。

SNAPHADR 管理ビューと SNAP_GET_HADR 表関数

この管理ビューおよびこの表関数は、データベース・スナップショットから HADR に関する情報を返します (特に、HADR 論理データ・グループ)。

その他のスナップショット管理ビューおよび表関数

以下のスナップショット管理ビューと表関数 (HADR 固有ではなく、より幅広い情報を戻す) を使用して、HADR 情報のサブセクションを照会することができます。

- ADMIN_GET_STORAGE_PATHS
- MON_GET_TRANSACTION_LOG
- SNAPDB
- SNAPDB_MEMORY_POOL
- SNAPDETAILLOG
- SNAP_GET_DB
- SNAP_GET_DB_MEMORY_POOL

計画外の停止への応答

データベース管理ソフトウェアまたはクラスター管理ソフトウェアがデータベース・サーバーで発生した障害を検出する場合、使用しているデータベース・ソリューションがその障害に可能な限り迅速かつスムーズに応答するようにしなければなりません。そのデータベース・ソリューションによって可能であればワークロード

を再ルーティングしてユーザー・アプリケーションを障害から保護しようと試みる必要がありますし、使用できるのであれば 2 次データベースまたはスタンバイ・データベースにフェイルオーバーするように努めなければなりません。

手順

データベース管理ソフトウェアまたはクラスター管理ソフトウェアがデータベース・サーバーで発生した障害を検出する場合、ユーザー、あるいはデータベースまたはクラスター管理ソフトウェアは、以下を実行する必要があります。

1. 障害の発生したデータベース・サーバーの操作をテークオーバーする 2 次データベース・サーバーを識別し、オンラインにして、初期化します。

   DB2 高可用性災害時リカバリー (HADR) を使用して 1 次データベース・サーバーとスタンバイ・データベース・サーバーを管理している場合は、HADR がスタンバイ・データベースが 1 次データベースとの同期を維持できるように管理し、スタンバイ・データベースによる 1 次データベースのテークオーバーも管理します。
2. ユーザー・アプリケーションのワークロードを 2 次データベース・サーバーに転送します。

   DB2 クライアント・リルートは、クライアント・アプリケーションを、障害の発生したデータベース・サーバーから事前にこの目的で指定および構成されていた 2 次データベース・サーバーに自動的に転送します。
3. 障害の発生したデータベース・サーバーをシステムから除去して、修理します。

   ユーザー・アプリケーションが 2 次またはスタンバイ・データベース・サーバーに転送されると、障害が発生したデータベース・サーバーはそのサーバーが再始動または修理されるまで、クライアント・アプリケーションの要求を処理できません。例えば、1 次データベースのデータベース・インスタンスが予期せずに終了したために障害が発生した場合には、DB2 障害モニター機能によって 1 次データベースが自動的に再始動します。

自動クライアント・リルートの例

DB2 Data Server クライアント・リルートは、クライアント・アプリケーションを障害が生じたデータベース・サーバーから、この目的のために事前に指定および構成された 2 次データベース・サーバーに自動的にリルートできます。この DB2 Data Server の機能をテストおよび例示するクライアント・アプリケーションは簡単に作成できます。

以下は、クライアント・アプリケーションでの自動クライアント・リルートの例です (疑似コードだけを使用しています)。

```
int checkpoint = 0;

check_sqlca(unsigned char *str, struct sqlca *sqlca)
{
   if (sqlca->sqlcode == -30081)
   {
      // as communication is lost, terminate the application right away
      exit(1);
   }
   else
```

```
      // print out the error
      printf(...);

 if (sqlca->sqlcode == -30108)
 {
    // connection is re-established, re-execute the failed transaction
         if (checkpoint == 0)
         {
             goto checkpt0;
         }
    else if (checkpoint == 1)
         {
            goto checkpt1;
         }
         else if (checkpoint == 2)
         {
             goto checkpt2;
         }
         ....
         exit;
 }
 }
}

main()
{
   connect to mydb;
   check_sqlca("connect failed", &sqlca);

checkpt0:
   EXEC SQL set current schema XXX;
   check_sqlca("set current schema XXX failed", &sqlca);

   EXEC SQL create table t1...;
   check_sqlca("create table t1 failed", &sqlca);

   EXEC SQL commit;
   check_sqlca("commit failed", &sqlca);

   if (sqlca.sqlcode == 0)
   {
      checkpoint = 1;
   }

checkpt1:
   EXEC SQL set current schema YYY;
   check_sqlca("set current schema YYY failed", &sqlca);

   EXEC SQL create table t2...;
   check_sqlca("create table t2 failed", &sqlca);

   EXEC SQL commit;
   check_sqlca("commit failed", &sqlca);

   if (sqlca.sqlcode == 0)
   {
      checkpoint = 2;
   }
...
}
```

クライアント・マシンでは、「mydb」というデータベースがカタログされてノード「hornet」を参照します。この「hornet」もまた、ノード・ディレクトリーでカタログされます (ホスト名「hornet」、ポート番号 456)。

非 HADR データベースを使用する例

サーバー「hornet」(ホスト名およびポート番号は hornet と同じ) において、データベース「mydb」が作成されます。さらに、代替サーバー (ホスト名「montero」、ポート番号 456) にもデータベース「mydb」が作成されます。それに加えて、サーバー「hornet」において、データベース「mydb」用の代替サーバーを以下のように更新する必要があります。

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

上記のサンプル・アプリケーションでは、自動クライアント・リルート・フィーチャーをセットアップしない場合、 `create table t1` ステートメントで通信エラーが発生するとアプリケーションは終了します。自動クライアント・リルート・フィーチャーをセットアップした場合、 DB2 データベース・マネージャーはホスト「hornet」(ポート 456) への接続を再び確立しようとします。このホストがまだ非稼働であれば、DB2 データベース・マネージャーは ALTERNATE SERVER の場所 (ホスト「montero」、ポート 456) への接続を試行します。代替サーバー・ロケーションへの接続で通信エラーが発生しなければ、アプリケーションは後続のステートメントを実行し続ける (そして、失敗したトランザクションを再実行する) ことができます。

HADR データベースを使用する例

サーバー「hornet」(ホスト名およびポート番号は hornet と同じ) において、 1 次データベース「mydb」が作成されます。さらに、ホスト「montero」(ポート 456) においてスタンバイ・データベースが作成されます。 1 次およびスタンバイ・データベース用に HADR をセットアップする方法については、「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」を参照してください。それに加えて、データベース「mydb」用の ALTERNATE SERVER を以下のように更新する必要があります。

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

上記のサンプル・アプリケーションでは、自動クライアント・リルート・フィーチャーをセットアップしない場合、 `create table t1` ステートメントで通信エラーが発生するとアプリケーションは終了します。自動クライアント・リルート・フィーチャーをセットアップした場合、 DB2 データベース・システムはホスト「hornet」(ポート 456) への接続を再び確立しようとします。これがうまくいかない場合、DB2 データベース・システムは代替サーバー・ロケーション (ポート 456 のホスト「montero」) への接続を試行します。代替サーバー・ロケーションへの接続で通信エラーが発生しなければ、アプリケーションは後続のステートメントを実行し続ける (そして、失敗したトランザクションを再実行する) ことができます。

SSL を使用する例

接続に SSL を使用している間も、クライアント・リルートを使用できます。セットアップは、前述の例で示した HADR データベースのものと似ています。

クライアント・マシンでは、データベース「mydb」のためのデータベース別名「mydb_ssl」がカタログされて、ノード「hornet_ssl」を参照します。「hornet_ssl」はノード・ディレクトリーにカタログされます (ホスト名「hornet」、SSL ポート番号 45678、セキュリティー・パラメーターは SSL に設定)。

データベース別名は、代替サーバーにもカタログされます (ホスト名は「montero」、SSL ポート番号は 45678、セキュリティー・パラメーターは SSL に設定)。それに加えて、サーバー「hornet」において、別名「mydb_ssl」用の代替サーバーを以下のように更新する必要があります。

```
db2 update alternate server for database mydb_ssl using hostname montero port 45678
```

上記のサンプル・アプリケーションで、接続ステートメントを `connect to mydb_ssl` に変更します。自動クライアント・リルート・フィーチャーをセットアップしない場合、 `create table t1` ステートメントで通信エラーが発生するとアプリケーションは終了します。自動クライアント・リルート・フィーチャーをセットアップした場合、 DB2 データベース・マネージャーは SSL を使用してホスト「hornet」(ポート 45678) への接続を再び確立しようとします。これがうまくいかない場合、DB2 データベース・マネージャーは SSL を使用して代替サーバー・ロケーション (ポート 45678 のホスト「montero」) への接続を試行します。代替サーバー・ロケーションへの接続で通信エラーが発生しなければ、アプリケーションは後続のステートメントを実行し続ける (そして、失敗したトランザクションを再実行する) ことができます。

HADR フェイルオーバー操作の実行

現在の 1 次データベースが使用可能でないために、現在のスタンバイ・データベースを新しい 1 次データベースにすることを希望する場合、フェイルオーバーを実行することができます。

このタスクについて

警告:

この手順では、データが消失する可能性があります。この非常手順を実行する前に、次の情報を検討してください。

- 1 次データベースがデータベース・トランザクションを処理していないことを確認してください。 1 次データベースが実行中だが、スタンバイ・データベースと通信できない場合、(**BY FORCE** オプションを指定した **TAKEOVER HADR** コマンドを発行して) 強制的にテークオーバー操作を実行すると、 1 次データベースが 2 つできる可能性があります。 1 次データベースが 2 つ存在するときには、それぞれのデータベースには異なるデータが存在し、これらの 2 つのデータベースが自動的に同期化されることはなくなります。
  - 1 次データベースを非活動化するか、可能であれば、そのインスタンスを停止してください。 (1 次システムが、ハング、破損、またはアクセス不能である場合、これは不可能である可能性があります。) テークオーバー操作の実行後、障害が発生したデータベースが後で再始動される場合、自動的に 1 次データベースの役割であると見なされることはありません。
- トランザクション消失の可能性と程度は、それぞれ特定の構成および環境に応じて異なります。
  - ピア状態または切断済みピア状態のときに 1 次データベースに障害が発生する場合で、同期モードが同期 (SYNC) である場合、スタンバイ・データベースは、1 次データベースの障害発生前にアプリケーションへコミットされたと報告のあったトランザクションを消失することはありません。

– ピア状態または切断済みピア状態のときに 1 次データベースに障害が発生する場合で、同期モードが準同期 (NEARSYNC) である場合、スタンバイ・データベースは、 1 次データベースとスタンバイ・データベースの両方に同時に障害が発生する場合に、1 次データベースによってコミットされたトランザクションだけを消失する可能性があります。
– ピア状態または切断済みピア状態のときに 1 次データベースに障害が発生する場合で、同期モードが非同期 (ASYNC) である場合、スタンバイ・データベースは、スタンバイ・データベースがテークオーバー操作の実行前にトランザクションの全ログ・レコードを受け取らなかった場合に、1 次データベースによってコミットされたトランザクションを消失する可能性があります。スタンバイ・データベースは、受け取ったすべてのログをディスクに書き込む前にクラッシュした場合も、1 次データベースによってコミットされたトランザクションを失うことがあります。

  注: ASYNC モードではピア・ウィンドウは許可されません。したがって、このモードでは 1 次データベースが切断済みピア状態になることはありません。
– リモート・キャッチアップ状態のときに 1 次データベースに障害が発生し、同期モードが超非同期 (SUPERASYNC) である場合、テークオーバー操作の実行前にスタンバイ・データベースがトランザクションの全ログ・レコードを受け取っていなければ、スタンバイ・データベースは 1 次データベースによってコミットされたトランザクションを失う可能性があります。スタンバイ・データベースは、受け取ったすべてのログをディスクに書き込む前にクラッシュした場合も、1 次データベースによってコミットされたトランザクションを失うことがあります。

  注: SUPERASYNC モードでは、データベースがピア状態になったり切断済みピア状態になったりすることはありません。リモート・キャッチアップ状態でフェイルオーバー (強制的テークオーバー) が許可されるのは、同期モードが SUPERASYNC の場合のみです。
– リモート・キャッチアップ・ペンディング状態のときに 1 次データベースに障害が発生する場合、スタンバイ・データベースが受け取って処理していないトランザクションは消失します。

  注: データベース・スナップショットにログのギャップが示される場合、それは、 1 次データベースとスタンバイ・データベースが最後に相互に通信した時点でのギャップです。 1 次データベースは、その時点以降、非常に大量のトランザクションを処理した可能性があります。

• 新しい 1 次に接続する (またはクライアント・リルートによって新しい 1 次に転送される) あらゆるアプリケーションが、以下の事態に対処する準備ができていることを確認します。
  – フェイルオーバー中にはデータ損失があります。新しい 1 次には、古い 1 次でコミットされたすべてのトランザクションがあるわけではありません。これは、**hadr_syncmode** 構成パラメーターが SYNC に設定されている場合でも発生することがあります。HADR スタンバイはログを順次適用するので、SQL セッションの中のあるトランザクションが新しい 1 次上でコミットされている場合、同じセッションの前のトランザクションもすべて新しい 1 次上でコミ

ットされていると想定できます。複数のセッションに渡るトランザクションのコミットの順序は、ログ・ストリームを詳細に分析しないと判断できません。
  - トランザクションを元の 1 次に発行し、元の 1 次でコミットし、新しい 1 次 (元のスタンバイ) に複製することはできても、これをコミット済みとして報告できない場合があります。このような事態が生じるのは、トランザクションがコミットされたという報告を元の 1 次からクライアントへ送れるようになる前に、元の 1 次が破損したためです。アプリケーションを作成するときには、元の 1 次に発行されたものの、元の 1 次でコミットされたとして報告されないトランザクションを、新しい 1 次 (元のスタンバイ) でコミットするという処理を行えるようにしなければなりません。
  - 一部、複製されない操作があります。データベース構成への変更や、外部 UDF オブジェクトへの変更などです。
- **TAKEOVER HADR** コマンドは、スタンバイ・データベース上でのみ発行できます。
- HADR は、障害が発生したデータベースを自動的に再始動する際に使用できる、DB2 障害モニター (db2fm) とのインターフェースはありません。障害モニターが使用可能な場合、障害が発生したと思われる 1 次データベースでの、行われる可能性のある障害モニター・アクションに注意する必要があります。
- テークオーバー操作は、1 次データベースとスタンバイ・データベースがピア状態であるか、スタンバイ・データベースがリモート・キャッチアップ・ペンディング状態の場合のみ行えます。スタンバイ・データベースが他の状態である場合、エラーが戻されます。

  注: ローカル・キャッチアップ状態のスタンバイ・データベースを、標準データベースに変換することにより、通常の使用で使用可能にすることができます。このためには、**DEACTIVATE DATABASE** コマンドを発行してデータベースをシャットダウンしてから、 **STOP HADR** コマンドを発行します。 HADR が停止したら、以前のスタンバイ・データベースを使用可能にする前に、以前のスタンバイ・データベースでロールフォワード操作を完了する必要があります。データベースをスタンバイ・データベースから標準データベースへ変換した後は、そのデータベースを HADR ペアに再統合することはできません。 2 つのサーバーで HADR を再始動するには、HADR を初期設定するための次の手順に従ってください。
  ピア・ウィンドウを構成した場合、関連したフェイルオーバーでトランザクションが消失しないようにするため、ウィンドウの有効期限が切れる前に 1 次データベースをシャットダウンします。

フェイルオーバー・シナリオでは、テークオーバー操作は、コマンド行プロセッサー (CLP)、または db2HADRTakeover アプリケーション・プログラミング・インターフェース (API) を使用して実行できます。

手順

次の手順では、CLP を使用して、1 次データベースまたはスタンバイ・データベースでフェイルオーバーを開始する方法を示します。

1. 障害が発生した 1 次データベースを完全に使用不可にします。データベースに内部エラーが生じる場合、通常シャットダウン・コマンドでは、 1 次データベースを完全にシャットダウンできません。プロセス、共有メモリー、またはネットワーク接続などのリソースを除去するには、オペレーティング・システム・コマンドを使用しなければならない可能性があります。

2. スタンバイ・データベースで **BY FORCE** オプションを指定した **TAKEOVER HADR** コマンドを発行します。 次の例では、フェイルオーバーはデータベース LEAFS で行われます。

   ```
   TAKEOVER HADR ON DB LEAFS BY FORCE
   ```

   1 次データベースはオフラインになるものと予想されるため、 **BY FORCE** オプションが必要になります。

   1 次データベースを完全に使用不可にしない場合、スタンバイ・データベースは、1 次データベースに接続されたままになり、シャットダウンするようにというメッセージを 1 次データベースに送信します。スタンバイ・データベースは、1 次データベースがシャットダウンされたことを示す確認を受け取るかどうかにかかわらず、 1 次データベースの役割に切り替えられます。

高可用性災害時リカバリーでのデータベース役割の切り替え (HADR)

高可用性災害時リカバリー (HADR) 時に、1 次データベースとスタンバイ・データベースの役割を切り替えるには、 **TAKEOVER HADR** コマンドを使用します。

このタスクについて

- **TAKEOVER HADR** コマンドは、スタンバイ・データベース上でのみ発行できます。コマンドの発行時に 1 次データベースがスタンバイ・データベースに接続されていない場合、テークオーバー操作は失敗します。
- **TAKEOVER HADR** コマンドは、1 次データベースとスタンバイ・データベースがピア状態の場合にだけ、それらのデータベース間の役割を切り替えるために使用できます。スタンバイ・データベースが他の状態である場合、エラー・メッセージが戻されます。

手順

HADR データベースの役割を切り替えるには、次のようにします。

- CLP を使用して、スタンバイ・データベースでテークオーバー操作を開始して、スタンバイ・データベース上で **BY FORCE** オプションを指定しない **TAKEOVER HADR** コマンドを発行します。

  次の例では、スタンバイ・データベース LEAFS でテークオーバー操作が行われます。

  ```
  TAKEOVER HADR ON DB LEAFS
  ```

  ログ・フル・エラーは、テークオーバー操作の直後にやや起こりがちなエラーです。そのようなエラーの可能性を抑えるため、各テークオーバーの最後に非同期バッファー・プール・フラッシュが自動的に開始されます。非同期バッファー・プール・フラッシュが進行するにつれて、ログ・フル・エラーの可能性は下がります。それに加えて、アクティブ・ログ・スペースの量が十分な構成の場合、ログ・フル・エラーの可能性はさらに小さくなります。ログ・フル・エラーが発生すると、現行のトランザクションは異常終了し、ロールバックされます。

  注: **BY FORCE** オプションを指定しないで **TAKEOVER HADR** コマンドを発行すると、現在 HADR 1 次データベースに接続しているアプリケーションはすべて強制的にオフになります。このアクションは、役割の切り替え後にクライアントを

新しい HADR 1 次データベースに転送することを支援するために、自動クライアント・リルートと連携して動作することを意図したものです。しかし、アプリケーションを 1 次データベースから強制的にオフにすることがご使用の環境に破壊的な影響を及ぼしかねない場合は、役割の切り替えを実行する前にそのようなアプリケーションをシャットダウンし、役割の切り替えが完了してから新しい HADR 1 次データベースをターゲットとしてそれらのアプリケーションを再始動する、という独自の手順をインプリメントすることができます。

- アプリケーションから db2HADRTakeover アプリケーション・プログラミング・インターフェース (API) を呼び出します。
- IBM Data Studio で、**TAKEOVER HADR** コマンドのタスク・アシストを開きます。

関連情報:

テークオーバー操作後のデータベースの再統合

1 次データベースに障害が発生したために、テークオーバー操作を高可用性災害時リカバリー (HADR) 環境で実行した場合、障害の発生したデータベースをオンラインに戻してスタンバイ・データベースとして使用するか、1 次データベースとしての状況に戻すことができます。

手順

障害の発生した 1 次データベースを新しいスタンバイ・データベースとして HADR ペアに再統合するには、次のようにします。

1. 元の 1 次データベースが存在したシステムを修復します。このことは、破損したハードウェアを修復することや、障害の発生したオペレーティング・システムをリブートすることを意味する場合があります。
2. 障害の発生した 1 次データベースをスタンバイ・データベースとして再始動します。 次の例では、データベース LEAFS がスタンバイ・データベースとして開始されます。

   ```
   START HADR ON DB LEAFS AS STANDBY
   ```

   注: データベースの 2 つのコピーが非互換のログ・ストリームを持つ場合には、再統合は失敗します。特に、HADR は、元のスタンバイ・データベースを新しい 1 次データベースに切り替える際に、元のスタンバイ・データベースで反映されていない更新情報が、元の 1 次データベースに適用されていてはいけません。この状況が生じると、新しい 1 次データベースのバックアップ・イメージをリストアするか、スプリット・ミラーを初期設定することにより、元の 1 次データベースをスタンバイ・データベースとして再始動できます。

   このコマンドが正常に戻されても、再統合が成功したことを示すわけではありません。単にデータベースが開始されたという意味に過ぎません。再統合はまだ進行中です。その後で再統合が失敗すると、データベースはシャットダウンします。**GET SNAPSHOT FOR DATABASE** コマンドまたは **db2pd** ツールを使用してスタンバイ状態をモニターし、スタンバイ・データベースがオンラインのままであること、および通常の状態遷移を進めていることを確認することが必要です。必要に応じて、管理通知ログ・ファイルおよび **db2diag** ログ・ファイルを調べてデータベースの状況を確認することができます。

次のタスク

元の 1 次データベースがスタンバイ・データベースとして HADR ペアに再結合されたら、フェイルバック操作を実行することを選択し、データベースの役割を切り替えて、元の 1 次データベースをもう一度 1 次データベースにすることができます。このフェイルバック操作を実行するには、スタンバイ・データベースで次のコマンドを発行します。

```
TAKEOVER HADR ON DB LEAFS
```

注:

1. HADR データベースがピア状態ではないか、ペアが接続されていない場合、このコマンドは失敗します。
2. 1 次データベースのオープン・セッションは強制的にクローズされ、処理中のトランザクションはロールバックされます。
3. 1 次データベースの役割とスタンバイ・データベースの役割を切り替える場合、**TAKEOVER HADR** コマンドの **BY FORCE** オプションは指定できません。

第 6 章 DB2 クラスター・サービスを使用した障害管理

このセクションには、DB2 クラスター・サービスの概要と、特定のタイプのホスト、クラスター・キャッシング・ファシリティー、およびメンバーの障害を処理する方法に関する詳しい情報が含まれています。

DB2 クラスター・サービスは、自動ハートビート障害検出を提供し、障害が検出された後、必要なリカバリー操作を自動的に開始するソフトウェアです。また、DB2 pureScale インスタンス内の各ホストが共通ファイル・システムにアクセスできるようにするクラスター・ファイル・システムも提供します。 DB2 クラスター・サービスには、IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) ソフトウェア、IBM Reliable Scalable Clustering Technology (RSCT) ソフトウェア、および IBM General Parallel File System (GPFS) ソフトウェアからのテクノロジーが組み込まれています。このテクノロジーは、DB2 pureScale Feature の重要な部分としてパッケージされています。

クラスター・キャッシング・ファシリティーの自動フェイルオーバー

1 次クラスター・キャッシング・ファシリティー (CF) で障害が発生した場合、DB2 クラスター・サービスは自動的にその再始動を試行し、1 次の役割を 2 次クラスター・キャッシング・ファシリティーにフェイルオーバーします (推奨されている 2 つのクラスター・キャッシング・ファシリティーセットアップであることを前提とします)。

DB2 pureScale インスタンス内でクラスター・キャッシング・ファシリティーが果たす重要な役割により、障害が発生したクラスター・キャッシング・ファシリティーは再始動し、できるだけ早く復元されます。障害の影響は、以下の要因によって異なります。

- DB2 pureScale インスタンス内に存在するクラスター・キャッシング・ファシリティーの数

  DB2 pureScale インスタンス内に存在するクラスター・キャッシング・ファシリティーが 1 つだけの場合、インスタンスはダウンします。ソフトウェア障害が原因でクラスター・キャッシング・ファシリティーで障害が発生する場合、グループ再始動が自動的に開始されます。ハードウェア障害が原因でクラスター・キャッシング・ファシリティーで障害が発生する場合、ユーザーは問題を修正する必要があります。問題が修正された後、グループ再始動が自動的に開始されます。

  DB2 pureScale インスタンス内に 2 つのクラスター・キャッシング・ファシリティーが存在する場合 (推奨のセットアップ)、DB2 クラスター・サービスは 1 次の役割の 2 次クラスター・キャッシング・ファシリティーへのフェイルオーバーを試行します。ソフトウェア障害が原因で 1 次クラスター・キャッシング・ファシリティーで障害が発生する場合、2 次クラスター・キャッシング・ファシリティーとして自動的に再始動し、復元されます。ハードウェア障害が原因で 1 次クラスター・キャッシング・ファシリティーで障害が発生する場合は、ユーザーが

問題を修正する必要があります。問題を修正すると自動的に再始動して、2 次クラスター・キャッシング・ファシリティーとして再統合されます。

- 1 次クラスター・キャッシング・ファシリティーで障害が発生したときの 2 次クラスター・キャッシング・ファシリティーの状態

  2 次クラスター・キャッシング・ファシリティーが PEER 状態にある場合、DB2 クラスター・サービスは 1 次の役割を 2 次にフェイルオーバーします。

  2 次クラスター・キャッシング・ファシリティーが PEER 状態にない場合、インスタンスはダウンします。インスタンスがダウンすると、DB2 クラスター・サービスはグループ再始動を開始し、2 次クラスター・キャッシング・ファシリティーだったものが 1 次の役割になります。

自動再始動

DB2 pureScale環境では、DB2 クラスター・サービスによって自動的にソフトウェアおよびハードウェア障害が検出され、発生した障害のタイプに応じてメンバー再始動またはグループ再始動が開始されます。

自動再始動は、ホスト、メンバー、またはクラスター・キャッシング・ファシリティーの障害によってデータベースが受ける影響を最小限に抑える上で役立ちます。メンバーの障害 (およびその後の再始動) はアプリケーションに透過的であり、障害が発生したメンバー上で実行されている非コミット・トランザクションに与える影響もほんの一時的です。複数のホストまたはメンバーで障害が発生していても、アプリケーションは引き続きデータベースにアクセスできます。障害が発生したメンバーを元のホストで DB2 クラスター・サービスによって再始動できない状況では、障害が発生したメンバーは、*restart light* と呼ばれるプロセスにより、別のホスト上で再始動します。 DB2 pureScale インスタンスのすべてのクラスター・キャッシング・ファシリティーが失われるなど、いくつかの極端な障害の場合には、データベース障害が発生します。そのような場合には、 DB2 クラスター・サービスは、クラスター・キャッシング・ファシリティーおよびメンバーのグループ再始動を開始します。

メンバー再始動とクラッシュ・リカバリー

メンバー再始動 とは、障害が発生したメンバー上のデータベース・サーバー・プロセスを再始動し、メンバー・クラッシュ・リカバリーが必要な各データベースでそれを実行するプロセスのことです。

ホスト上のソフトウェアまたはハードウェア障害が原因でメンバーで障害が発生する場合、DB2 クラスター・サービスはその障害を検出し、メンバーを自動的に再始動します。メンバー再始動は、*ローカル再始動* か *restart light* のいずれかになります。前者は、元のホスト (ホーム・ホスト) で再始動することを意味し、後者は別のホストで再始動することを意味します。

ローカル再始動

ソフトウェア障害が原因でメンバーで障害が発生したが、メンバーのホーム・ホストは引き続きアクティブになっている場合、 DB2 クラスター・サービスによってローカル再始動が試行されます。ローカル・メンバー再始動では、未完了データを整合状態に迅速にリカバリーするために、削減メモリ

ー・モデルが使用されます。未完了データがリカバリーされた後、完全なトランザクション処理を行うために、データベースの通常のメモリー・モデルが初期化されます。

restart light
メンバーのホーム・ホストが非アクティブであるか、ローカル再始動の試行に失敗する場合、メンバーは、最小限のリソースを使用して別のメンバーのホーム・ホスト上のゲスト・メンバー として自動的に再始動します。
restart light モードで実行されているメンバーは、新しいトランザクションの処理を行いません。メンバー・クラッシュ・リカバリーを実行することがその唯一の目的だからです。

複数のメンバーで障害が発生した場合、一般的には、個々のメンバーの再始動リカバリーを同時に行うことでリカバリーできます。したがって、グループ再始動は通常、必要ありません。障害が発生していない他のメンバーがアクセスできるように、データベースは引き続き開いたままになります。障害が発生したメンバー上の未完了データについてのみ、メンバーの再始動の間、使用できなくなります。

メンバー・クラッシュ・リカバリー

メンバー・クラッシュ・リカバリーは、メンバー再始動の一部として、未完了トランザクションをロールバックし、コミット済みトランザクションを完了する役割を持ちます。これにより、このメンバーによって変更されたデータベース・データは整合状態に戻されます。メンバー・クラッシュ・リカバリーの終了後に未確定トランザクションが残っている場合、それを解決するためにメンバーが使用可能になります。

メンバー・クラッシュ・リカバリーは、実行可能なクラスター・キャッシング・ファシリティーが引き続き有効で、メンバー上のデータベースがアクティブになっている場合に実行されます。これにより、異常終了が原因でディスク上のメンバーのログ・ストリームが不整合な状態にあることが分かります。ほとんどの場合、メンバー・クラッシュ・リカバリーは、メンバー再始動および自動リカバリー・エージェントによって自動的に呼び出されます。自動リカバリー・エージェントはインスタンスが始動したときに開始され、メンバー上のデータベースが不整合状態にあることを検出したときにアクションを取ります。

データベースのメンバー・クラッシュ・リカバリーが完了した後、メンバーは、元のホストで再始動した場合に、他のアプリケーションからの着信接続要求を受け入れることができるようになります。

グループ再始動とクラッシュ・リカバリー

グループ再始動 とは、すべてのメンバーのデータベース・サーバー・プロセスおよびクラスター・キャッシング・ファシリティーを再始動し、グループ・クラッシュ・リカバリーを実行してデータベースをオンライン状態に戻すことにより、DB2 pureScale インスタンス全体を再始動するプロセスのことです。

グループ再始動は、実行可能な 1 次クラスター・キャッシング・ファシリティーが DB2 pureScale インスタンスにないときに発生します。このイベントは、DB2 クラスター・サービス によって自動的に検出されて処理されます。グループ再始動は、1 次クラスター・キャッシング・ファシリティーとメンバーが使用可能になるとす

ぐ、自動的に開始されます。グループ再始動が発生すると、データベースはすべてのメンバーからアクセス不能になります。

グループ再始動が必要になる可能性のある状況は、多くはありませんが、いくつかあります。例えば、以下のような状況です。

- 1 つのクラスター・キャッシング・ファシリティーだけを使ってインスタンスが実行されており、そのクラスター・キャッシング・ファシリティーで障害が発生する場合。
- 2 次クラスター・キャッシング・ファシリティーが PEER 状態になる前に 1 次クラスター・キャッシング・ファシリティーで障害が発生する場合。
- 両方のクラスター・キャッシング・ファシリティーで障害が発生する場合。

グループ・クラッシュ・リカバリー

グループ・クラッシュ・リカバリーは、データベースの整合性を保つ役割を持ちます。ディスクに書き込まれなかったすべての処理を再実行し、障害が発生したときにコミットされなかったすべてのトランザクションをロールバックすることによってこれを行います。グループ・クラッシュ・リカバリーは DB2 pureScale環境外の DB2 クラッシュ・リカバリーに似ています。ただし、グループ・クラッシュ・リカバリーでは、データベースでアクティブになっているすべてのメンバーからのマージ・ログ・ストリームが使用されます。グループ・クラッシュ・リカバリーはグループ再始動の一部として自動的に発生するので、一般的には、機能しているクラスター・マネージャーが存在している間は、グループ・クラッシュ・リカバリーが必要なときにユーザーが何らかのアクションを取る必要はありません。

restart light

障害が発生した メンバーを元のホスト、つまりホーム・ホスト で再始動できない場合、 DB2 クラスター・サービスは DB2 pureScale インスタンス内の使用可能な別のホストの 1 つでそのメンバーを再始動します。このプロセスを *restart light* と呼びます。このプロセスでは、インスタンス内の他のホストに著しい影響を与えることなくメンバー・クラッシュ・リカバリーを実行できます。

別のホストで restart light モードで再始動したメンバーをゲスト・メンバー と呼び、ホーム・ホストで実行されているメンバーを*常駐メンバー* と呼びます。ゲスト・メンバーが使用するリソースは常駐メンバーより少なく、ゲスト・メンバーは外部アプリケーションからのインスタンス接続またはデータベース接続を受け入れることができません (SQL1032N)。

restart light モードでメンバーを開始する唯一の目的は、メンバー・クラッシュ・リカバリーを実行することです。ゲスト・メンバーは、ホストが restart light に使用される常駐メンバーへの影響を最小限に抑えるために、事前に割り振られた削減メモリー・モデルを使用して再始動します。restart light モードで実行されているゲスト・メンバーは、必要なすべてのデータベース上でメンバー・クラッシュ・リカバリーを完了した後、ホーム・ホストにフェイルバックされるのを待ちます。ホーム・ホストにフェイルバックされるまで、新しいトランザクションを処理できません。 restart light モードのメンバーは、その状態が照会される場合に、稼働中として表示されます (ただし WAITING_FOR_FAILBACK 状態)。障害が発生したホーム・ホストがオンラインに戻ると、restart light モードのメンバーは DB2 クラスタ

ー・サービスによって自動的に独自のホーム・ホストにフェイルバックされ、新しいトランザクションを再び処理できる、完全にアクティブなメンバーになります。

restart light は自動化されたプロセスで、次のように機能します。ソフトウェアで障害が発生したときにメンバーのホーム・ホストが引き続きアクティブになっている場合、 DB2 クラスター・サービスは、そのホーム・ホストで、障害が発生したメンバーの再始動を試行します。障害が発生したメンバーをホーム・ホストで再始動する最初の試行が失敗すると、そのメンバーは別のホスト上でゲスト・メンバーとして light モードで再始動します。 restart light は、以下のいずれかの状況が発生した場合にメンバーで即時に開始されます。

- ネットワーク障害が発生し、ホーム・ホストがDB2 pureScale インスタンスの他のホストとの接続を失う場合
- 電源障害またはハードウェア、LPAR、VM、または OS のリセットによりホーム・ホストが予期せずに非アクティブになる場合
- 保守のためにホーム・ホストが停止されるときにメンバーが異常終了する場合

restart light は、非常に高速なプロセスです。各ホストに DB2 アイドル・プロセスのセットがあるからです。このプロセスは、ゲスト・メンバーの再始動のためのリソースを事前に割り振ります。 DB2 は、新しいプロセスを作成する代わりにそのプロセスをアクティブにして、メンバーの再始動を light モードで実行します。restart light では新しいプロセスが作成されず、ゲスト・メンバーと常駐のメンバーの間でリソースを獲得するために競合することがないので、メンバーのリカバリー処理が速くなります。

自動メンバー・フェイルバックの無効化

状況によっては、自動フェイルバックを遅らせたい場合があります。これを行うには **db2cluster** コマンドを使用します。

このタスクについて

デフォルトでは、DB2 クラスター・サービスは、メンバーのホーム・ホストが使用可能になり、アラートがすべてクリアされると即時に、restart light モードで実行しているメンバーをフェイルバックします。ホスト障害の原因を調査したい場合は、適当な時期が来るまで、メンバーの自動フェイルバックを無効にすると役立つことがあります。

手順

自動フェイルバックを無効にするには、以下のようにします。

1. 次のように **db2cluster** コマンドを発行します。

   ```
   db2cluster -cm -set -option autofailback -value off
   ```

2. DB2 pureScaleインスタンスを再始動します。

タスクの結果

restart light モードにあるすべてのメンバー、および将来のすべての restart light メンバーは、メンバー アラートがすべてクリアされ、メンバーの常駐ホストがアクティブになっている場合でも、それぞれのゲスト・ホスト上に残ります。

例

メンバーが light モードで再始動するホスト障害が繰り返し発生するので、DBA は、障害の原因を判別できるようになるまで障害のあるホストからそれぞれの常駐メンバーを切り離しておくのが良いと決定しました。 DBA は次のコマンドを実行します。

```
db2cluster -cm -set -option autofailback -value off
```

次のメッセージが返されます。

```
The db2cluster command succeeded. The AUTOFAILBACK option has been set to "OFF". Automatic
failback will be disabled the next time that the DB2 database manager instance is restarted.
```

注: 何らかの理由で自動フェイルバックが既に使用不可になっていた場合は、コマンドは正常に完了しますが、インスタンスの再始動について言及するメッセージは出ません。

DBA はこのインスタンスを再始動します。ホスト障害の原因の調査が完了すると、DBA は次のコマンドを実行して、自動メンバー・フェイルバックをオンにします。

```
db2cluster -cm -set -option autofailback on
```

次のメッセージが返されます。

```
The db2cluster command succeeded. The AUTOFAILBACK option has been set to "ON". Automatic
failback will be enabled the next time that the DB2 database manager instance is restarted.
```

インスタンスが再始動した後に DBA は次のコマンドを使用して、自動メンバー・フェイルバックが有効になっていることを確認します。

```
db2cluster -cm -list -option -autofailback
```

次のメッセージが返されます。

```
The AUTOFAILBACK option is set to "ON".
```

restart light のためのメモリーの考慮事項

DB2 が始動すると、restart light モードでのメンバーのリカバリーを容易にするために、限られた量のメモリーがリカバリーの目的で予約されます。この予約された restart light メモリーは事前定義されているため、リカバリーのパフォーマンスが向上します。それは、メモリーが予約されており、リカバリー中にいつでも即時に使用できるからです。

特定のホストでの restart light リカバリーのために予約できるメモリーの量は、*rstrt_light_mem* データベース・マネージャー構成パラメーターによって制限されます。 *rstrt_light_mem* のデフォルト値は AUTOMATIC です。これは、DB2 が開始されるときに、DB2 によって自動的に、restart light リカバリーのために事前に割り振られて予約されるメモリーの量の固定上限が計算され、値が設定されることを意味します。 DB2 は、*instance_memory* および *numdb* 構成パラメーターの設定と、ホスト上のメンバーの数に基づいて、この値を計算します。自動的に計算される値の範囲はインスタンスのメモリー限度の 1% から 10% の間であり、インスタンスのメモリーの合計量に組み込まれます。ただし、予約される restart light メモリーの量は常駐メンバーのパフォーマンスに影響を与える可能性があるため、ユーザーは特定のワークロードに適切になるように restart light メモリー構成を調整できます。

予約済み restart light メモリーの表示

DB2 ホストで割り振られているメモリーの合計量に関する情報を表示するには、-totalmem オプションを指定した **db2pd** コマンドを使用します。この情報には、アクセスされる現行 DB2 ホスト上で事前に割り振られている予約済み restart light メモリーの量が含まれます。クラスター内のすべてのホストの情報を取得するには、個々のホストで並行して db2pd を実行します。以下の例では、db2pd は メンバー 20 を持つホスト B で実行されています。

```
db2pd -totalmem

                         Controller     Memory        Current        HWM           Cached
                         Automatic      Limit         Usage          Usage         Memory
Member 20                Yes           25750080 KB   9031201 KB     9391744 KB    480064 KB
 Restart Light Memory   Yes         2575008 KB     64182 KB       69265 KB       5250 KB

Total current usage: 9095383 KB
Total cached memory:  485314 KB
```

非表示バッファー・プールのリカバリー

メンバー・クラッシュ・リカバリーでは、削減メモリー・モデルがバッファー・プールに対して使用されます。通常バッファー・プールは、データベース共有メモリー・セット内のメモリー量を最も多く消費するため、大きなバッファー・プールの割り当てに非常に多くの時間がかかります。削減メモリー・モデルでは、コストが非常に高いユーザー定義の大きなバッファー・プールの代わりに、小さなリカバリー非表示バッファー・プールが割り振られるため、リカバリー・パフォーマンスが向上します。既存の非表示バッファー・プールとまったく同様、4 つのリカバリー非表示バッファー・プールがあります。サイズはそれぞれ、4K、8K、16K、および 32K です。ただし、非表示バッファー・プールのサイズは常に 16 ページです。リカバリー非表示バッファー・プールの最小サイズは 250 ページですが、restart light メモリー・セット・サイズおよびバッファー・プール・サイズの計算に応じて、それよりも大きくすることができます。

以下の例では、100 ページを持つ BP1 と 200 ページを持つ BP2 の 2 つのユーザー・バッファー・プールがデータベース TESTDB に作成されています。メンバー 0 は restart light モードにあり、メンバー 1 は restart light モードにありません。この例には、以下の db2pd コマンドからの出力の一部が含まれています。メンバー 1 は、ユーザーが作成したバッファー・プールおよび非表示バッファー・プールを示していますが、メンバー 0 は 4 つのリカバリー非表示バッファー・プールのみを示しています。

```
db2pd -allmembers -db testdb -bufferpools

Database Member 1--Database TESTDB--Active--Up 0 days 00:00:14--Date 08/12/2010 18:55:19

Bufferpools:
First Active Pool ID      1
Max Bufferpool ID         3
Max Bufferpool ID on Disk 3
Num Bufferpools           7

Address            Id   Name               PageSz     PA-NumPgs  BA-NumPgs  BlkSize
0x00002AAADE443140 1    IBMDEFAULTBP       4096       1000       0          0
0x00002AAADE45B080 2    BP1                4096       100        0          0
0x00002AAADE45F060 3    BP2                4096       200        0          0
0x00002AAADDB83CC0 4096 IBMSYSTEMBP4K      4096       16         0          0
0x00002AAADDB13CC0 4097 IBMSYSTEMBP8K      8192       16         0          0
0x00002AAADDB03CC0 4098 IBMSYSTEMBP16K     16384      16         0          0
0x00002AAADE453140 4099 IBMSYSTEMBP32K     32768      16         0          0

...NumTbsp    PgsToRemov CurrentSz  PostAlter  SuspndTSCt Automatic
```

```
   3          0          1000       1000       0          False
   0          0          100        100        0          False
   0          0          200        200        0          False
   0          0          16         16         0          False
   0          0          16         16         0          False
   0          0          16         16         0          False
   0          0          16         16         0          False


Database Member 0 -- Database TESTDB -- Active -- Up 0 days 00:00:13

Bufferpools:
First Active Pool ID      4096
Max Bufferpool ID         0
Max Bufferpool ID on Disk 3
Num Bufferpools           4

Address            Id   Name               PageSz     PA-NumPgs  BA-NumPgs  BlkSize
0x00002AAAD9F946E0 4096 IBMSYSTEMBP4K      4096       9954       0          0
0x00002AAADA743140 4097 IBMSYSTEMBP8K      8192       250        0          0
0x00002AAADA733140 4098 IBMSYSTEMBP16K     16384      250        0          0
0x00002AAADA74B080 4099 IBMSYSTEMBP32K     32768      250        0          0

...NumTbsp    PgsToRemov CurrentSz  PostAlter  SuspndTSCt Automatic
   3          0          9954       9954       0          False
   0          0          250        250        0          False
   0          0          250        250        0          False
   0          0          250        250        0          False
```

restart light の際のメモリーの消費

障害が発生したメンバーを、そのメンバーのホーム・ホスト以外のホストで、そのホストの常駐メンバーに影響を与えることなく、再始動によって迅速にリカバリーできるようにすることが理想的です。これを実現するために、restart light 用の予約済みリカバリー・メモリーをまず使用して、データベース・リカバリー操作を実行します。しかし、restart light メモリー割り振りを超えるメモリー・リソースがデータベース・リカバリーで必要になる場合、空きインスタンス・メモリーを求めて restart light によって追加メモリー要求が行われます。この重大なメモリー要求により、常駐メンバーによる現行メモリー使用量の削減が試行されます。それでもメモリー・リソースが不十分で restart light プロセスを終了できない場合、DB2 はオペレーティング・システムに追加のメモリーを要求します。これが行われる場合、重大でない他のすべてのメモリー要求は、リカバリー操作のための十分なメモリーが解放されるまで失敗します。常駐メンバー上で実行されているアプリケーションはメモリー不足障害を引き起こす可能性がありますが、常駐メンバーは稼働状態を維持します。リカバリーが完了し、データベースが整合していると、ゲスト・メンバーによって使用された、最初に予約されていたリカバリー・メモリーを超える追加メモリーはすべて解放されます。

この restart light のための追加メモリー・リソース要求は一時的なものですが、常駐メンバーのワークロードに悪影響を与える可能性があります。予約済みリカバリー・メモリーが不足していることに気付いた場合は、*rstrt_light_mem* データベース・マネージャー構成パラメーターのサイズを大きくすることを検討してください。このパラメーターは構成可能ですがオンラインではないので、変更にはグローバルな **db2stop** および **db2start** か必要になるか、またはメンバーごとに *rstrt_light_mem* を更新する場合 (すなわち、同時にすべての メンバーを停止しない場合) は、各 メンバー、および各メンバーのホスト上のインスタンスを以下のようにして停止してから始動する必要があります。

```
db2 update dbm cfg using RSTRT_LIGHT_MEM 5

db2stop member 10
```

```
db2stop instance on hostA.torolab.ibm.com
db2start instance on hostA.torolab.ibm.com
db2start member 10

db2stop member 20
db2stop instance on hostB.torolab.ibm.com
db2start instance on hostB.torolab.ibm.com
db2start member 20
```

restart light メモリー使用量の表示

常駐メンバーおよびゲスト・メンバーがホスト上で使用しているメモリーの合計量に関する情報を表示するには、以下の 2 つの方法があります。

1. 各ホストで、-totalmem オプションを指定して **db2pd** コマンドを実行する。以下に例を挙げます。
   - ホスト A 上のメンバー 0
   - ホスト B 上のメンバー 1
   - ホスト C 上のメンバー 2

   メンバー 0 は、restart light モードのホスト B にフェイルオーバーします。ユーザーは、常駐メンバー 1 を持つホスト B で **db2pd** コマンドを実行します。これは、ゲスト・メンバー 0 が restart light モードで実行されているからです。ユーザーはホスト C で **db2pd** コマンドを実行し、メンバー 2 のメモリーを表示します。表示出力では、メンバー 0 に "guest" というラベルが付けられており、メモリー使用量はキロバイト単位で表示されています。 db2pd はデータベース接続を必要としません。

   ホスト B:

```
$ db2pd -totalmem
Total Memory Statistics in KB

                    Controller    Memory     Current       HWM       Cached
                    Automatic     Limit       Usage       Usage      Memory
=============================================================================+=====
Member 0 (guest)         Yes       20677572      242496      244032       15168
Member 1                 Yes       20677572      186624      697088       46080
Restart Light Memory     Yes         839720      459392      459392       19200

Total current usage:  888512
Total cached memory:  80448
```

   ホスト C:

```
$ db2pd -totalmem
Total Memory Statistics in KB

                    Controller    Memory     Current       HWM       Cached
                    Automatic     Limit       Usage       Usage      Memory
=============================================================================+=====
Member 2                 Yes       20153284     4728832     4728832     1055104
Restart Light Memory     Yes         839720      689088      689088       28800

Total current usage:  5417920
Total cached memory:  1083904
```

2. SQL インターフェースを使用して、restart light モードのメンバーを含むすべてのメンバーのメモリー使用量を表示する。これは 1 つのホストのみから実行する必要があります。ただし、この方法ではデータベース接続が必要になるため、restart light モードのメンバーからは実行できません。 restart light メンバーは接

続を受け入れないからです。この表示では、メンバーに "guest" というラベルは付けられておらず、メモリー使用量はバイト単位で表示されています。

```
$ db2 'SELECT * FROM TABLE (SYSPROC.ADMIN_GET_MEM_USAGE()) AS T'

DBPARTITIONNUM MAX_PARTITION_MEM    CURRENT_PARTITION_MEM PEAK_PARTITION_MEM
-------------- -------------------- --------------------- --------------------
             1          21173833728            4605345792           4605345792
             0          21173833728             248840192            249888768
             2          20636962816            4651352064           4651352064

  3 record(s) selected.
```

restart light のメンバーのモニター

任意のアクティブなメンバーに対して **LIST UTILITIES** コマンドを実行することにより、restart light モードのメンバーのリカバリー進行状況をモニターできます。**LIST UTILITIES** コマンドは、restart light モードのメンバーを含む、すべてのメンバーのグローバル・リカバリー状況を返すことができます。

```
 LIST UTILITIES SHOW DETAIL

 ID                                   = 1
 Type                                 = MEMBER CRASH RECOVERY
 Database Name                        = SAMPLE
 Partition Number                     = 0
 Description                          = Member Crash Recovery (Light Mode)
 Start Time                           = 11/22/2007 15:20:05.646020
 State                                = Executing
 Invocation Type                      = Automated
 Progress Monitoring:
    Estimated Percentage Complete     = 0
    Phase Number [Current]            = 1
       Description                    = Forward
       Total Work                     = 4193976 bytes
       Completed Work                 = 0 bytes
       Start Time                     = 11/22/2007 15:20:05.646121
    Phase Number                      = 2
       Description                    = Backward
       Total Work                     = 4193976 bytes
       Completed Work                 = 0 bytes
       Start Time                     = Not Started
```

DB2 pureScale環境のポイント・イン・タイム・ビューを得るために使用できる方法はいくつかあります。例えば、 DB2_GET_INSTANCE_INFO 表関数、DB2_MEMBER 管理ビュー、**LIST INSTANCE** コマンド、および **db2instance** コマンドなどです。これらのうちのいずれかの方法を使用することで、いずれかのメンバーが restart light モードになっているかどうか、どのホストでそれがリカバリーされているか、および現在 restart light リカバリーのどの状態にあるかを判別できます。

次の例では、DB2_MEMBER 管理ビューにメンバー 2 がホスト so3 で light モードで再始動されていることが示されています。

```
SELECT ID,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CURRENT_HOST,
       varchar(STATE,21) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_MEMBER

ID     HOME_HOST CURRENT_HOST STATE      ALERT
------ --------- ------------ ---------- -----
```

```
     1 so1       so1          STARTED    NO
     2 so2       so3          RESTARTING NO
     3 so3       so3          STARTED    NO

     3 record(s) selected.
```

シナリオ: restart light

このシナリオでは、メンバーを light モードで再始動する際に生じるステップについて説明します。ここでは、最もよく見られるケースについて取り上げます。つまり、単一のホスト障害が原因で、ホストの常駐メンバーが、アクティブ状態を維持している別のホスト上で、ゲストメンバーとして自動的に再始動するというケースです。シナリオでは、ゲスト・メンバーがそのホーム・ホストにフェイルバックされる方法についても取り上げられています。

初期セットアップ

DB2 pureScale インスタンスには、6 つのホスト
(HostA、HostB、HostC、HostD、HostE、HostF) が存在しています。

- メンバー 10 は HostA (そのホーム・ホスト) で実行されています
- メンバー 20 は HostB (そのホーム・ホスト) で実行されています
- メンバー 30 は HostC (そのホーム・ホスト) で実行されています
- メンバー 40 は HostD (そのホーム・ホスト) で実行されています
- クラスター・キャッシング・ファシリティー 128 (CF 128) は HostE で実行されています
- クラスター・キャッシング・ファシリティー 129 (CF 129) は HostF で実行されています

各ホストのインスタンスには DB2 アイドル・プロセスのセットがあります。各ホストでは、restart light リカバリーのための予約済みメモリーが事前に割り振られています。 DB2 クラスター・サービスは、クラスター内のすべてのリソースをモニターします。

ホスト、メンバー、および CF の状況情報は、**LIST INSTANCE** コマンドを使用して表示できます。この時点で **LIST INSTANCE** コマンドは以下を返します。

```
LIST INSTANCE

MEMBER_ID TYPE    STATE   HOME_HOST CURRENT_HOST ALERT
--------- ------- ------- --------- ------------ -----
       10 MEMBER  STARTED hostA     hostA        NO
       20 MEMBER  STARTED hostB     hostB        NO
       30 MEMBER  STARTED hostC     hostC        NO
       40 MEMBER  STARTED hostD     hostD        NO
      128 CF      PRIMARY hostE     -            NO
      129 CF      PEER    hostF     -            NO


HOSTNAME STATE  INSTANCE_STOPPED ALERT
-------- ------ ---------------- -----
hostA    ACTIVE NO               NO
hostB    ACTIVE NO               NO
hostC    ACTIVE NO               NO
hostD    ACTIVE NO               NO
hostE    ACTIVE NO               NO
hostF    ACTIVE NO               NO
```

ホスト障害

HostA サーバーで電源障害が発生します。DB2 クラスター・サービスは HostA 上のメンバー 10 を再始動できないため、次に使用可能なホストである HostB 上で light モードでメンバーを再始動します。

この時点で **LIST INSTANCE** コマンドは、メンバー 10 の状態が現在 RESTARTING で、その現行ホストが現在 HostB であること、および HostA の状態が INACTIVE で (インスタンスが HostA で手動で停止されなかったために INSTANCE_STOPPED フィールドが設定されていないことに注意)、アラートが出ていることを示しています。

```
LIST INSTANCE

MEMBER_ID TYPE    STATE      HOME_HOST CURRENT_HOST ALERT
--------- ------- ---------- --------- ------------ -----
       10 MEMBER  RESTARTING hostA     hostB        NO
       20 MEMBER  STARTED    hostB     hostB        NO
       30 MEMBER  STARTED    hostC     hostC        NO
       40 MEMBER  STARTED    hostD     hostD        NO
      128 CF      PRIMARY    hostE     -            NO
      129 CF      PEER       hostF     -            NO


HOSTNAME STATE    INSTANCE_STOPPED ALERT
-------- -------- ---------------- -----
hostA    INACTIVE NO               YES
hostB    ACTIVE   NO               NO
hostC    ACTIVE   NO               NO
hostD    ACTIVE   NO               NO
hostE    ACTIVE   NO               NO
hostF    ACTIVE   NO               NO
```

フェイルバックの待機

プロセス・モデルが開始された後、メンバー・クラッシュ・リカバリーを必要とするそれぞれのデータベースでそれが実行されます。メンバー・クラッシュ・リカバリーの進行状況を確認するには、restart light のメンバーのモニターの説明に従って、**SHOW DETAIL** オプションを指定した **LIST UTILITIES** コマンドを使用します。メンバー・クラッシュ・リカバリーが完了した後、メンバー 10 は HostA にフェイルバックされるのを待機します。フェイルバックされるまでは、新しいトランザクションを処理できません。メンバー 10 がフェイルバックされることを待っているため、解決しなければならない未確定トランザクションが存在している可能性があります。

この時点で、**LIST INSTANCE** コマンドによって、メンバー 10 の状態が現在 WAITING_FOR_FAILBACK であることが示されます。

```
LIST INSTANCE

MEMBER_ID TYPE    STATE                HOME_HOST CURRENT_HOST ALERT
--------- ------- -------------------- --------- ------------ -----
       10 MEMBER  WAITING_FOR_FAILBACK hostA     hostB        NO
       20 MEMBER  STARTED              hostB     hostB        NO
       30 MEMBER  STARTED              hostC     hostC        NO
       40 MEMBER  STARTED              hostD     hostD        NO
      128 CF      PRIMARY              hostE     -            NO
      129 CF      PEER                 hostF     -            NO
```

```
HOSTNAME STATE    INSTANCE_STOPPED ALERT
-------- -------- ---------------- -----
hostA    INACTIVE NO               YES
hostB    ACTIVE   NO               NO
hostC    ACTIVE   NO               NO
hostD    ACTIVE   NO               NO
hostE    ACTIVE   NO               NO
hostF    ACTIVE   NO               NO
```

解決されたホストに関する問題

Power® が HostA に復元され、それによって HostA が DB2 pureScaleインスタンスで再びアクティブになります。

この時点で、**LIST INSTANCE** コマンドによって、HostA が現在アクティブであることが示され、アラートはクリアされています。

```
LIST INSTANCE

MEMBER_ID TYPE    STATE                HOME_HOST CURRENT_HOST ALERT
--------- ------- -------------------- --------- ------------ -----
       10 MEMBER  WAITING_FOR_FAILBACK hostA     hostB        NO
       20 MEMBER  STARTED              hostB     hostB        NO
       30 MEMBER  STARTED              hostC     hostC        NO
       40 MEMBER  STARTED              hostD     hostD        NO
      128 CF      PRIMARY              hostE     -            NO
      129 CF      PEER                 hostF     -            NO


HOSTNAME STATE  INSTANCE_STOPPED ALERT
-------- ------ ---------------- -----
hostA    ACTIVE NO               NO
hostB    ACTIVE NO               NO
hostC    ACTIVE NO               NO
hostD    ACTIVE NO               NO
hostE    ACTIVE NO               NO
hostF    ACTIVE NO               NO
```

ホーム・ホストへのフェイルバック

DB2 クラスター・サービスは HostA がアクティブであることを検出し、メンバー 10 をそのホストに自動的にフェイルバックします。

この時点で **LIST INSTANCE** コマンドは、メンバー 10 の状態が現在 RESTARTING で、その現行ホストが再び HostA になったことを示しています。

```
LIST INSTANCE SHOW DETAIL

MEMBER_ID TYPE    STATE      HOME_HOST CURRENT_HOST ALERT
--------- ------- ---------- --------- ------------ -----
       10 MEMBER  RESTARTING hostA     hostA        NO
       20 MEMBER  STARTED    hostB     hostB        NO
       30 MEMBER  STARTED    hostC     hostC        NO
       40 MEMBER  STARTED    hostD     hostD        NO
      128 CF      PRIMARY    hostE     -            NO
      129 CF      PEER       hostF     -            NO


HOSTNAME STATE  INSTANCE_STOPPED ALERT
-------- ------ ---------------- -----
hostA    ACTIVE NO               NO
```

```
hostB    ACTIVE NO              NO
hostC    ACTIVE NO              NO
hostD    ACTIVE NO              NO
hostE    ACTIVE NO              NO
```

ホーム・ホストでの再始動

メンバー 10 が HostA でのメンバー再始動を正常に完了すると、その状態が STARTED に変更されます。これにより、新しいトランザクションを処理し、ユーザー接続を受け入れることができるようになります。この時点で、**LIST INSTANCE** コマンドによって、DB2 pureScale インスタンスに関して初期セットアップと同じ情報が返されます。

障害状況における手操作による介入

ほとんどの障害状況では、必要なタイプの再始動またはクラッシュ・リカバリーは DB2 クラスター・サービスによって自動的に開始されるので、ユーザーがアクションを取る必要はありません。ただし、状況によっては、再始動またはクラッシュ・リカバリーを手動で開始するか、グループ・クラッシュ・リカバリーのための手順を完了する必要があります。

グループ・クラッシュ・リカバリーの開始

ほとんどの場合、グループ・クラッシュ・リカバリーは、ユーザー処置を必要とせず、自動的に実行されます。グループ・クラッシュ・リカバリーを手動で開始するのは、クラスター・マネージャーがグループ・クラッシュ・リカバリーを実行しようとして繰り返し失敗する状況、あるいは (おそらくはクラスター管理が無効になっているために) クラスター・マネージャーが存在しない状況、または **autorestart** 構成パラメーターが OFF に設定されている場合だけです。

このタスクについて

データベースでグループ・クラッシュ・リカバリーを実行できるメンバーは、1 度に 1 人だけです。 2 人のメンバーが同じデータベースのグループ・クラッシュ・リカバリーを同時に開始した場合、 2 人目のメンバーのコマンドは、最初のメンバーのグループ・クラッシュ・リカバリーが完了するまで待機します。

クラッシュ・リカバリー操作の最後でグループ・クラッシュ・リカバリーのメンバーに未確定トランザクションがある場合、グループ・クラッシュ・リカバリーを実行したメンバーはアクティブのままになり、新しい作業を受け入れることができます。ただし、未確定トランザクションによってアクセスされる行は、その後解決されるまで保護される (つまりロックされる) 必要があるため、未確定トランザクションに関連付けられたデータにはアクセスできません。グループ・クラッシュ・リカバリーのメンバーに未確定トランザクションが存在するということの意味は、未確定トランザクションが解決される前にメンバーに障害が発生した場合、データベースがそのメンバーで次にアクティブ化されるときに、後続のメンバー・クラッシュ・リカバリーが行われる必要があるということです。

グループ・クラッシュ・リカバリーがメンバー A で発行され、グループ・クラッシュ・リカバリーの過程で メンバー B がそれと 1 つ以上の未確定トランザクションを関連付けていることが検出された場合、メンバー B はグループ・クラッシュ・リ

カバリーの後で不整合のままになります。このことは、グループ・クラッシュ・リカバリーが完了した後で メンバー B をスタートする前に、メンバー・クラッシュ・リカバリーがメンバー B で発生する必要があることを意味します。メンバー・クラッシュ・リカバリーが完了した後で、メンバー B は新しい接続を受け入れることができるようになります。この後続のメンバー・クラッシュ・リカバリーは、再実行作業や取り消し作業を行う必要がない (クラッシュ・リカバリーで行う必要がある作業は未確定トランザクションをメンバーのトランザクション表にロードすることのみである) ため、非常に速く完了します。さらに、未確定トランザクションによって影響を受けるデータの保護に必要なすべてのロックは、クラスター・キャッシング・ファシリティー のメンバーのために既に予約されていることに注意してください。また、この後続の RESTART 操作は、ほとんどの場合ユーザーには非表示になっています。これは、AUTORESTART を使用可能にすると、特定のメンバーのデータベースに最初に接続したときにメンバー・クラッシュ・リカバリーが自動的に実行されるためです。

手順

グループ・クラッシュ・リカバリーを手動で開始するには、メンバーの 1 人から **RESTART DATABASE** コマンドを発行します。

タスクの結果

グループ・クラッシュ・リカバリーが完了すると、コマンドが実行されたメンバーは新しい接続を受け入れられるようになり、未解決の未確定トランザクションによってロックされたままになっているデータ以外のすべてのデータが使用可能になります。 **RESTART DATABASE** コマンドが完了した後、データベースに対する CONNECT 特権がユーザーにあれば、データベースとの接続は維持されます。グループ・クラッシュ・リカバリーが実行されたメンバー以外のメンバーでは、データベースはアクティブ化されません。

メンバー・クラッシュ・リカバリーの開始

ほとんどの場合、メンバー・クラッシュ・リカバリーは、ユーザー処置を必要とせず、自動的に実行されます。メンバー・クラッシュ・リカバリーを手動で開始するのは、クラスター・マネージャーがクラッシュ・リカバリーを実行しようとして繰り返し失敗する状況、あるいは (おそらくはクラスター管理が無効になっているために) クラスター・マネージャーが存在しない状況、または **autorestart** 構成パラメーターが OFF に設定されている場合だけです。

このタスクについて

メンバー・クラッシュ・リカバリーを必要とするメンバーが DB2 pureScale インスタンスに複数存在する場合、クラスター・マネージャーは複数のメンバー・クラッシュ・リカバリーを並行して実行します。あるメンバーで障害が発生し、それによって、そのメンバーが保持するリソースに依存している別のメンバーでも障害が発生してハング状況が生じることがありますが、この方法を使うとそれを回避できます。

手順

メンバー・クラッシュ・リカバリーを手動で開始するには、**RESTART DATABASE** コマンドを発行します。
RESTART DATABASE コマンドが発行されると、DB2 Database for Linux, UNIX, and Windows は、グループ・クラッシュ・リカバリーが必要か、それともメンバー・クラッシュ・リカバリーが必要かを判別します。

タスクの結果

メンバー・クラッシュ・リカバリーが完了した後、特定のメンバーのデータベースがアクティブ化され、他のアプリケーションからの接続を受信できるようになります。

損傷を受けた表スペースのリカバリー

表スペース・コンテナーが損傷しているときにデータベースでグループ・クラッシュ・リカバリーが必要になる場合、グループ・クラッシュ・リカバリー操作は失敗します。自動リカバリー機能を持つ DB2 pureScale環境では、以下のようにしてそのようなシナリオを解決できます。

始める前に

このシナリオは、DB2 pureScale環境内の損傷した表スペースの問題を解決して、自動グループ・クラッシュ・リカバリーを進められるようにする方法を示しています。

このタスクについて

自動リカバリーはアクティブであるものの、表スペースが損傷しているためにデータベースのリカバリーに失敗するものと仮定します。繰り返し失敗した後、 DB2 メンバーは別のホストに移動され、restart light が試行されます。しかし、その後もデータベースの自動リカバリーは失敗し続け、クラスター・マネージャーはアラートを出します。失敗したメンバーは開始されず、非アクティブになります。

手順

1. **autorestart** 構成パラメーターを OFF に設定することにより、自動リカバリーを無効にします。 **autorestart** 構成パラメーターは動的ではなく、その新しい値は、メンバーが次回開始されるときに有効になります。 次のコマンドは、WSDB という名前のデータベースの **autorestart** 構成パラメーターを OFF に変更します。

   ```
   UPDATE DATABASE CONFIGURATION FOR WSDB USING AUTORESTART OFF
   ```

2. **db2cluster** コマンドを使用して、すべてのメンバーのすべての未解決アラートをクリアします。 また、DB2 クラスター・サービスは **db2start** コマンドを自動的に発行して、そのホーム・ホストでメンバーを再始動します。再始動するメンバーは、変更された **autorestart** 値を選択しますが、自動リカバリーは開始しません。

   ```
   db2cluster –clear -alert
   ```

3. **RESTART DATABASE** コマンドを発行して、損傷を受けた表スペースを解決します。 損傷を受けた表スペースにはオフラインのマークが付けられ、ドロップ保留状態になります。

   ```
   RESTART DATABASE WSDB DROP PENDING TABLESPACES (tbsp1, tbsp2)
   ```

4. **autorestart** 構成パラメーターを ON に設定することにより、自動リカバリーを再び有効にします。 例えば、次のコマンドは、WSDB という名前のデータベースの **autorestart** 構成パラメーターを ON に変更します。

   ```
   UPDATE DATABASE CONFIGURATION FOR WSDB USING AUTORESTART ON
   ```

   autorestart パラメーターは動的ではなく、その新しい値は、メンバーが次回開始されるときに有効になります。

第 2 部 データ・リカバリー

リカバリーとは、メディアやストレージの障害、停電、アプリケーションの障害などの問題が発生した後の、データベースや表スペースの再構築のことを言います。データベースや個々の表スペースのバックアップを取っていれば、何らかの理由でそれらが損傷したり破壊されたりしても再構築できます。

リカバリーには、次の 3 つのタイプがあります。

- クラッシュ・リカバリーは、トランザクション (作業単位とも呼ばれる) が突然中断した際、データベースが不整合または使用不可の状態のままになることを防ぎます。
- バージョン・リカバリーは、以前のバージョンのデータベースの修復であり、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・リカバリーでは、バックアップ作成後にコミットされたトランザクションによって行われた変更を再適用できます。

停電後、DB2 データベース・マネージャーはクラッシュ・リカバリーを自動的に開始して、データベースのリカバリーを試みます。損傷したデータベースをリカバリーするには、バージョン・リカバリーまたはロールフォワード・リカバリーを使用します。

第 7 章 バックアップとリカバリーの計画の作成

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

リカバリー・ストラテジーを開発するときに答えが必要な質問のいくつかは、以下のとおりです。

- データベースはリカバリー可能ですか?
- データベース・リカバリーにどれくらい時間がかけられるか。
- 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、それともデータベースのフル・バックアップが必要か。
- スタンバイ・システムを、手動で構成するか、高可用性災害時リカバリー (HADR) を使用して構成するか。

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

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

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

このデータベースの再作成のことを*リカバリー* といいます。*バージョン・リカバリー* は、以前のバージョンのデータベースの修復であり、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・リカバリー* では、データ

ベースまたは表スペースのバックアップ・イメージがリストアされた後で、データベース・ログ・ファイル中に記録されているトランザクションが再度適用されます。

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

データベースを作成すると、リカバリー・ログ・ファイルとリカバリー履歴ファイルが自動的に作成されます (289 ページの図 11)。消失または損傷したデータをリカバリーする必要がある場合には、それらのログ・ファイルは重要になります。

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

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

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

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

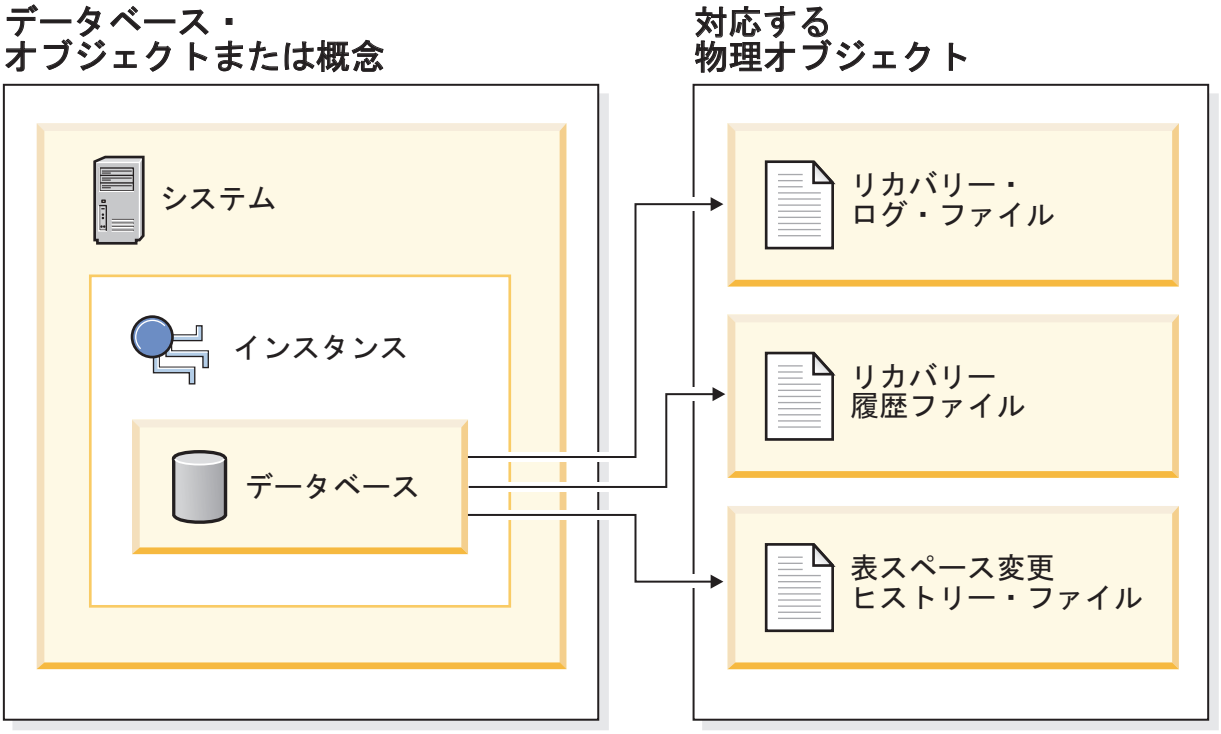


図 11. データベース・リカバリー・ファイル

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

容易に再作成できない データは、リカバリー可能データベースに保管する必要があります。これには、ロード後にソースが破棄されるデータ、表の中に手動で入力するデータ、およびデータベースにロードした後にアプリケーション・プログラムまたはユーザーによって修正されるデータが含まれます。*リカバリー可能データベース* では、**logarchmeth1** または **logarchmeth2** データベース構成パラメーターが OFF 以外の値が設定されたものです。アクティブ・ログをクラッシュ・リカバリーで使用できますが、*アーカイブ・ログ* もあり、これにはコミット済みトランザクション・データが含まれています。このようなデータベースは、オフラインでのみリストアできます。リストアすると、バックアップ・イメージが取られたときの状態に戻ります。しかし、ロールフォワード・リカバリーによって、アクティブ・ログ

およびアーカイブ・ログを使用して、特定の時点またはアクティブ・ログの最後までデータベースをロール・フォワードする (つまり、バックアップ・イメージが取られたときよりも進める) ことができます。

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

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

自動バックアップ操作

バックアップ操作のような保守活動を実行するかどうか、またいつ実行するかを判別することは時間がかかる場合があるため、自動保守を使用することができます。自動保守では、いつ自動保守を実行するかを含む、保守の方針を指定します。 DB2 は、これらの方針を使用して、保守活動を実行する必要があるかどうかを判別し、次回の保守時間枠 (自動保守活動の実行用にユーザーが定義した時間枠) 内で必要な保守活動だけを実行します。

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

バックアップの頻度の決定

データベースのバックアップには時間もシステム・リソースも必要となるため、リカバリー計画では、定期的なバックアップ操作も含める必要があります。全データベースのフル・バックアップと増分バックアップ操作を組み合わせて計画に含めることができます。また、バックアップを実行する頻度およびタイプもデータベースのリカバリー時間に影響します。

ロールフォワード・リカバリーを可能にするためにログをアーカイブしている場合でも、データベースのフル・バックアップを定期的にとるようにしてください。データベースをリカバリーするには、すべての表スペース・バックアップ・イメージを含む完全なデータベース・バックアップ・イメージを使用するか、または選択した表スペース・イメージを使用してデータベースを再構築できます。表スペースのバックアップ・イメージは、単独のディスク障害やアプリケーション・エラーからリカバリーする場合にも有効です。パーティション・データベース環境では、失敗

したデータベース・パーティション上にある表スペースのみをリストアすれば十分です。すべての表スペースまたはすべてのデータベース・パーティションをリストアする必要はありません。

表スペース・イメージからデータベースを再構築できるようになったので、データベース・リカバリーの際に完全なデータベース・バックアップは必要なくなりましたが、依然として時折データベースのフルバックアップを取ることは良い習慣です。

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

更新頻度の高いデータベースのリカバリーとロールフォワードにおいて、アーカイブ・ログを適用するために要する時間が主要な関心事である場合は、頻繁にデータベースのバックアップをとることによるコストを検討してください。バックアップの頻度を上げると、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。

オンライン・バックアップおよびオフライン・バックアップの考慮事項

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

データベースが使用できなくなる時間を短くするには、オンライン・バックアップ操作の使用が考えられます。ロールフォワード・リカバリーが可能である場合にのみ、オンライン・バックアップ操作がサポートされます。ロールフォワード・リカバリーを使用でき、リカバリー・ログの完全セットがある場合は、必要が生じたときにデータベースをリストアできます。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のみ、オンライン・バックアップ・イメージをリカバリーに使用できます。

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

選択的表スペース・バックアップの考慮事項

バックアップ・ユーティリティーを使用して、選択した表スペースのみをバックアップできます。 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、ロング・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。データを複数の表スペースに分けておくと、ディスクに障害が発生した場合でも、その障害の影響は 1 つの表スペースのみに限定される可能性が高まります。これらの表スペースの 1 つをリストアまたはロールフォワードするために要する時間は、すべてのデータを含む単一の表スペースをリストアする時間よりも短くてすみます。

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

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

表の再編成の考慮事項

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

表スペースの変更状況の考慮事項

表スペースをバックアップするかどうかについて、表スペースの変更状況を確認することにより、具体的な情報に基づいて判断を下すこともできます。**db2pd -tablespaces trackmodstate** コマンドおよび **tbsp_trackmode_state** モニター・エレメントは、最後または次のバックアップについて表スペースの状況を表示します。この情報を利用して、表スペースが変更されたかどうかや、表スペースをバックアップする必要があるかどうかを判断できます。

データベース・リカバリー時間の考慮事項

データベースのリカバリーに必要な時間は、次の 2 つの部分で構成されます。

- バックアップのリストアを完了するために必要な時間。
- データベースのフォワード・リカバリーが使用可能な場合に、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。

リカバリー計画を立てるときは、これらのリカバリーに必要な時間や作業と、それらが業務操作に与える影響を考慮してください。全般的なリカバリー計画をテストすることで、データベースをリカバリーするために必要な時間が、業務上の要件を考慮して正当かどうかを判別することができます。各テストを実施した後、バックアップを作成する頻度を増やすこともできます。リカバリー計画の一部としてロールフォワード・リカバリーを実行する場合は、このバックアップ頻度の増加により、バックアップ間にアーカイブされるログの数が減少し、その結果、リストア操作後にデータベースをロールフォワードするための時間が短縮されます。

リカバリー時のストレージに関する考慮事項

どのリカバリー方式を使用するかを決定する際には、ストレージ・スペースの要件を考慮する必要があります。バックアップ・ログ・ファイルおよびアーカイブ・ログ・ファイルの圧縮は、ご使用のデータベース環境でのストレージ・コストの削減に役立ちます。

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

表の中にロング・フィールドまたはラージ・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検討できます。このアクションは、リカバリーの計画に影響するだけでなく、ストレージ・スペースに関する考慮事項にも影響します。ロング・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、ロング・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、これらの表スペースのバックアップ頻度を少なくしたリカバリー計画にするよう決定できるかもしれません。表を作成または変更して LOB 列を組み込む際に、これらの列に対する変更内容を記録しないことを選択することもできます。このアクションにより、必要なログ・スペースおよび対応するアーカイブ・ログ・ファイル・スペースのサイズが縮小します。

メディア障害が発生したときにデータベースが破壊され、それをリストア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維持するようにしてください。データベースを作成したら、データベース・ログを別々の装置に配置するよう、 *newlogpath* 構成パラメーターを使うことを強く推奨します。

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・リカバリー方式を使用する予定の場合は、アーカイブ・ログを管理および圧縮する方法を決定する必要があります。選択できる方法は以下のとおりです。

- LOGARCHMETH1 または LOGARCHMETH2 構成パラメーターを使用して、アーカイブ・ログ・ファイル方式を指定する。
- LOGARCHCOMPR1 および LOGARCHCOMPR2 構成パラメーターを使用して、アーカイブ・ログ・ファイルの圧縮を使用可能にする。
- すでにアクティブ・セットではなくなったログを、データベース・ログ・パス・ディレクトリー以外のストレージ・デバイスまたはディレクトリーに手動でコピーする。
- ユーザー出口プログラムを使用して、これらのログを別のストレージ・デバイスにコピーする。

バックアップの圧縮

アクティブ・データベースの行圧縮によってストレージの節約が可能になるのに加えて、バックアップの圧縮を使用すると、データベース・バックアップのサイズを削減できます。

行圧縮が表単位で処理を行うのに対して、バックアップの圧縮を使用する場合は、バックアップ・イメージ内のすべて のデータ (カタログ表、索引オブジェクト、LOB オブジェクト、データベース補助ファイル、およびデータベースのメタデータを含む) が圧縮されます。

行圧縮を使用する表が含まれていても、バックアップの圧縮を使用できます。ただし、バックアップの圧縮には、追加の CPU リソースおよび時間が必要となることに留意してください。表圧縮の使用だけで、バックアップ・ストレージの縮小要件を十分に実現できる可能性があります。行圧縮を使用している場合は、バックアップの実行に余分な時間がかかることを差し置いても、ストレージの最適化を優先したい場合にのみ、バックアップの圧縮を検討してください。

ヒント: 以下の条件に当てはまる場合は、圧縮されたデータを含まない表スペースにのみ、バックアップの圧縮を使用することを検討してください。

- データおよび索引のオブジェクトが、LOB およびロング・フィールド・データと分離されている。また、
- データ表および索引の大半に、それぞれ行圧縮と索引圧縮を使用している。

バックアップの圧縮を使用するには、 **BACKUP DATABASE** コマンドの COMPRESS オプションを使用します。

アーカイブ・ログ・ファイルの圧縮

DB2 バージョン 10.1 と同様に、アーカイブ・ログ・ファイルを圧縮することができます。この機能では、データと索引を圧縮するだけでなく、バックアップも圧縮することで、データベース環境で必要なディスク・スペースの量が削減されます。

アーカイブ・ログ・ファイルは、ロールフォワード・リカバリー可能データベースの 3 番目の主なスペース・コンシューマーです。アーカイブ・ログ・ファイルには大量のデータが含まれており、それらのアーカイブは急速にサイズが増大する可能性があります。変更されたデータが既に圧縮された表にある場合、ログ・レコードに圧縮レコード・イメージが含まれるため、ロギングは減ります。アーカイブ・ログ・ファイルの圧縮により、このような環境でもストレージがさらに節約されます。

アーカイブ・ログ・ファイルの圧縮を使用するために、**UPDATE DB CFG** コマンドを使用して、**logarchcompr1** および **logarchcompr2** の各構成パラメーターを ON に設定することができます。

制約事項

- アーカイブ・ログ・ファイルの圧縮は、以下の条件下では効果はありません。
  - 対応するアーカイブ・ログ・ファイル方式が DISK、TSM、VENDOR のいずれにも設定されていない。対応するアーカイブ・ログ・ファイル方式がこれらの値に設定されている場合、ログ・ファイルは物理的にアクティブ・ログ・パスまたはミラー・ログ・パスの外に移動されます。
  - アーカイブ・ログ・ファイルの圧縮は使用可能だが、対応するログ・アーカイブ方式が OFF、LOGRETAIN または USEREXIT に設定されているときは常に、アーカイブ・ログ・ファイルを圧縮しても効果がない。このようなシナリオにな

る **logarchmeth1** および **logarchmeth2** または **logarchcompr1** および **logarchcompr2** データベース構成パラメーターの更新では、警告 SQL1663W が返されます。

- **db2adutl** を使用する手動によるアーカイブおよびリトリーブ。
  - **db2adutl** ユーティリティーでは、UPLOAD または EXTRACT の操作中に圧縮や圧縮解除は行われません。圧縮ログ・ファイルのアーカイブ・ロケーションへの移動およびアーカイブ・ロケーションからの移動は、**db2adutl** により完全にサポートされます。
  - **db2adutl** を使用してログを Tivoli Storage Manager にアップロードした状態で、アーカイブ・ログ・ファイルを圧縮する場合、ログをディスク・ロケーションにアーカイブする際 (**db2adutl** でログが取得される前) にアーカイブ・ログ・ファイルの圧縮を使用可能にする必要があります。**db2adutl** を使用して圧縮ログを手動でリトリーブすると、そのログは最初のアクセス時に抽出されます。
- アーカイブ・ログ・ファイルの圧縮は、ロー・デバイスがデータベース・ロギングで使用される場合はサポートされません。
  - アーカイブ・ログ・ファイルの圧縮は、**logpath** または **newlogpath** データベース構成パラメーターがロー・デバイスを指している場合はサポートされません。**logpath** または **newlogpath** データベース構成パラメーターがロー・デバイスを指している間は、アーカイブ・ログ・ファイルの圧縮を使用可能にするデータベース構成の更新はすべて失敗し、SQL1665N が返されます。
- **logarchcompr1** および **logarchcompr2** データベース構成パラメーターを使用して、アーカイブ・ログ・ファイルの圧縮を使用可能にする場合、バックアップ・イメージに既に保管されているログには影響しません。

関連データの一括維持

データベースの設計中に、表間に存在するリレーションシップを理解できるようにします。これらのリレーションシップは以下のレベルで表現できます。

- アプリケーション・レベル。この場合は、トランザクションは複数の表を更新します。
- データベース・レベル。この場合は、表間に参照整合性が存在するか、またはある表のトリガーが別の表に影響を与えます。

リカバリー計画を作成するときは、これらのリレーションシップ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維持することで、すべてのデータの整合性が保証されている時点まで、リカバリー処理を実行できます。これは、表スペースに対しポイント・イン・タイム指定ロールフォワード・リカバリーを実行できるようにする場合、特に重要です。

異なるオペレーティング・システムおよびハードウェア・プラットフォーム間のバックアップおよびリストア操作

DB2 データベース・システムは、異なるオペレーティング・システムおよびハードウェア・プラットフォーム間のバックアップおよびリストア操作をサポートします。

注: DB2 pureScale環境では、下位レベルのバックアップ・イメージをリストアすることはサポートされていません。

DB2 のバックアップおよびリストア操作がサポートされるプラットフォームは、 3 つのファミリーにグループ化することができます。

- ビッグ・エンディアン Linux および UNIX
- リトル・エンディアン Linux および UNIX
- Windows

あるプラットフォーム・ファミリーからのデータベース・バックアップは、同じプラットフォーム・ファミリー内のいずれかのシステムでのみリストアできます。 Windows オペレーティング・システムでは、DB2 Universal Database (DB2 UDB) V8 で作成されたデータベースを DB2 バージョン 9 データベース・システムにリストアできます。 Linux および UNIX オペレーティング・システムでは、バックアップおよびリストア・プラットフォームのエンディアン (ビッグ・エンディアンおよびリトル・エンディアン) が同じかぎり、DB2 UDB V8 で作成されたバックアップを DB2 バージョン 9 にリストアすることができます。

次の表は、DB2 がサポートする Linux および UNIX プラットフォームのそれぞれが、リトル・エンディアンとビッグ・エンディアンのいずれであるかを示します。

表 15. DB2 がサポートする、Linux および UNIX オペレーティング・システムのエンディアン。

プラットフォーム	エンディアン
AIX	ビッグ・エンディアン
IA64 上の HP	ビッグ・エンディアン
Solaris x64	リトル・エンディアン
Solaris SPARC	ビッグ・エンディアン
zSeries 上の Linux	ビッグ・エンディアン
pSeries® 上の Linux	ビッグ・エンディアン
Linux on IA-64	リトル・エンディアン
Linux on AMD64 and Intel EM64T	リトル・エンディアン
32-bit Linux on x86	リトル・エンディアン

ターゲット・システムは、ソース・システムと同じ (またはより新しい) バージョンの DB2 データベースを持っている必要があります。あるバージョンのデータベース製品で作成されたバックアップを、そのデータベース製品の以前のバージョンが稼働するシステムにリストアすることはできません。例えば、DB2 UDB V8 バックアップを DB2 V9 データベース・システムでリストアすることができますが、DB2 V9 バックアップを DB2 UDB V8 データベース・システムでリストアすることはできません。

注: 32 ビット・レベルでとったバックアップ・イメージから 64 ビット・レベルにデータベースをリストアすることはできますが、その逆はできません。データベースのバックアップおよびリストアには、DB2 のバックアップおよびリストア・ユーティリティーを使用しなければなりません。ファイル・セットを 1 つのマシンから別のマシンに移動することは、データベースの保全性を失わせる可能性があるので、推奨されません。

あるバックアップとリストアの組み合わせが許可されない状態のとき、以下に示す他の方式で DB2 データベース間で表を移動することができます。

- **db2move** コマンド
- **エクスポート**・ユーティリティーに続けて、**インポート**または**ロード**・ユーティリティーを使用する

第 8 章 リカバリー履歴ファイル

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

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

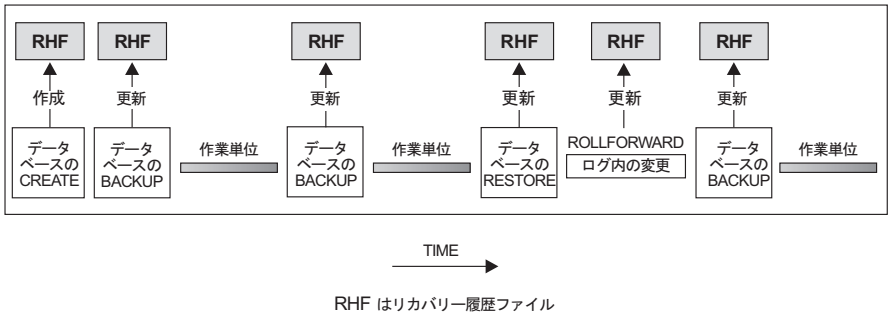


図 12. リカバリー履歴ファイルの作成と更新

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

- それぞれのレコードを一意的に識別するための識別 (ID) フィールド
- コピーされたデータベースの部分とその方法
- コピーが作成された時刻
- コピーの位置 (装置情報とこのコピーにアクセスする論理方法とを示す)
- リストア操作が行われた最終時刻

- 表スペースの名前が変更された時刻 (その表スペースの以前の名前および現在の名前を示す)
- バックアップ操作の状況: アクティブ、非アクティブ、有効期限切れ、または削除済み
- データベース・バックアップにより保管された、またはロールフォワード・リカバリー操作中に処理された最後のログ・シーケンス番号

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

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

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

ファイルのサイズは、**rec_his_retentn** 構成パラメーターで制御されており、そのファイルの項目の保持期間が日数単位で指定されます。このパラメーターの値がゼロ (0) に設定されていても、最新のデータベースのフル・バックアップ (とそのリストア・セット) は保持されます。 (このコピーを除去する唯一の方法は、 **FORCE** オプションを指定した **PRUNE** を使用することです。) 保持期間のデフォルト値は 366 日間です。この期間に **-1** を使用すると、不特定の日数を設定できます。その場合、ファイルの明示的な整理が必要になります。

リカバリー履歴ファイル項目の状況

データベース・マネージャーはリカバリー履歴ファイル内にバックアップ操作、リストア操作、表スペース作成、その他のイベントに関する項目を作成します。リカバリー履歴ファイル内の各項目には、アクティブ、非アクティブ、有効期限切れ、pending delete (削除ペンディング)、削除、または do_not_delete (削除禁止) の状況が関連付けられます。

データベース・マネージャーはリカバリー履歴ファイル項目の状況を使用して、その項目に関連付けられた物理ファイルがデータベースのリカバリーに必要かどうかを判別します。自動プルーニングの一部として、データベース・マネージャーはリカバリー履歴ファイル項目の状況を更新します。

アクティブ・データベース・バックアップ

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

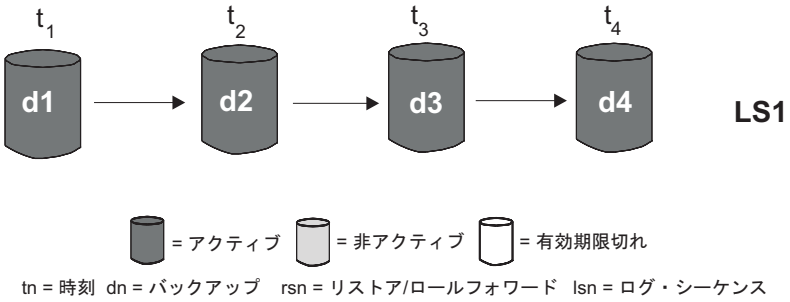


図 13. アクティブ・データベース・バックアップ: **num_db_backups** の値は 4 に設定されています。

非アクティブ・データベース・バックアップ

非アクティブ・データベース・バックアップは、リストアの際に、データベースを直前の状態に戻します。

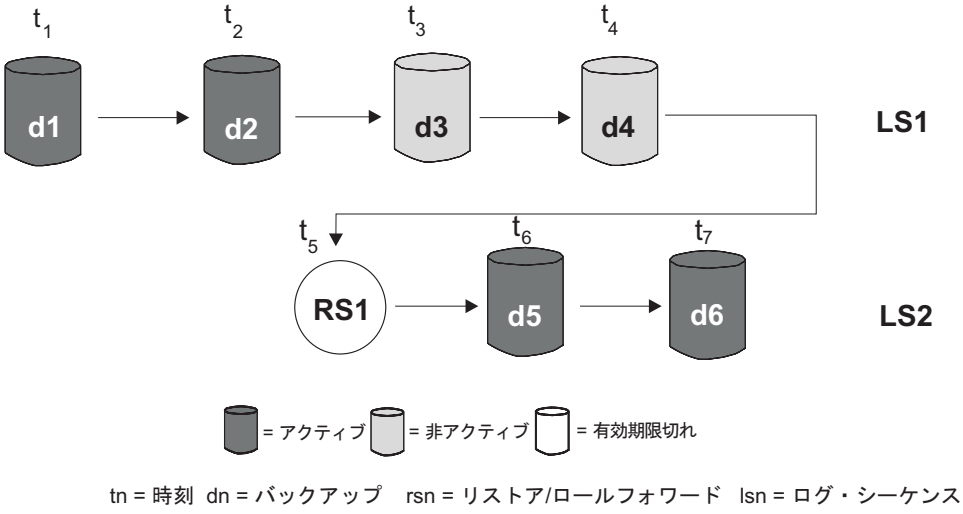


図 14. 非アクティブ・データベース・バックアップ

有効期限が切れたデータベース・バックアップ

有効期限が切れたデータベース・バックアップ・イメージは、より新しいバックアップ・イメージが使用可能であるために必要ありません。

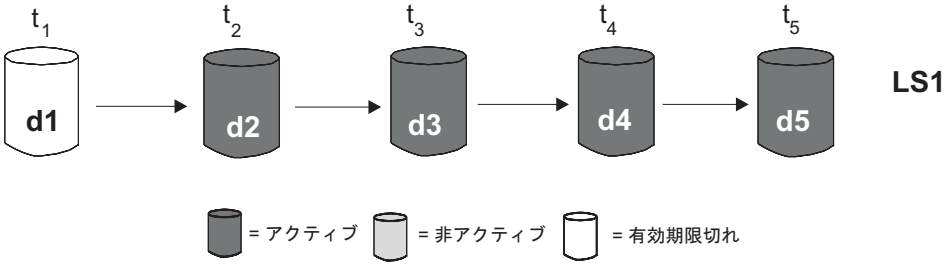


図 15. 有効期限が切れたデータベース・バックアップ

「do_not_delete」とマーク付けされた項目

PRUNE HISTORY コマンドまたは db2Prune API を使用して、リカバリー履歴ファイル項目を除去 (整理) できます。データベース・マネージャーは、リカバリー履歴ファイル項目も自動プルーニングの一部として整理します。

「do_not_delete」とマーク付けされた項目を整理する方法は、以下の 3 つだけです。

- **PRUNE HISTORY** コマンドに **WITH FORCE** オプションを指定して呼び出す
- ADMIN_CMD プロシージャーに **PRUNE HISTORY** および **WITH FORCE** オプションを指定して呼び出す
- db2Prune API に DB2_PRUNE_OPTION_FORCE オプションを指定して呼び出す

「do_not_delete」とマーク付けされた項目は、これら 3 つのアクションのいずれかを実行しない限り、リカバリー履歴ファイルから整理されません。

データベース・マネージャーは、リカバリー履歴ファイル項目の状況を「do_not_delete」に設定しません。開発者は、**UPDATE HISTORY** コマンドを使用してリカバリー履歴ファイル項目の状況を「do_not_delete」に設定できます。

「delete_pending」とマーク付けされた項目

「pending_delete」とマーク付けされた項目は、削除過程にあります。この項目は、整理操作を途中で終了した場合に、残ることがあります。このケースでは、その項目に関連付けられたファイルはまだ存在する場合とそうでない場合があり、(削除された項目と同様に) あたかも存在しないかのように扱われます。

さまざまな状況のリカバリー履歴ファイル項目の、その他の例を以下に示します。

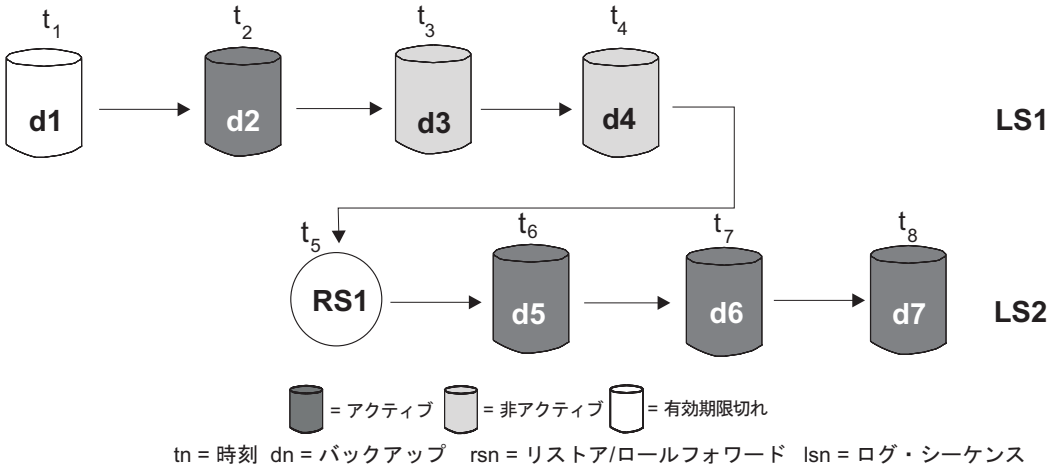


図 16. アクティブ、非アクティブ、有効期限切れのデータベース・バックアップの混合

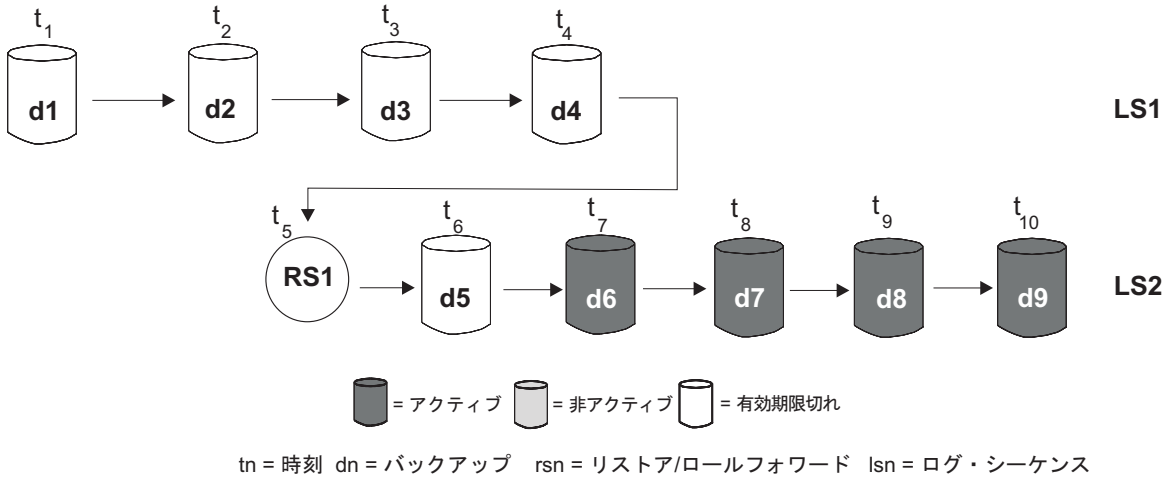


図 17. 有効期限が切れたログ・シーケンス

DB_HISTORY 管理ビューを使用したリカバリー履歴ファイル項目の表示

DB_HISTORY() 管理ビューを使用して、データベース履歴ファイルの内容にアクセスすることができます。これは、**LIST HISTORY** CLP コマンドまたは C 履歴 API を使用する方法に代わるものです。

始める前に

この関数を使用するにはデータベース接続が必要です。

このタスクについて

データベース履歴ファイルの削除と更新は、**PRUNE HISTORY** または **UPDATE HISTORY** コマンドによってのみ行えます。

この管理ビューは、DB2 Universal Database バージョン 8.2 以前を使用して作成されたデータベースでは使用できません。

手順

SQL SELECT ステートメント内で DB_HISTORY() 管理ビューを使用して、接続先データベースのデータベース履歴ファイルか、または **DB2NODE** 環境変数によって指定されたデータベース・パーティション上のデータベース履歴ファイルにアクセスします。 例えば、履歴ファイルの内容を表示するには、以下を使用します。

```
SELECT * FROM TABLE(DB_HISTORY()) AS LIST_HISTORY
```

例

管理ビューの構文を非表示にするには、次のようにしてビューを作成できます。

```
CREATE VIEW LIST_HISTORY AS
   SELECT * FROM TABLE(DB_HISTORY()) AS LIST_HISTORY
```

このビューを作成した後、ビューに対して照会を実行できます。例:

```
SELECT * FROM LIST_HISTORY
```

または

```
SELECT dbpartitionnum FROM LIST_HISTORY
```

または

```
SELECT dbpartitionnum, start_time, seqnum, tabname, sqlstate
  FROM LIST_HISTORY
```

リカバリー履歴ファイルのプルーニング

データベース・マネージャーはリカバリー履歴ファイル内にバックアップ操作、リストア操作、表スペース作成、その他のイベントに関する項目を作成します。関連付けられたリカバリー・オブジェクトがデータベースのリカバリーに不要になったためにリカバリー履歴ファイル内の項目が関連性がなくなっている場合、それらの項目をリカバリー履歴ファイルから除去または*整理*できます。

手順

以下のメソッドを使用して、リカバリー履歴ファイル内の項目を整理できます。

- **PRUNE HISTORY** コマンドを呼び出します。
- db2Prune API を呼び出します。
- ADMIN_CMD プロシージャーに PRUNE_HISTORY パラメーターを指定して呼び出します。

次のタスク

これらのメソッドの 1 つを使用してリカバリー履歴ファイルを整理すると、データベース・マネージャーはリカバリー履歴ファイルから指定したタイム・スタンプよりも古い項目を除去 (整理) します。

リカバリー履歴ファイル項目がプルーニング用に指定した基準に適合する場合、その項目がデータベースのリカバリーにまだ必要であれば、 **WITH FORCE** パラメーターまたは DB2PRUNE_OPTION_FORCE フラグを使用しない限りデータベース・マネージャーはその項目を整理しません。

AND DELETE パラメーターまたは DB2PRUNE_OPTION_DELETE フラグを使用する場合、整理された項目に関連付けられたログ・ファイルも同様に削除されます。

AUTO_DEL_REC_OBJ データベース構成パラメーターを ON に設定して、 **AND DELETE** パラメーターまたは DB2PRUNE_OPTION_DELETE フラグを使用する場合、整理された項目に関連付けられたログ・ファイル、バックアップ・イメージ、およびロード・コピー・イメージも同様に削除されます。

リカバリー履歴ファイルのプルーニングの自動化

リカバリー履歴ファイル項目の状況を自動的に整理および更新するように、データベース・マネージャーを構成できます。

UPDATE HISTORY コマンド、db2HistoryUpdate API、または ADMIN_CMD プロシージャーに "UPDATE_HISTORY" パラメーターを指定して使用することにより、リカバリー履歴ファイル内の項目の状況を手動で更新できます。 **PRUNE HISTORY** コマンド、db2Prune API、または ADMIN_CMD プロシージャーに "PRUNE_HISTORY" パラメーターを指定して使用することにより、リカバリー履歴ファイル内の項目を手動で除去または整理できます。ただし、リカバリー履歴ファイルを手動で更新および整理する代わりに、データベース・マネージャーがリカバリー履歴ファイルを管理するように構成することが勧められています。

データベース・マネージャーは、以下のタイミングで、リカバリー履歴ファイルの項目を自動的に更新および整理します。

- 完全な (増分でない) データベース・バックアップ操作または完全な (増分でない) 表スペース操作が正常に完了した後。
- ロールフォワード操作不要のデータベース・リストア操作が正常に完了した後。
- データベース・ロールフォワード操作が正常に完了した後。

自動プルーニングの際に、データベース・マネージャーは以下の 2 つの操作を実行します。

1. リカバリー履歴ファイルの項目の状況を更新します。
2. 有効期限切れとなったリカバリー履歴ファイルの項目を整理します。

データベース・マネージャーは、以下の方法でリカバリー履歴ファイルの項目を更新します。

- 必要でなくなったアクティブ・データベース・バックアップのイメージには、すべて「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、有効期限が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。すべての関連する非アクティブ表スペース・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」とマークされます。

- アクティブのデータベース・バックアップ・イメージがリストアされるものの、履歴ファイルに記録されている最新のデータベース・バックアップではない場合には、これと同じログ・シーケンスに属する後続のデータベース・バックアップ・イメージは「非アクティブ」としてマークされます。
- 非アクティブのデータベース・バックアップ・イメージがリストアされた場合、現行のログ・シーケンスに属する非アクティブ・データベース・バックアップは、再び「アクティブ」としてマークされるようになります。現行のログ・シーケンスに入っていないすべてのアクティブなデータベース・バックアップ・イメージには、「非アクティブ」のマークが付けられます。
- 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) に対応していないデータベースまたは表スペース・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。

  現行のログ・シーケンスは、リストアされたデータベース・バックアップ・イメージと、処理済みのログ・ファイルにより判別されます。データベース・バックアップ・イメージがリストアされると、以後のデータベース・バックアップ・イメージはすべて非アクティブになります。リストアされたイメージは、新しくログ・チェーンを始めるためです。 (このことは、バックアップ・イメージをロールフォワードせずにリストアした場合も当てはまります。ロールフォワード操作を行うと、ログ・チェーン内の中断後に取られたデータベース・バックアップは、非アクティブのマークが付けられます。ロールフォワード・ユーティリティーにより、損傷した現行バックアップ・イメージを含むログ・シーケンスが調べられるので、古いデータベース・バックアップ・イメージをリストアする必要が生じることが考えられます。)
- 表スペースをリストアした後で、現行のログ・シーケンスを適用して、データベースの現行の状態に達することができない場合には、その表スペース・レベルのバックアップ・イメージは非アクティブになります。
- 状況が「do_not_delete」の項目は整理されず、それらに関連付けられたログ・ファイル、バックアップ・イメージ、およびロード・コピー・ロード・コピーは削除されません。
- データベースのアップグレード時に、履歴ファイル中のすべてのオンライン・データベース・バックアップ項目とすべてのオンラインまたはオフライン表スペース・バックアップ項目は、有効期限切れとマークされるので、自動再構築の際にこれらの項目は再構築に必要なイメージとして選択されません。ロード・コピー・イメージとログ・アーカイブ項目はリカバリーの目的で使用できないので、これらのタイプの項目も有効期限切れとマークされます。

以下のデータベース構成パラメーターは、データベース・マネージャーがどの項目を整理するかを制御します。

num_db_backups
データベースのために保持するデータベース・バックアップの数を指定します。

rec_his_retentn
バックアップの履歴情報を保持する日数を指定します。

auto_del_rec_obj
整理されるリカバリー履歴ファイルの項目に関連付けられたログ・ファイル、バ

ックアップ・イメージ、およびロード・コピー・イメージを、データベース・マネージャーが削除するかどうかを指定します。

データベース・マネージャーがリカバリー履歴ファイルを自動的に管理するように構成するには、以下の構成パラメーターを設定します。

- **num_db_backups**
- **rec_his_retentn**
- **auto_del_rec_obj**

「**auto_del_rec_obj**」が ON に設定されているとき、成功したデータベース・バックアップ項目の数が「**num_db_backups**」構成パラメーターよりも大きい場合には常に、データベース・マネージャーが「**rec_his_retentn**」よりも古いリカバリー履歴ファイルの項目を自動的に整理します。

リカバリー履歴ファイル項目が整理されないように保護する

リカバリー履歴ファイル項目の状況を「do_not_delete」に設定することにより、重要なリカバリー履歴ファイル項目が整理されないように、および関連付けられたリカバリー・オブジェクトが削除されないように保護できます。

このタスクについて

PRUNE HISTORY コマンド、PRUNE_HISTORY を指定した ADMIN_CMD プロシージャー、または db2Prune API を使用して、リカバリー履歴ファイル項目を除去 (整理) できます。 **AND DELETE** パラメーターを **PRUNE HISTORY** と共に使用するか、または DB2PRUNE_OPTION_DELETE フラグを db2Prune と共に使用する場合、**auto_del_rec_obj** データベース構成パラメーターが ON に設定されていれば、関連付けられたリカバリー・オブジェクトも物理的に削除されます。

データベース・マネージャーは、リカバリー履歴ファイル項目も自動プルーニングの一部として整理します。**auto_del_rec_obj** データベース構成パラメーターが ON に設定されている場合、データベース・マネージャーは整理される項目に関連付けられたリカバリー・オブジェクトを削除します。

手順

重要なリカバリー履歴ファイル項目および関連付けられたリカバリー・オブジェクトを保護するには、以下のようにします。

UPDATE HISTORY コマンド、db2HistoryUpdate API、または UPDATE_HISTORY を指定した ADMIN_CMD プロシージャーを使用して、重要なリカバリー・ファイル項目の状況を「do_no_delete」に設定します。
「do_not_delete」とマーク付けされた項目を整理する方法は、以下の 3 つです。

- **PRUNE HISTORY** コマンドに **WITH FORCE** オプションを指定して呼び出す
- ADMIN_CMD プロシージャーに **PRUNE HISTORY** および **WITH FORCE** オプションを指定して呼び出す
- db2Prune API に DB2_PRUNE_OPTION_FORCE **iOption** を指定して呼び出す

「do_not_delete」とマーク付けされた項目は、これら 3 つの手順のいずれかを実行しない限り、リカバリー履歴ファイルから整理されません。

制約事項:

- 「do_not_delete」に設定できるのは、バックアップ・イメージ、ロード・コピー・イメージ、およびログ・ファイルの状況だけです。
- バックアップ項目の状況は、ロード・コピー・イメージ、増分ではないバックアップ、またはバックアップ操作に関連したログ・ファイルには波及しません。特定のデータベース・バックアップ項目とその関連するログ・ファイル項目を保管する場合には、そのデータベース・バックアップ項目と関連する各ログ・ファイルの項目の状況を設定する必要があります。

第 9 章 リカバリー・オブジェクトの管理

データベースを定期的にバックアップすると、累積するデータベース・バックアップ・イメージがとても大きくなり、データベース・ログ、およびロード・コピー・イメージが大量になる場合があります。IBM Data Server データベース・マネージャーを使用すると、こうしたリカバリー・オブジェクトを簡単に管理できます。

このタスクについて

リカバリー・オブジェクトの保管には、かなり大量のストレージ・スペースが消費される可能性があります。次にバックアップ操作を行うと、古いリカバリー・オブジェクトはデータベースのリストアで不要になるのでそれらを削除できます。ただし、古いリカバリー・オブジェクトの除去には時間がかかる場合があります。また、古いリカバリー・オブジェクトを削除する際に、まだ必要なリカバリー・オブジェクトを知らないうちに損傷してしまう可能性があります。

手順

データベースのリストアに不要なリカバリー・オブジェクトをデータベース・マネージャーを使用して削除するには、以下の 2 とおりの方法があります。

- **PRUNE HISTORY** コマンドを **AND DELETE** パラメーターを指定して実行するか、db2Prune API を DB2PRUNE_OPTION_DELETE フラグを使用して呼び出します。
- 不要なリカバリー・オブジェクトを自動的に削除するように、データベース・マネージャーを構成できます。

PRUNE HISTORY コマンドまたは db2Prune API を使用するデータベース・リカバリー・オブジェクトの削除

auto_del_rec_obj データベース構成パラメーターおよび **PRUNE HISTORY** コマンドまたは db2Prune API を使用して、リカバリー・オブジェクトを削除できます。

このタスクについて

PRUNE HISTORY コマンドを呼び出すとき、または db2Prune API を呼び出すとき、IBM Data Server データベース・マネージャーは以下を行います。

- 状況が DB2HISTORY_STATUS_DO_NOT_DEL ではないリカバリー履歴ファイルから項目を整理します。

PRUNE HISTORY コマンドに **AND DELETE** パラメーターを指定して呼び出すか、または db2Prune API に DB2PRUNE_OPTION_DELETE フラグを指定して呼び出すとき、データベース・マネージャーは以下を行います。

- 指定したタイム・スタンプよりも古く、状況が DB2HISTORY_STATUS_DO_NOT_DEL ではないリカバリー履歴ファイルから項目を整理します。
- 整理される項目に関連付けられた物理ログ・ファイルを削除します。

auto_del_rec_obj データベース構成パラメーターを ON に設定した場合は、 **PRUNE HISTORY** コマンドに **AND DELETE** パラメーターを指定して呼び出すか、または db2Prune API に DB2PRUNE_OPTION_DELETE フラグを指定して呼び出すとき、データベース・マネージャーは以下を行います。

- 状況が DB2HISTORY_STATUS_DO_NOT_DEL ではないリカバリー履歴ファイルから項目を整理します。
- 整理される項目に関連付けられた物理ログ・ファイルを削除します。
- 整理される項目に関連付けられたバックアップ・イメージを削除します。
- 整理される項目に関連付けられたロード・コピー・イメージを削除します。

手順

不要なリカバリー・オブジェクトを削除するには、次のようにしてください。

1. **auto_del_rec_obj** データベース構成パラメーターを ON に設定します。
2. **PRUNE HISTORY** コマンドに **AND DELETE** パラメーターを指定して呼び出すか、または db2Prune API に DB2PRUNE_OPTION_DELETE フラグを指定して呼び出します。

データベース・リカバリー・オブジェクト管理の自動化

auto_del_rec_obj データベース構成パラメーターおよび自動リカバリー履歴ファイル・プルーニングを使用して、IBM Data Server データベース・マネージャーを、毎回のデータベースのフル・バックアップ操作の後に不要なリカバリー・オブジェクトを自動削除するように構成できます。

このタスクについて

毎回の正常な全体 (増分ではない) データベース・バックアップ操作の後に、データベース・マネージャーは **num_db_backup** および **rec_his_retentn** 構成パラメーターの値に従ってリカバリー履歴ファイルを整理します。

- リカバリー履歴ファイル内のデータベース・バックアップ項目の数が **num_db_backup** 構成パラメーターの値よりも多い場合、データベース・マネージャーは **rec_his_retentn** 構成パラメーターの値よりも古く状況が DB2HISTORY_STATUS_DO_NOT_DEL ではない項目をリカバリー履歴ファイルから整理します。

auto_del_rec_obj データベース構成パラメーターを ON に設定した場合は、データベース・マネージャーはリカバリー履歴ファイルから項目を整理することに加えて、以下を行います。

- 整理される項目に関連付けられた物理ログ・ファイルを削除します。
- 整理される項目に関連付けられたバックアップ・イメージを削除します。
- 整理される項目に関連付けられたロード・コピー・イメージを削除します。

現行のリカバリー履歴に対象となる使用可能なフル・データベース・バックアップ・イメージが (おそらく、いままでに取られたことがなく) 存在しない場合、**rec_his_retentn** で指定された時間範囲よりも古いイメージが削除されます。

ファイルがリカバリー履歴ファイルにリストされた場所になくなっているためにデータベース・マネージャーがファイルを削除できない場合、データベース・マネージャーは履歴項目を整理します。

データベース・マネージャーとストレージ・マネージャーまたはデバイスとの間の通信エラーのためにデータベース・マネージャーがファイルを削除できない場合、データベース・マネージャーは履歴ファイル項目を整理しません。エラーが解決されると、ファイルは次の自動整理の際に削除可能です。

手順

不要なリカバリー・オブジェクトを自動削除するようにデータベース・マネージャーを構成するには、次のようにしてください。

1. **auto_del_rec_obj** データベース構成パラメーターを ON に設定します。
2. **rec_his_retentn** および **num_db_backups** 構成パラメーターを設定して、自動リカバリー履歴ファイル・プルーニングを使用可能にします。

リカバリー・オブジェクトの削除に対する保護

自動リカバリー・オブジェクト管理によって、管理時間を節約できるとともにストレージ・スペースも抑えることができます。ただし、特定のリカバリー・オブジェクトが自動的に削除されないようにしたい場合もあります。関連するリカバリー履歴ファイル項目の状況を do_not_delete に設定すると、主要なリカバリー・オブジェクトが削除されないようにできます。

このタスクについて

auto_del_rec_obj データベース構成パラメーターを ON に設定すると、リカバリー・オブジェクトに関連付けられているリカバリー履歴ファイル項目が整理されると、そのリカバリー・オブジェクトは削除されます。リカバリー履歴ファイル項目は、以下のいずれかが生じると整理されます。

- **PRUNE HISTORY** コマンドを **AND DELETE** パラメーターを指定して呼び出す
- db2Prune API を DB2PRUNE_OPTION_DELETE フラグを使用して呼び出す
- データベース・マネージャーがリカバリー履歴ファイルを自動的に整理する。表スペースまたはデータベースの全バックアップが正常に行われると、その度にこれが生じます。

PRUNE HISTORY コマンドを実行する場合、db2Prune API を呼び出す場合、またはリカバリー履歴ファイルの項目を自動的に整理するようにデータベース・マネージャーを構成する場合であっても、do_not_delete としてマークを付けられた項目は整理されず、その関連するリカバリー・オブジェクトは削除されません。

制約事項

- 「do_not_delete」に設定できるのは、バックアップ・イメージ、ロード・コピー・イメージ、およびログ・ファイルの状況だけです。
- バックアップ項目の状況は、ログ・ファイル、ロード・コピー・イメージ、またはそのバックアップ操作に関連する非増分のバックアップには伝搬されません。特定のデータベース・バックアップ項目とその関連するログ・ファイル項目を保

管する場合には、そのデータベース・バックアップ項目と関連する各ログ・ファイルの項目の状況を設定する必要があります。

手順

UPDATE HISTORY コマンドを使用して、関連するリカバリー・ファイル項目の状況を do_no_delete に設定します。

スナップショットのバックアップ・オブジェクトの管理

db2acsutil コマンドを使用してスナップショットのバックアップ・オブジェクトを管理する必要があります。ファイル・システムのユーティリティーを使用してスナップショットのバックアップ・オブジェクトを移動したり削除したりしないでください。

始める前に

スナップショット・バックアップ操作およびリストア操作を実行するには、ご使用のストレージ・デバイス用の DB2 ACS API ドライバーが必要です。 IBM Data Server には、以下のストレージ・ハードウェアのための DB2 ACS API ドライバーが組み込まれています。

- IBM TotalStorage SAN ボリューム・コントローラー
- IBM System Storage® DS6000™
- IBM System Storage DS8000®
- IBM System Storage N シリーズ
- NetApp V-series
- NetApp FAS series

スナップショットのバックアップ・オブジェクトを管理する前に、DB2 拡張コピー・サービス (ACS) を使用可能にする必要があります。 466 ページの『DB2 拡張コピー・サービス (ACS) の使用可能化』を参照してください。

制約事項

db2acsutil コマンドは、現行では AIX および Linux でのみサポートされています。

手順

1. 使用可能なスナップショットのバックアップ・オブジェクトをリストするには、**QUERY** パラメーターを使用します。

   例えば、dbminst1 というデータベース・マネージャー・インスタンスの使用可能なスナップショットのバックアップ・オブジェクトをリストするには、以下の構文を使用します。

   ```
   db2acsutil query instance dbminst1
   ```

2. 指定されたスナップショットのバックアップ操作の進行を確認するには、**STATUS** パラメーターを使用します。

例えば、database1 と呼ばれるデータベースで現在実行されているスナップショットのバックアップ操作の進行を確認するには、以下の構文を使用します。

```
db2acsutil query status db database1
```

3. 特定のスナップショットのバックアップ・オブジェクトを削除するには、**DELETE** パラメーターを使用します。

   例えば、10 日前の database1 と呼ばれるデータベースのすべてのスナップショットのバックアップ・オブジェクトを削除するには、以下の構文を使用します。

   ```
   db2acsutil delete older than 10 days ago db database1
   ```

第 10 章 リストア操作の進行状況をモニターする

LIST UTILITIES コマンドを使用して、データベースでのリストア操作をモニターできます。

手順

LIST UTILITIES コマンドを発行して、**SHOW DETAIL** パラメーターを指定します。

```
LIST UTILITIES SHOW DETAIL
```

タスクの結果

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

例

次に示すのは、リストア操作でのパフォーマンスをモニターするときの出力例です。

```
ID                                 = 6
Type                               = RESTORE
Database Name                      = SAMPLE
Partition Number                   = 0
Description                        = db
Start Time                         = 08/04/2011 12:24:47.494191
State                              = Executing
Invocation Type                    = User
Progress Monitoring:
      Completed Work               = 4096 bytes
      Start Time                   = 08/04/2011 12:24:47.494197
```

第 11 章 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、バックアップするデータベースの別名を指定することだけです。例えば、以下のようにします。

```
db2 backup db sample
```

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます: データベースのバックアップ. タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、およびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

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。 DB2 バージョン 9.5 以降で作成されたバックアップ・イメージは、ファイル・モード 600 で生成されます。つまり、UNIX ではインスタンスの所有者だけが読み取りおよび書き込み特権を持ち、Windows では DB2ADMNS (および管理者) グループのメンバーのみがバックアップ・イメージに対するアクセス権限を持ちます。

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

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

- ディレクトリー (ディスクまたはディスケットへのバックアップの場合)
- 装置 (テープへのバックアップの場合)
- Tivoli Storage Manager (TSM) サーバー
- 他のベンダーのサーバー

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

バックアップがファイルとして保管されている場合は、必要なくなった古いバックアップ・イメージを削除したい場合、そのファイルを削除することができます。その後 **BACKUP** オプションを指定して **LIST HISTORY** コマンドを実行すると、削除されたバックアップ・イメージに関する情報も戻されます。リカバリー履歴ファイルからこれらの項目を除去するには、**PRUNE** コマンドを使用しなければなりません。

リカバリー・オブジェクトが Tivoli Storage Manager (TSM) を使用して保存されている場合は、**db2adutl** ユーティリティーを使用して、リカバリー・オブジェクトの照会、抽出、検証、および削除を行うことができます。Linux および UNIX ではこのユーティリティーは sqllib/adsm ディレクトリーにあり、 Windows オペレーティング・システムでは sqllib¥bin にあります。スナップショット用には、sqllib/bin にある **db2acsutil** ユーティリティーを使用します。

すべてのオペレーティング・システムで、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.DBPART*nnn*.timestamp.Seq_num

例えば、以下のようにします。

STAFF.0.DB201.DBPART000.19950922120112.001

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 から 8 文字のデータベース別名。

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

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

データベース・パーティション番号
単一パーティション・データベース環境では、この値は常に DBPART000 です。パーティション・データベース環境では、この値は DBPART*xxx* です。*xxx* は、db2nodes.cfg ファイル中でデータベース・パーティション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 から 9999)
- *mm* は月 (01 から 12)
- *dd* は日 (01 から 31)
- *hh* は時 (00 から 23)
- *nn* は分 (00 から 59)
- *ss* は秒 (00 から 59)

シーケンス番号
ファイル拡張子として使用する 3 桁の番号。

バックアップ・イメージをテープに書き込む際には、以下のようになります。

- ファイル名は作成されません。しかし、検査するために、前述の情報がバックアップ・ヘッダーに保管されます。
- テープ装置が標準オペレーティング・システム・インターフェースを介して使用可能でなければなりません。しかし、大規模なパーティション・データベース環境では、テープ装置を各データベース・パーティション・サーバーに専用接続す

ることは実際的でないことがあります。この場合は、複数のテープ装置を 1 つまたは複数の TSM サーバーに接続することができます。これは、これらのテープ装置に対するアクセスを、各データベース・パーティション・サーバーから可能にするためです。

- パーティション・データベース環境では、REELlibrarian 4.2 または CLIO/S などの仮想テープ装置機能を提供している製品を使用することもできます。これらの製品を使用し、疑似テープ装置を介して他のノード (データベース・パーティション・サーバー) に接続されているテープ装置にアクセスすることができます。リモート・テープ装置へのアクセスはユーザーが意識せずに実行され、標準オペレーティング・システム・インターフェースを介して疑似テープ装置にアクセスできます。

標準またはバックアップ・ペンディング状態になっていないデータベースのバックアップをとることはできません。標準またはバックアップ・ペンディング状態になっている表スペースのバックアップをとることはできます。表スペースが標準またはバックアップ・ペンディング状態になっていない場合は、バックアップは許可される場合もあれば許可されない場合もあります。

同じ表スペースに対する並行バックアップ操作は許可されていません。特定の表スペースに対してバックアップ操作が開始されたら、それ以降の操作は失敗します (SQL2048N)。

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

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

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

バックアップ情報の表示

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

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

データのバックアップ

BACKUP DATABASE コマンドを使用してデータベースのデータのコピーを取り、別のメディアに保管します。このバックアップ・データは、元のデータに障害や損傷が発生した場合に使用できます。データベース全体、またはデータベース・パーティションをバックアップすることもでき、選択された表スペースのみをバックアップすることもできます。

始める前に

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する必要はありません。指定したデータベースへの接続はデータベース・バックアップ・ユーティリティーにより自動的に確立され、この接続はバックアップ操作が完了すると終了します。バックアップする予定のデータベースに接続している場合、接続を切断してから **BACKUP DATABASE** コマンドを発行し、バックアップ操作を進めます。

データベースは、ローカルとリモートのいずれかです。 Tivoli Storage Manager (TSM) または DB2 拡張コピー・サービス (ACS) 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。

オフライン・バックアップを実行する予定であり、**ACTIVATE DATABASE** コマンドを使用してデータベースを活動化した場合は、オフライン・バックアップを実行する前にデータベースを非活動化する必要があります。データベースへのアクティブな接続がある場合、データベースの非活動化を成功させるには、SYSADM 権限を持つユーザーがデータベースに接続し、以下のコマンドを発行する必要があります。

```
CONNECT TO database-alias
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
TERMINATE;
DEACTIVATE DATABASE database-alias
```

パーティション・データベース環境では **BACKUP DATABASE** コマンドを使ってデータベース・パーティションを個別にバックアップできます。または、**ON DBPARTITIONNUM** コマンド・パラメーターを使っていくつかのデータベース・パーティションを一度にバックアップしたり、**ALL DBPARTITIONNUMS** パラメーターを使ってすべてのデータベース・パーティションを同時にバックアップすることができます。 **LIST DBPARTITIONNUMS** コマンドを使用すると、バックアップ対象となるユーザー表を含んでいるデータベース・パーティションを識別できます。

シングル・システム・ビュー (SSV) バックアップを使用しておらず、パーティション・データベース環境でオフライン・バックアップを実行する場合には、他のすべてのデータベース・パーティションとは別にカタログ・パーティションのバックアップを取る必要があります。例えば、最初にカタログ・パーティションのバックアップを取り、次に、他のデータベース・パーティションのバックアップを取ることができます。このアクションが必要な理由は、バックアップ操作の際にカタログ・パーティションに対する排他的データベース接続が必要な場合があり、その間は他のデータベース・パーティションは接続できないからです。オンライン・バックアップを実行する場合は、すべてのデータベース・パーティション (カタログ・パーティションを含む) のバックアップを同時に取ったり、任意の順序で取ったりできます。

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

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

制約事項

バックアップ・ユーティリティーには、以下の制限が適用されます。

- 別々の表スペースであっても、表スペースのバックアップ操作と表スペースのリストア操作とを同時に実行することはできません。
- パーティション・データベース環境でロールフォワード・リカバリーを使用できるようにする場合は、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す。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ノードでも)。データベースに関するユーザー・データを含んでいないデータベース・パーティション・サーバーで、データベース・パーティションのバックアップ・イメージが必要となるのは、次の 2 つの場合です。
  - 最後のバックアップを作成した後にデータベース・システムにデータベース・パーティション・サーバーを追加し、このデータベース・パーティション・サーバーについて順方向リカバリーを実行する必要がある場合。
  - 特定時点のリカバリーを使用する場合。この場合は、システム内のすべてのデータベース・パーティションがロールフォワード・ペンディング状態でなければなりません。
- DMS 表スペースのオンライン・バックアップ操作は、以下の操作との互換はありません。
  - ロード
  - 再編成 (オンラインおよびオフライン)
  - 表スペースのドロップ
  - 表の切り捨て
  - 索引の作成
  - NOT LOGGED INITIALLY (CREATE TABLE および ALTER TABLE ステートメントと共に使用)
- 現在アクティブになっているデータベースのオフライン・バックアップを実行しようとすると、エラーを受け取ります。オフライン・バックアップを実行する前に、**DEACTIVATE DATABASE** コマンドを発行すると、データベースがアクティブ状態ではないことを確認できます。

手順

バックアップ・ユーティリティーを起動するには、次のようにします。

- コマンド行プロセッサー (CLP) で **BACKUP DATABASE** コマンドを発行します。

- BACKUP DATABASE パラメーターを使って ADMIN_CMD プロシージャーを実行します。
- db2Backup アプリケーション・プログラミング・インターフェース (API) を使用します。
- **BACKUP DATABASE** コマンドに関する IBM Data Studio のタスク・アシストを開きます。

例

以下は、CLP を介して発行される **BACKUP DATABASE** コマンドの例です。

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

次のタスク

オフライン・バックアップを実行した場合は、バックアップの完了後に、次のようにしてデータベースを再び活動化させる必要があります。

```
ACTIVATE DATABASE sample
```

関連情報:

スナップショットのバックアップの実行

スナップショットのバックアップ操作は、ストレージ・デバイスの高速コピー・テクノロジーを使用して、バックアップのデータのコピー部分を実行します。

始める前に

スナップショット・バックアップ操作およびリストア操作を実行するには、ご使用のストレージ・デバイス用の DB2 ACS API ドライバーが必要です。 IBM Data Server には、以下のストレージ・ハードウェアのための DB2 ACS API ドライバーが組み込まれています。

- IBM TotalStorage SAN ボリューム・コントローラー
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N シリーズ
- NetApp V-series
- NetApp FAS series

スナップショットのバックアップを実行する前に、DB2 拡張コピー・サービス (ACS) を使用可能にする必要があります。 466 ページの『DB2 拡張コピー・サービス (ACS) の使用可能化』を参照してください。

制約事項

スナップショット・バックアップを使用して、個々の表スペースをリカバリーすることはできません。

統合されたスナップショット・バックアップを使用して、リダイレクト・リストアを実行することはできません。 FlashCopy® リストアは、すべてのデータベース・

パスを含む完全なボリューム・グループのセットを、以前のポイント・イン・タイムに戻します。

手順

スナップショット・バックアップを実行するには、以下のいずれかの方法を使用してください。

- 次の例のように、**BACKUP DATABASE** コマンドに **USE SNAPSHOT** パラメーターを付けて発行する。

```
db2 backup db sample use snapshot
```

- 次の例に示すように、**BACKUP DB** および **USE SNAPSHOT** パラメーターを設定して ADMIN_CMD プロシージャーを呼び出す。

```
CALL SYSPROC.ADMIN_CMD
   ('backup db sample use snapshot')
```

- 次の例に示すように、SQLU_SNAPSHOT_MEDIA メディア・タイプを設定して db2Backup API を発行する。

```
int sampleBackupFunction( char dbAlias[],
                          char user[],
                          char pswd[],
                          char workingPath[] )
{
  db2MediaListStruct mediaListStruct = { 0 };

  mediaListStruct.locations = &workingPath;
  mediaListStruct.numLocations = 1;
  mediaListStruct.locationType = SQLU_SNAPSHOT_MEDIA;


  db2BackupStruct backupStruct = { 0 };

  backupStruct.piDBAlias = dbAlias;
  backupStruct.piUsername = user;
  backupStruct.piPassword = pswd;
  backupStruct.piVendorOptions = NULL;
  backupStruct.piMediaList = &mediaListStruct;

  db2Backup(db2Version950, &backupStruct, &sqlca);

  return 0;
}
```

スプリット・ミラーをバックアップ・イメージとして使用する

DB2 pureScale 環境以外でバックアップ・イメージとして使用するために、同一システム上の別の場所にデータベースのスプリット・ミラーを作成するには、以下の手順に従ってください。この手順は、データベースでデータベース・バックアップ操作を実行する代わりに使用できます。

手順

スプリット・ミラーをバックアップ・イメージとして使用するには、以下を実行します。

1. 次のコマンドを使用して、1 次データベースに対する入出力書き込み操作をサスペンドします。

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

   注: データベースが中断状態のときは、他のユーティリティーやツールを実行しないようにしてください。データベースのコピーを作成するだけにしてください。オプションで、**SET WRITE** SUSPEND を発行する前にすべてのバッファー・プールをフラッシュして、リカバリー・ウィンドウを最小化することができます。それには FLUSH BUFFERPOOLS ALL ステートメントを使用します。
2. 適切なオペレーティング・システム・レベルおよびストレージ・レベルのコマンドを使用して、 1 次データベースから 1 つまたは複数のスプリット・ミラーを作成します。

   注: ボリューム・ディレクトリーを含め、データベース・ディレクトリー全体をコピーするようにしてください。さらに、データベース・ディレクトリー外にある、ログ・ディレクトリーおよびコンテナー・ディレクトリーもコピーする必要があります。複数のストレージ・グループを使用している場合は、すべてのパスを、それらのパスのファイルおよびサブディレクトリーを含めて、コピーする必要があります。この情報を収集するには、DBPATHS 管理ビューを使用してください。このビューは、スプリット・ミラーを作成する必要のあるデータベースのすべてのファイルとディレクトリーを表示します。
3. 次のコマンドを使用して、1 次データベース上で入出力書き込み操作を再開します。

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

システムに障害が発生した場合は、次の手順を実行して、スプリット・ミラー・データベースをバックアップとして使用してデータベースをリストアしてください。

1. 次のコマンドを使用して、データベース・インスタンスを停止します。

   ```
   db2stop
   ```
2. スプリットされたデータを、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コピーします。

   重要: その際、スプリットされたログ・ファイルはコピーしないでください。オリジナルのログがロールフォワード・リカバリーに必要です。
3. 次のコマンドを使用して、データベース・インスタンスを開始します。

   ```
   db2start
   ```
4. 1 次データベースを初期化します。

   ```
   db2inidb database_alias as mirror
   ```

   ここで *database_alias* は、データベース別名を表しています。
5. データベースを、to end of logs オプションで、あるいは、to point-in-time にてポイント・イン・タイム指定で、ロールフォワードし、stop オプションで、ロールフォワードを完了させます。

DB2 pureScale 環境でスプリット・ミラーをバックアップ・イメージとして使用する

DB2 pureScale環境でバックアップ・イメージとして使用するために、同一システム上の別の場所にデータベースのスプリット・ミラーを作成するには、以下の手順に従ってください。この手順は、データベースでデータベース・バックアップ操作を実行する代わりに使用できます。

手順

スプリット・ミラーをバックアップ・イメージとして使用するには、以下を実行します。

1. 1 次クラスターから設定を取り出してインポートすることにより、2 次クラスター上に General Parallel File System (GPFS) を構成します。1 次クラスター上で、以下の GPFSコマンドを実行します。

   ```
   mmfsctl filesystem syncFSconfig -n remotenodefile
   ```

   ここで、*remotenodefile* は、2 次クラスター内のホストのリストです。
2. 次のコマンドを使用して、1 次データベースに対する入出力書き込み操作をサスペンドします。

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

   注: データベースが中断状態のときは、他のユーティリティーやツールを実行しないようにしてください。データベースのコピーを作成するだけにしてください。オプションで、**SET WRITE SUSPEND** を発行する前にすべてのバッファー・プールをフラッシュして、リカバリー・ウィンドウを最小化することができます。それには FLUSH BUFFERPOOLS ALL ステートメントを使用します。
3. 次のコマンドを使用して、サスペンドしてコピーする必要のあるファイル・システムを判断します。

   ```
   db2cluster -cfs -list -filesystem
   ```
4. 次のコマンドを使用して、コンテナー・データまたはログ・データを含む各 GPFS ファイル・システムをサスペンドします。

   ```
   /usr/lpp/mmfs/bin/mmfsctl filesystem suspend-write
   ```

   ここで *filesystem* は、データまたはログ・データを含むファイル・システムを表しています。

   注: GPFS ファイル・システムがサスペンドされている間は、読み取りおよび書き込み操作の両方がブロックされます。読み取り操作がブロックされる時間を最小限に抑えるには、この間にスプリット・ミラー操作以外は実行すべきではありません。
5. 適切なオペレーティング・システム・レベルおよびストレージ・レベルのコマンドを使用して、 1 次データベースから 1 つまたは複数のスプリット・ミラーを作成します。

   注: ボリューム・ディレクトリーを含め、データベース・ディレクトリー全体をコピーするようにしてください。さらに、データベース・ディレクトリー外にあるログ・ディレクトリー (すべてのログ・ストリーム・サブディレクトリーも含む)、およびコンテナー・ディレクトリーもコピーする必要があります。

6. サスペンドした各 GPFS ファイル・システムに対して次のコマンドを使用して、ファイル・システムをサスペンド状態から再開します。

   ```
   /usr/lpp/mmfs/bin/mmfsctl filesystem resume
   ```

   ここで *filesystem* は、データまたはログ・データを含む、サスペンドされたファイル・システムを表しています。
7. 次のコマンドを使用して、1 次データベースに対する入出力書き込み操作を再開します。

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

スプリット・ミラーをバックアップ・イメージとして使用してデータベースをリストアする必要がある場合には、次の手順を実行してください。

1. 次のコマンドを使用して、1 次データベース・インスタンスを停止します。

   ```
   db2stop
   ```
2. 次のコマンドを使用して、クラスター・マネージャーのドメインをリストします。

   ```
   db2cluster -cm -list -domain
   ```
3. 次のコマンドを使用して、各ホスト上のクラスター・マネージャーを停止します。

   ```
   db2cluster -cm -stop -host host -force
   ```

   注: このコマンドを発行するホストを最後にシャットダウンする必要があります。
4. 次のコマンドを使用して、1 次データベースで GPFS クラスターを停止します。

   ```
   db2cluster -cfs -stop -all
   ```
5. 適切なオペレーティング・システム・レベルのコマンドを使用して、スプリットしたデータを、1 次データベースからコピーします。

   重要: その際、スプリットされたログ・ファイルはコピーしないでください。オリジナルのログがロールフォワード・リカバリーに必要です。
6. 次のコマンドを使用して、1 次データベース・インスタンス上の GPFS クラスターを開始します。

   ```
   db2cluster -cfs -start -all
   ```
7. 次のコマンドを使用して、クラスター・マネージャーを開始します。

   ```
   db2cluster -cm -start -domain domain
   ```
8. 次のコマンドを使用して、データベース・インスタンスを開始します。

   ```
   db2start
   ```
9. 次のコマンドを使用して、1 次データベースを初期化します。

   ```
   db2inidb database_alias as mirror
   ```
10. 1 次データベースを、to end of logs オプションで、あるいは、to point-in-time にてポイント・イン・タイム指定で、ロールフォワードし、stop オプションで、ロールフォワードを完了させます。

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 (例えば、AIX でブロック・サイズがゼロに設定されている場合など) に当てはまります。

バックアップ時に使用できる固定ブロック・サイズ数には制限があります。このような制限があるのは、DB2 データベース・システムがバックアップ・イメージ・ヘッダーを 4 KB ブロックとして書き出すためです。 DB2 データベース・システム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 バイトのみです。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 データベース・システムでサポートされるどのサイズでもない場合、バックアップ操作が正常に完了しないことがあります。

データベースが巨大な場合、固定ブロック・サイズを使用すると、バックアップ操作の完了に、予期される以上の時間がかかる可能性があります。パフォーマンスを改善するため、可変長ブロック・サイズを使用することができます。

注: 可変長ブロック・サイズを使用する場合は、正常にリカバリーできるような、十分にテストされた手順に従ってください。これには、可変長ブロック・サイズを使って作成されるバックアップ・イメージを使用し、**BACKUP** および **RESTORE** コマンドでバッファー・サイズを明示的に指定する操作が含まれます。

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

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

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

各部分の定義は次のとおりです。

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

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

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

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

これにより、`backup_filename.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。

デバイス	アタッチ	ブロック・サイズの上限	DB2 バッファー・サイズの上限 (4 KB ページ単位)
8 mm	scsi	131,072	32
3420	s370	65,536	16
3480	s370	61 440	15
3490	s370	61 440	15
3490E	s370	65,536	16
7332 (4 mm)[1]	scsi	262,144	64
3490e	scsi	262,144	64
3590[2]	scsi	2,097,152	512
3570 (Magstar MP)		262,144	64

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。

テープ装置の互換性の検査

UNIX、Linux、および AIX オペレーティング・システムの場合のみ、ご使用のテープ装置による DB2 データベースのバックアップがサポートされているかどうかを判別するには、以下の手順を実行します。

データベース・マネージャー・インスタンス所有者としてオペレーティング・システム・コマンド **dd** を実行して、テープ装置に対して読み取りまたは書き込みを行います。**dd** コマンドが成功した場合は、そのテープ装置を使用して DB2 データベースのバックアップをとることができます。

Named PIPE へのバックアップ

UNIX ベースのシステムでは、ローカル Named PIPE 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 Named PIPE からデータベースをリストアする) ためのサポートを使用できるようになりました。

始める前に

Named PIPE 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。パイプはローカル・ファイル・システム上に存在する必要があります。 Named PIPE はローカル装置として扱われるので、宛先として Named PIPE を指定する必要はありません。

手順

1. Named PIPE を作成します。 以下に AIX の場合の例を示します。

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

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

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

3. データベースのバックアップ操作の宛先としてこのパイプを使用します。

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

パーティション・データベースのバックアップ

パーティション・データベース環境でデータベースのバックアップを行うには、難しい問題が関係しています。例えば、各データベース・パーティションで正常に行われたバックアップのトラッキング、複数のログ・ファイルとバックアップ・イメージの管理、およびすべてのデータベース・パーティションにおけるログ・ファイルとバックアップ・イメージをデータベースにリストアするために必要なリカバリー時間を最短にすることなどです。シングル・システム・ビュー (SSV) バックアップを使用するのが、パーティション・データベースをバックアップする最も簡単な方法です。

このタスクについて

パーティション・データベース環境でデータベースをバックアップするには、以下の 3 つの方法が考えられます。

- **BACKUP DATABASE** コマンド、ADMIN_CMD プロシージャーを使用する **BACKUP DATABASE** コマンド、または db2Backup API 関数を使用して、一度に 1 つずつデータベース・パーティションをバックアップする
- **db2_all** コマンドを **BACKUP DATABASE** コマンドと共に使用して、指定のすべてのデータベース・パーティションをバックアップする

- シングル・システム・ビュー (SSV) バックアップを実行して、データベース・パーティションすべて、またはその一部を同時にバックアップする
- IBM Data Studio のタスク・アシストを使用して、データベースのバックアップ・プロセスをガイドする

一度に 1 つずつデータベース・パーティションをバックアップすると時間がかかり、エラーも起こりやすくなります。**db2_all** コマンドを使ってすべてのパーティションをバックアップする方法の場合、通常は 1 つのコマンド呼び出しだけが必要であるため、データベース・パーティションを個々にバックアップする方法に比べて簡単です。ただし **db2_all** を使ってパーティション・データベースをバックアップする場合、カタログが含まれるデータベース・パーティションと非カタログ・データベース・パーティションを同時にバックアップすることはできないので、やはり何度か **db2_all** を呼び出す必要が生じることがあります。データベース・パーティションを一度に 1 つずつバックアップする場合でも、**db2_all** を使用する場合でも、こうした方法で作成されたバックアップ・イメージを管理することは困難です。それぞれのデータベース・パーティションのバックアップ・イメージのタイム・スタンプが異なることに加えて、複数のデータベース・パーティションのバックアップ・イメージのリカバリー時間が最短になるように調整するのも難しいためです。

前述のような理由で、パーティション・データベース環境でデータベースのバックアップを行うために推奨されている方法は、SSV バックアップを使用することです。

手順

SSV バックアップを使用して、パーティション・データベースのデータベース・パーティションすべて、またはその一部をバックアップするには、以下のようにします。

1. オプション: オンラインのままであることをデータベースに許可するか、データベースをオフラインにします。

   データベースがオンラインであってもオフラインであっても、パーティション・データベースをバックアップできます。データベースがオンラインの場合、バックアップ・ユーティリティーは他のデータベース・パーティションに対する共有接続を取得するので、ユーザー・アプリケーションはバックアップ中にもデータベース・パーティションに接続できます。
2. データベース・カタログが含まれるデータベース・パーティションで、適切なパラメーターを指定してパーティション・データベースに対してバックアップを実行します。
   - **ON DBPARTITIONNUMS** パラメーター指定して **BACKUP DATABASE** コマンドを実行します。
   - ADMIN_CMD プロシージャーを使用して **ON DBPARTITIONNUMS** パラメーターを指定して **BACKUP DATABASE** コマンドを実行します。
   - **iAllNodeFlag** パラメーターを指定して db2Backup API を呼び出します。
   - IBM Data Studio で、**BACKUP DATABASE** コマンドのタスク・アシストを開きます。

3. オプション: リカバリーに必要なログ・ファイルをバックアップ・イメージに組み込んでください。

   SSV バックアップを実行する場合 (つまり **ON DBPARTITIONNUM** パラメーターを指定する場合)、デフォルトでは、バックアップ・イメージにログ・ファイルが含まれます。バックアップ・イメージにログ・ファイルを含めないようにするには、バックアップ実行時に **EXCLUDE LOGS** コマンド・パラメーターを使用してください。 SSV 以外のバックアップの場合、デフォルトでは、ログ・ファイルはバックアップ・イメージから除外されます。

   詳しくは、 184 ページの『ログ・ファイルをバックアップ・イメージに含める』を参照してください。

4. オプション: 以前のバックアップ・イメージを削除します。 古いバックアップ・イメージを削除するために使用する方法は、バックアップ・イメージを保管する方法によって異なります。 例えば、バックアップ・イメージをディスクに保管した場合、そのファイルを削除できますし、Tivoli storage manager を使用してバックアップ・イメージを保管したのであれば、**db2adutl** ユーティリティーを使用してバックアップ・イメージを削除できます。DB2 拡張コピー・サービス (ACS) を使用する場合、**db2acsutil** を使用してスナップショットのバックアップ・オブジェクトを削除することができます。

関連情報:

IBM Tivoli Space Manager 階層ストレージ管理を使用したパーティション表のバックアップ

Tivoli Space Manager 階層ストレージ管理 (HSM) クライアント・プログラムは、適格なファイルを 2 次ストレージに自動的にマイグレーションして、ローカル・ファイル・システム上に特定のレベルのフリー・スペースを維持します。

表パーティションを使用した場合、表データはデータ・パーティションと呼ばれる複数のストレージ・オブジェクトに分割されます。HSM は、個別のデータ・パーティションの 2 次ストレージへのバックアップをサポートしています。

SMS 表スペースを使用する場合、各データ・パーティションの範囲は、対応するディレクトリー内のファイルとして表されます。このため、個別のデータの範囲 (データ・パーティション) を 2 次ストレージにマイグレーションすることはとても簡単です。

DMS 表スペースを使用する場合、各コンテナーがファイルとして表されます。この場合、アクセスの頻度が少ない範囲は、独自の表スペースに保管すべきです。EVERY 節を使用して CREATE TABLE ステートメントを発行するときに、NO CYCLE 節を使用して、表レベルの IN 節にリストされる表スペースの数が、作成されるデータ・パーティションの数と一致するようにしてください。これは以下の例で示されています。

例 1

```
CREATE TABLE t1 (c INT) IN tbsp1, tbsp2, tbsp3 NO CYCLE
  PARTITION BY RANGE(c)
  (STARTING FROM 2 ENDING AT 6 EVERY 2);
```

自動バックアップの使用可能化

データベースは、ハードウェアまたはソフトウェアのさまざまな障害のために使用不可能になることがあります。データベースの最新の全バックアップを持っておくことは、システムの災害時リカバリーを計画し、実装する上で不可欠です。災害時リカバリー計画の一部として自動データベース・バックアップを使用し、DB2 が適切かつ定期的にデータベースをバックアップできるようにします。

このタスクについて

自動バックアップは、コマンド行インターフェース、または AUTOMAINT_SET_POLICY システム・ストアード・プロシージャーを使用して構成できます。また、ヘルス・インディケーター db.db_backup_req を使用可能にする必要もありますが、これはデフォルトで使用可能です。アクティブ・データベースのみが評価の対象であることに注意してください。

手順

- コマンド行インターフェースを使って、自動バックアップを構成するには、次のデータベース構成パラメーターをそれぞれ ON に設定します。
  - **AUTO_MAINT**
  - **AUTO_DB_BACKUP**

  ヘルス・インディケーター db.db_backup_req を使用可能にするには、次のコマンドを発行します。

  ```
  db2 update alert configuration for database on database-alias
   using db.db_backup_req set THRESHOLDSCHECKED YES
  ```
- IBM Data Studio を使って自動バックアップを構成するには、データベースを右クリックし、タスク・アシストを選択して自動バックアップを構成します。
- AUTOMAINT_SET_POLICY システム・ストアード・プロシージャーを使って、自動バックアップを構成するには、次のようにします。
  1. バックアップ・メディア、バックアップをオンラインまたはオフラインのどちらで行うか、バックアップの頻度などの詳細を指定して、構成 XML 入力を作成します。

     SQLLIB/samples/automaintcfg ディレクトリー内の DB2DefaultAutoBackupPolicy.xml と呼ばれるサンプル・ファイルの内容をコピーして、その XML を構成要件に適合するように変更できます。
  2. オプション: 構成 XML 入力を含む XML 入力ファイルを作成します。
  3. 以下のパラメーターを指定して AUTOMAINT_SET_POLICY を呼び出します。
     - 保守タイプ: AutoBackup
     - 構成 XML 入力: 構成入力 XML テキストを含む BLOB、または構成 XML 入力を含むファイル名。

     AUTOMAINT_SET_POLICY システム・ストアード・プロシージャーの使用方法について詳しくは、 77 ページの『SYSPROC.AUTOMAINT_SET_POLICY または SYSPROC.AUTOMAINT_SET_POLICYFILE を使用した自動保守ポリシーの構成』のトピックを参照してください。

関連情報:

自動データベース・バックアップ

データベースは、ハードウェアまたはソフトウェアのさまざまな障害のために使用不可能になることがあります。自動データベース・バックアップの機能を使用すると、データベースの最新のフル・バックアップが必要に応じて確実に実行されるようになるため、 DBA がデータベース・バックアップを簡単に管理できるようになります。バックアップ操作の実行が必要かどうかは、次のことを基準にして判断されます。

- データベースのフル・バックアップを一度も実行したことがない
- 最後に実行したフル・バックアップ以降に経過した時間が、指定された時間を超えた
- 最後に実行されたバックアップ以降に消費されたトランザクション・ログ・スペースが指定された (4 KB) ページ数を超えた (アーカイブ・ロギング・モードの場合のみ)。

システムの災害時リカバリー・ストラテジーの計画を立ててインプリメントすることによって、データを保護してください。実際の必要によっては、自動データベース・バックアップ・フィーチャーを、バックアップとリカバリーのストラテジーの一部として含めることもできます。

データベースがロールフォワード・リカバリー (アーカイブ・ロギング) 対応なら、オンライン・バックアップかオフライン・バックアップのいずれかについて、自動データベース・バックアップを有効にすることができます。そうでない場合に利用できるのは、オフライン・バックアップだけです。自動データベース・バックアップでは、ディスク、テープ、 Tivoli Storage Manager (TSM)、およびベンダー固有の DLL メディア・タイプがサポートされています。

ディスクへのバックアップを選択した場合、自動バックアップ・フィーチャーにより、自動データベース・バックアップ構成で指定されたディレクトリーからバックアップ・イメージが定期的に削除されます。自動バックアップ・ポリシー・ファイルに指定されているフルバックアップの回数に関わらず、どの時点においても、最新のバックアップ・イメージだけが使用可能になります。そのディレクトリーは、自動バックアップ・フィーチャー専用にし、他のバックアップ・イメージを格納するためには使用しないようにすることをお勧めします。

自動データベース・バックアップ・フィーチャーを有効または無効にするには、**auto_db_backup** および **auto_maint** のデータベース構成パラメーターを使用します。パーティション・データベース環境においては、それらのデータベース構成パラメーターが有効になっているデータベース・パーティションのそれぞれに対して、自動データベース・バックアップが実行されます。

AUTOMAINT_SET_POLICY および AUTOMAINT_SET_POLICYFILE と呼ばれるシステムのストアード・プロシージャーのいずれか 1 つを使用して自動バックアップを構成することもできます。

バックアップ操作のモニター

LIST UTILITIES コマンドを使用して、データベースでのバックアップ操作の進行をモニターできます。

手順

LIST UTILITIES コマンドを発行して、**SHOW DETAIL** パラメーターを指定します。

```
list utilities show detail
```

タスクの結果

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

例

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

```
ID                                 = 3
Type                               = BACKUP
Database Name                      = SAMPLE
Partition Number                   = 0
Description                        = offline db
Start Time                         = 08/04/2011 12:16:23.248367
State                              = Executing
Invocation Type                    = User
Throttling:
   Priority                        = Unthrottled
Progress Monitoring:
   Extimated Percentage Complete = 31
      Total Work                   = 123147277 bytes
      Completed Work               = 37857269 bytes
      Start Time                   = 08/04/2011 12:16:23.248377
```

バックアップのパフォーマンスの最適化

バックアップ操作を実行すると、DB2 データベース・マネージャーは、バッファー数、バッファー・サイズ、および並列処理設定の最適値を自動的に選択します。この値は、使用可能なユーティリティー・ヒープ・メモリーの大きさ、使用可能なプロセッサー数、およびデータベース構成に基づきます。したがって、システムに使用可能なストレージの量によっては、**util_heap_sz** 構成パラメーターを増やして、より多くのメモリーを割り振ることを検討します。目的は、バックアップ操作の完了にかかる時間を最小限に抑えることです。次の **BACKUP DATABASE** コマンド・パラメーターの値を明示的に入力しないと、DB2 データベース・マネージャー側が値を選択します。

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

バッファーの数とバッファー・サイズを指定しないと、DB2 データベース・マネージャーにより値が設定されますが、大規模データベースでは影響は最小限になるはずです。しかし、小規模なデータベースの場合、かなりの比率でバックアップ・イメージ・サイズが大きくなることがあります。ディスクに書き込まれた最新のデータ・バッファーに含まれているデータが小さい場合でも、とにかくバッファー全体がイメージに書き込まれます。したがって、小規模なデータベースでは、相当なパーセンテージのイメージ・サイズが空になる可能性があることになります。

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

- 表スペース・バックアップを指定します。

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

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

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

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

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・マネージャーの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。

オンライン・バックアップと他のユーティリティーの互換性

一部のユーティリティーはオンライン・バックアップと同時に実行できますが、そうでないものもあります。

以下のユーティリティーはオンライン・バックアップと互換性があります。

- **EXPORT**
- ONLINE **INSPECT**

以下のユーティリティーは、特定の環境においてのみオンライン・バックアップと互換性があります。

- ONLINE CREATE INDEX

  SMS モードでは、ALTER TABLE ロックにより、オンライン索引作成とオンライン・バックアップは並行して実行されません。オンライン索引作成が ALTER TABLE ロックを排他モードで取得するのに対し、オンライン・バックアップは共有モードで取得します。

  DMS モードでは、ほとんどの場合、オンライン索引作成とオンライン・バックアップは並行して実行することができます。多数の索引を持っている場合、オンライン索引作成が、並行するオンライン・バックアップと競合するオンライン・バックアップ・ロックを内部で取得する可能性があります。

- ONLINE INDEX REORG

  オンライン索引作成と同様、SMS モードでは、オンライン索引再編成は、ALTER TABLE ロックにより、オンライン・バックアップと並行して実行されません。オンライン索引再編成が ALTER TABLE ロックを排他モードで取得するのに対し、オンライン・バックアップは共有モードで取得します。さらに、オンライン索引再編成操作では、切り替えフェーズの前に表が静止され、オンライン・バックアップを妨げる Z ロックが取得されます。しかし、Z 表ロックが取得される前にも、ALTER TABLE ロックのためにオンライン・バックアップは並行して実行できません。

  DMS モードでは、オンライン索引再編成とオンライン・バックアップは並行して実行することができます。

  さらに、オンライン索引再編成は、切り替えフェーズの前に表を静止し、オンライン・バックアップを妨げる Z ロックを取得します。

- REBALANCE

  オンライン・バックアップおよびリバランサーが並行して実行中である場合、オンライン・バックアップはリバランサーを一時停止し、それが完了するまで待機することはありません。

- **IMPORT**

REPLACE パラメーターを指定して **IMPORT** コマンドが発行された場合を除き、インポート・ユーティリティーはオンライン・バックアップと互換性があります。REPLACE オプションを指定して IMPORT コマンドが発行された場合、インポートは表に対する Z ロックを取得し、オンライン・バックアップを並行して実行できないようにします。

- **ALLOW READ ACCESS LOAD**

  COPY NO パラメーターを指定して **LOAD** コマンドが発行された場合、**ALLOW READ ACCESS** ロード操作はオンライン・バックアップと互換性がありません。このモードでは、両方のユーティリティーが表スペースの状態を変更し、ユーティリティーのいずれかがエラーを報告することになります。

  COPY YES オプションを指定して **LOAD** コマンドが発行された場合、**ALLOW READ ACCESS** ロード操作はオンライン・バックアップと互換性があります。しかし、互換性の問題がいくつか生じる可能性があります。SMS モードでは、ユーティリティーは並行して実行できますが、非互換の表ロック・モードが保持されるため、結果として、表ロックの待機となる可能性があります。 DMS モードでは、ユーティリティーは両方とも非互換の「Internal-B」(OLB) ロック・モードを保持するため、そのロックを待機する可能性があります。ユーティリティーが同じ表スペースで並行して実行すると、ロード・ユーティリティーはその続行前に、バックアップ・ユーティリティーが表スペースの処理を完了するのを待機しなければならない場合があります。

- ONLINE **REORG TABLE**

  オンライン表再編成のクリーンアップ・フェーズは、オンライン・バックアップが実行中の間は開始できません。必要に応じて、表再編成を一時停止し、オンライン・バックアップを完了させてからオンライン表再編成を再開することができます。

  DMS 表スペースのオンライン・バックアップは、その同じ表スペース内の表がオンラインで再編成中であるときに開始できます。切り捨てフェーズ中に、再編成操作に関連したロック待機が発生する可能性があります。

  SMS 表スペースのオンライン・バックアップは、その同じ表スペース内の表がオンラインで再編成中であるときには開始できません。両方の操作において排他ロックが必要になります。

- Z ロックを必要とする DDL (ALTER TABLE、DROP TABLE、および DROP INDEX)

  オンライン DMS 表スペース・バックアップは、Z ロックを必要とする DDL と互換性があります。

  オンライン SMS 表スペース・バックアップは、Z ロックが解放されるまで待機する必要があります。

- ストレージ・グループ DDL

  以下のステートメントのいずれかを発行してデータベースのストレージ・グループを変更する場合は、この操作をオンライン・バックアップのスケジュールと調整する際に注意する必要があります。

- CREATE STOGROUP
- ALTER STOGROUP
- DROP STOGROUP
- RENAME STOGROUP
- ALTER DATABASE

進行中のオンライン・バックアップがある場合、ストレージ・グループ DDL は、適切なロックを取得するまで、その操作の後ろで待機します。これには長時間かかる場合があります。同様に、オンライン・バックアップは、進行中のストレージ・グループ DDL があれば、その DDL がコミットまたはロールバックされるまで、その後ろで待機します。

- **RUNSTATS** (ALLOW WRITE および ALLOW READ)

  システム・カタログ表スペースが SMS 表スペースである場合を除き、**RUNSTATS** コマンドはオンライン・バックアップと同時実行できます。システム・カタログが SMS 表スペースにある場合これが行えないのは、**RUNSTATS** コマンドとオンライン・バックアップが表の非互換表ロックを保持し、ロック待機が発生するためです。

- ALTER TABLESPACE

  表スペースのオンライン・バックアップ中には、自動サイズ変更を使用可能にしたり使用不可にしたりする操作や、自動サイズ変更コンテナーを変更する操作はできません。

以下のユーティリティーはオンライン・バックアップと互換性がありません。

- **REORG TABLE**
- **RESTORE DATABASE**
- **ROLLFORWARD DATABASE**
- オンライン **BACKUP DATABASE**
- **LOAD ALLOW NO ACCESS**
- **SET WRITE**

バックアップの例

このトピックでは、さまざまなバックアップ戦略のいくつの例を示します。

TSM へのバックアップ

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

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

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

増分バックアップ

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

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

磁気テープへのバックアップ

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

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

第 12 章 リカバリーの概要

RECOVER ユーティリティーは、リカバリー履歴ファイルの情報に基づき、必要なリストアおよびロールフォワード操作を実行してデータベースを指定した時点までリカバリーします。

このユーティリティーを使用するときには、データベースを特定時点までまたはログ・ファイルの最後までリカバリーすることを指定します。そうすると、このユーティリティーは、最適なバックアップ・イメージを選択して、リカバリー操作を実行します。

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます: データベースのリカバリー. タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、およびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

RECOVER ユーティリティーは、以下の **RESTORE DATABASE** コマンド・オプションをサポートしません。

- **TABLESPACE** *tablespace-name*。表スペース・リストア操作は、サポートされていません。
- **INCREMENTAL**。増分リストア操作は、サポートされていません。
- **OPEN** *num-sessions* **SESSIONS** TSM または他のベンダー製品とともに使用する入出力セッションの数は指定できません。
- **BUFFER** *buffer-size*。リストア操作に使用するバッファーのサイズは設定できません。
- **DLREPORT** *filename*。リンク解除されるレポート・ファイルのファイル名は指定できません。
- **WITHOUT ROLLING FORWARD**。正常なリストア操作後にデータベースをロールフォワード・ペンディング状態にしないことを指定できません。
- **PARALLELISM** *n*。リストア操作の並列処理の度合いは指定できません。
- **WITHOUT PROMPTING**。リストア操作を無人で実行することは指定できません。

また、RECOVER ユーティリティーの使用時に **REBUILD** オプションを指定することはできません。しかし RECOVER ユーティリティーは、リカバリー履歴ファイルの情報に基づいてデータベース・バックアップ・イメージを検出できない場合に、該当する **REBUILD** オプションを自動的に使用します。

データのリカバリー

RECOVER DATABASE コマンドは、リカバリー履歴ファイル内の情報に基づき、指定した時点までデータベースおよび全ストレージ・グループをリカバリーします。

始める前に

ロールフォワード・フェーズ中に終了した不完全なリカバリー操作の後に **RECOVER DATABASE** コマンドを発行する場合、リカバリー・ユーティリティーはリストア・フェーズを再実行せずに前のリカバリー操作を続けようとします。リカバリー・ユーティリティーにリストア・フェーズの再実行を強制する場合は、**RESTART** オプションを指定して **RECOVER DATABASE** コマンドを発行し、リカバリー・ユーティリティーを強制して、前の完了できなかったリカバリー操作をすべて無視するようにします。アプリケーション・プログラミング・インターフェース (API) を使用している場合は、**iRecoverAction** フィールドに呼び出し元アクション DB2RECOVER_RESTART を指定し、リストア・フェーズを再実行するようにリカバリー・ユーティリティーを強制します。

リストア・フェーズ中に **RECOVER DATABASE** コマンドが中断する場合にはこれを継続できません。 **RECOVER DATABASE** コマンドを再発行する必要があります。

リカバリーを実行するデータベースに接続していてはなりません。データベース・リカバリー・ユーティリティーは、指定されたデータベースに自動的に接続を確立し、この接続はリカバリー操作が完了すると終了します。

このタスクについて

データベースは、ローカルとリモートのいずれかです。

注: パーティション・データベース環境では、リカバリー・ユーティリティーはデータベースのカタログ・パーティションから起動する必要があります。

手順

リカバリー・ユーティリティーを起動するには、以下を使用します。

- **RECOVER DATABASE** コマンド、または
- db2Recover アプリケーション・プログラミング・インターフェース (API)。

例

次の例は、CLP での **RECOVER DATABASE** コマンドの使用方法を示すものです。

```
db2 recover db sample
```

db2adutl を使用したデータのリカバリー

db2adutl コマンドと **logarchopt1** および **vendoropt** データベース構成パラメーターを使用して、ノード間リカバリーを行えます。このリカバリーについて、いくつかの異なる Tivoli Storage Manager (TSM) 環境における例で示します。

以下の例では、コンピューター 1 (bar という名前) において AIX オペレーティング・システムが実行されています。このマシンのユーザーは roecken です。 bar 上のデータベースは zample という名前です。コンピューター 2 は dps という名前です。このコンピューターでも AIX オペレーティング・システムが実行され、ユーザーは regress9 です。

例 1: TSM サーバーがパスワードを自動管理する (PASSWORDACCESS オプションを GENERATE に設定)

このノード間リカバリーの例では、ログ・アーカイブとバックアップが TSM サーバー上に保管され、PASSWORDACCESS=GENERATE オプションを使ってパスワードが管理されるような場合に、1 つのコンピューターから別のコンピューターにデータをリカバリーする目的で 2 つのコンピューターをセットアップする方法を示します。

注: データベース構成を更新した後に、データベースのオフライン・バックアップを取っておく必要があるかもしれません。

1. bar コンピューターのデータベースのログを TSM サーバーにアーカイブできるようにするには、以下のコマンドを使用して、zample データベースに関するデータベース構成パラメーター **logarchmeth1** を更新します。

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   次の情報が戻されます。

   ```
   DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
   ```

2. 以下のコマンドを使用して、すべてのユーザーとアプリケーションをデータベースから切断します。

   ```
   db2 force applications all
   ```

3. 以下のコマンドを使用して、データベースに接続しているアプリケーションが存在しないことを確認します。

   ```
   db2 list applications
   ```

   データが戻されなかったことを示すメッセージを受け取るはずです。

   注: パーティション・データベース環境では、すべてのデータベース・パーティションでこのステップを実行する必要があります。

4. 以下のコマンドを使用して、TSM サーバー上にデータベースのバックアップを作成します。

   ```
   db2 backup db zample use tsm
   ```

   次のような情報が戻されます。

   ```
   Backup successful. The timestamp for this backup imagge is : 20090216151025
   ```

   注: パーティション・データベース環境では、すべてのデータベース・パーティションでこのステップを実行する必要があります。データベース・パーティション上でこのステップを実行する順序は、オンライン・バックアップを実行するか、それともオフライン・バックアップを実行するかによって違います。詳しくは、 320 ページの『データのバックアップ』を参照してください。

5. 以下のコマンドを使用して、zample データベースに接続します。

   ```
   db2 connect to zample
   ```

6. データベースに関する新しいトランザクション・ログを生成します。そうするには、以下のコマンドを使って表を作成し、TSM サーバーにデータをロードします。

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader replace
      into employee copy yes use tsm
   ```

この例では、表の名前は employee であり、区切り文字で区切られている mr という名前の ASCII ファイルからデータをロードします。**COPY** YES オプションを指定することにより、ロードされるデータのコピーを作成します。**USE TSM** オプションは、TSM サーバーにデータのコピーを保管することを指定します。

注: COPY YES オプションを指定できるのは、データベースのロールフォワード・リカバリーが使用可能になっている場合だけです。つまり、**logarchmeth1** データベース構成パラメーターが USEREXIT、 LOGRETAIN、DISK、または TSM に設定されている必要があります。

進行状況を示すために、ロード・ユーティリティーは次のような一連のメッセージを戻します。

```
SQL3109N  The utility is beginning to load data from file "/home/roecken/mr".

SQL3500W  The utility is beginning the "LOAD" phase at time "02/16/2009
15:12:13.392633".

SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3110N  The utility has completed processing.  "1" rows were read from the
input file.

SQL3519W  Begin Load Consistency Point. Input record count = "1".

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the "LOAD" phase at time "02/16/2009
15:12:13.445718".


Number of rows read         = 1
Number of rows skipped      = 0
Number of rows loaded       = 1
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 1
```

7. データが表にロードされた後、zample データベースに対して以下のような照会を実行して、TSM サーバー上に 1 つのバックアップ・イメージ、1 つのロード・コピー・イメージ、および 1 つのログ・ファイルが存在することを確認します。

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
```

次の情報が戻されます。

```
Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
    Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE
```

```
Retrieving LOAD COPY information.
    1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

8. ノード間リカバリーを使用可能にするには、bar コンピューターに関連するオブジェクトへのアクセス権限を、別のコンピューターおよびアカウントに与える必要があります。この例では、以下のコマンドを使用して、コンピューター dps およびユーザー regress9 にアクセス権限を付与します。

```
bar:/home/roecken/sqllib/adsm> db2adutl grant user regress9
   on nodename dps for db zample
```

次の情報が戻されます。

```
Successfully added permissions for regress9 to access ZAMPLE on node dps.
```

注: db2adutl 付与操作の結果を確認するには、以下のコマンドを発行して、現在のノードでの現在のアクセス権限リストを取得することができます。

```
bar:/home/roecken/sqllib/adsm> db2adutl queryaccess
```

次の情報が戻されます。

```
Node                Username             Database Name   Type
-----------------------------------------------------------------
DPS                 regress9             ZAMPLE          A
-----------------------------------------------------------------
 Access Types:  B - backup images   L - logs   A - both
```

9. この例では、コンピューター 2 (つまり dps) は zample データベースのノード間リカバリー用にまだセットアップされていません。以下のコマンドを使用して、このユーザーおよびコンピューターに関連付けられたデータが TSM サーバー上に存在しないことを確認します。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
```

次の情報が戻されます。

```
  --- Database directory is empty ---
Warning: There are no file spaces created by DB2 on the ADSM server
Warning: No DB2 backup images found in ADSM for any alias.
```

10. 以下のコマンドを使用して、ユーザー roecken およびコンピューター bar に関連付けられた zample データベースのオブジェクトのリストを TSM サーバーに照会します。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename
     bar owner roecken
```

次の情報が戻されます。

```
--- Database directory is empty ---

Query for database ZAMPLE

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
    Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
```

```
  No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
    1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

この情報は既に生成された TSM 情報と一致するため、このイメージを dps コンピューターにリストアできることが確認されます。

11. 以下のコマンドを使用して、zample データベースを TSM サーバーから dps コンピューターにリストアします。

```
dps:/home/regress9> db2 restore db zample use tsm options
"'-fromnode=bar -fromowner=roecken'" without prompting
```

次の情報が戻されます。

```
DB20000I  The RESTORE DATABASE command completed successfully.
```

注: dps に zample データベースがすでに存在している場合は、 **OPTIONS** パラメーターを省略し、データベース構成パラメーター **vendoropt** を使用することになります。この構成パラメーターは、バックアップまたはリストア操作の **OPTIONS** パラメーターをオーバーライドします。

12. 新しい表を作成して新しいデータをロードしたときに zample データベース・ログ・ファイルに記録されたトランザクションを適用するために、ロールフォワード操作を実行します。この例では、次のようなロールフォワード操作の試行は失敗します。ユーザーとコンピューターの情報が指定されないため、ロールフォワード・ユーティリティーはログ・ファイルを見つけることができません。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次のようなエラーがコマンドによって戻されます。

```
SQL4970N  Roll-forward recovery on database "ZAMPLE" cannot reach the
specified stop point (end-of-log or point-in-time) because of missing log
file(s) on node(s) "0".
```

適切な **logarchopt** 値を使用して、別のコンピューターに関連付けられたログ・ファイルをロールフォワード・ユーティリティーに強制的に検索させます。この例では、次のようなコマンドを使って **logarchopt1** データベース構成パラメーターを設定し、ユーザー roecken およびコンピューター bar に関連付けられたログ・ファイルを検索します。

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
"'-fromnode=bar -fromowner=roecken'"
```

13. 以下のコマンドを使って **vendoropt** データベース構成パラメーターを設定することにより、ロールフォワード・ユーティリティーがバックアップおよびロード・コピー・イメージを使用できるようにします。

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

14. 以下のコマンドを使用して、zample データベース・ログ・ファイルに記録されたトランザクションを適用することにより、ノード間のデータ・リカバリーを完了できます。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次の情報が戻されます。

```
                                  Rollforward Status

 Input database alias                   = zample
 Number of members have returned status = 1


 Member number  Rollforward  Next log to  Log files processed        Last committed transaction
                status       be read
 -------------  -----------  -----------  -------------------------  ------------------------------
         0      not pending               S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC

DB20000I  The ROLLFORWARD command completed successfully.
```

ユーザー regress9 の下でのコンピューター dps 上のデータベース zample が、ユーザー roecken の下でのコンピューター bar 上のデータベースと同じポイントにリカバリーされました。

例 2: パスワードがユーザーによって管理される (PASSWORDACCESS オプションを PROMPT に設定)

このノード間リカバリーの例では、ログ・アーカイブとバックアップが TSM サーバー上に保管され、パスワードがユーザーによって管理されるような場合に、1 つのコンピューターから別のコンピューターにデータをリカバリーする目的で 2 つのコンピューターをセットアップする方法を示します。これらの環境では、余分の情報 (特に、オブジェクトの作成場所となったコンピューターの TSM ノード名とパスワード) が必要です。

1. コンピューター bar はソース・コンピューターの名前であるため、クライアント dsm.sys ファイルを更新して次のような行を追加します。

```
NODENAME bar
```

注: Windows オペレーティング・システムでは、このファイルの名前は dsm.opt です。このファイルを更新した場合、変更内容を有効にするにはシステムをリブートする必要があります。

2. 以下のコマンドを使用して、ユーザー roecken およびコンピューター bar に関連付けられたオブジェクトのリストを TSM サーバーに照会します。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename bar
owner roecken password *******
```

次の情報が戻されます。

```
Query for database ZAMPLE

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
    Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
```

```
  No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
    1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

3. zample データベースがコンピューター dps に存在しない場合は、以下の手順を実行します。
   a. 以下のコマンドを使用して、空の zample データベースを作成します。

      ```
      dps:/home/regress9> db2 create db zample
      ```

   b. 以下のコマンドを使用して、データベース構成パラメーター **tsm_nodename** を更新します。

      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
      ```

   c. 以下のコマンドを使用して、データベース構成パラメーター **tsm_password** を更新します。

      ```
      dps:/home/regress9> db2 update db cfg for zample using
      tsm_password ********
      ```

4. 以下のコマンドを使用して、zample データベースのリストアを試行します。

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

   リストア操作は正常に完了するものの、警告が出されます。

   ```
   SQL2540W  Restore is successful, however a warning "2523" was
   encountered during Database Restore while processing in No
   Interrupt mode.
   ```

5. 以下のコマンドを使用して、ロールフォワード操作を実行します。

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   この例では、リストア操作によってデータベース構成ファイルが置き換えられたためにロールフォワード・ユーティリティーが適切なログ・ファイルを見つけられず、次のようなエラー・メッセージが戻されます。

   ```
   SQL1268N  Roll-forward recovery stopped due to error "-2112880618"
   while retrieving log file "S0000000.LOG" for database "ZAMPLE" on node "0".
   ```

   以下の TSM データベース構成値を適切な値に再設定します。
   a. 以下のコマンドを使用して、**tsm_nodename** 構成パラメーターを設定します。

      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
      ```

   b. 以下のコマンドを使用して、**tsm_password** データベース構成パラメーターを設定します。

      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
      ```

   c. ロールフォワード・ユーティリティーが適切なログ・ファイルを見つけられるようにするために、以下のコマンドを使って **logarchopt1** データベース構成パラメーターを設定します。

      ```
      dps:/home/regress9> db2 update db cfg for zample using logarchopt1
      "'-fromnode=bar -fromowner=roecken'"
      ```

d. さらに、ロールフォワード操作中にロード・リカバリー・ファイルも使われるようにするために、以下のコマンドを使用して **vendoropt** データベース構成パラメーターを設定します。

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

6. 以下のコマンドを使ってロールフォワード操作を実行することにより、ノード間リカバリーを完了できます。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次の情報が戻されます。

```
                                  Rollforward Status

Input database alias                   = zample
Number of members have returned status = 1


Member number  Rollforward  Next log to  Log files processed        Last committed transaction
               status       be read
-------------  -----------  -----------  -------------------------  -------------------------------
        0      not pending               S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC


DB20000I  The ROLLFORWARD command completed successfully.
```

ユーザー regress9 の下でのコンピューター dps 上のデータベース zample が、ユーザー roecken の下でのコンピューター bar 上のデータベースと同じポイントにリカバリーされました。

例 3: クライアント・プロキシー・ノードを使用するよう TSM サーバーを構成する

このノード間リカバリーの例では、ログ・アーカイブとバックアップが TSM サーバー上に保管され、PASSWORDACCESS=GENERATE オプションを使ってパスワードが管理されるような場合に、1 つのコンピューターから別のコンピューターにデータをリカバリーする目的で 2 つのコンピューターをプロキシー・ノードとしてセットアップする方法を示します。

注: データベース構成を更新した後に、データベースのオフライン・バックアップを取っておく必要があるかもしれません。

この例では、コンピューター bar および dps がプロキシー名 clusternode の下で登録されます。これらのコンピューターはプロキシー・ノードとして既にセットアップされています。

1. 以下のコマンドを使用して、TSM サーバー上でコンピューター bar および dps をプロキシー・ノードとして登録します。

```
REGISTER NODE clusternode mypassword
GRANT PROXYNODE TARGET=clusternode AGENT=bar,dps
```

2. データベースのログを TSM サーバーにアーカイブできるようにするには、以下のコマンドを使用して、zample データベースに関するデータベース構成パラメーター **logarchmeth1** を更新します。

```
bar:/home/roecken> db2 update db cfg for zample using
   LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
```

次の情報が戻されます。

```
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

3. 以下のコマンドを使用して、すべてのユーザーとアプリケーションをデータベースから切断します。

   ```
   db2 force applications all
   ```

4. 以下のコマンドを使用して、データベースに接続しているアプリケーションが存在しないことを確認します。

   ```
   db2 list applications
   ```

   データが戻されなかったことを示すメッセージを受け取るはずです。

   注: パーティション・データベース環境では、すべてのデータベース・パーティションでこのステップを実行する必要があります。

5. 以下のコマンドを使用して、TSM サーバー上にデータベースのバックアップを作成します。

   ```
   db2 backup db zample use tsm options "'-asnodename=clusternode'"
   ```

   次のような情報が戻されます。

   ```
   Backup successful. The timestamp for this backup image is : 20090216151025
   ```

   BACKUP DATABASE コマンドで **-asnodename** オプションを指定する代わりに、**vendoropt** データベース構成パラメーターを更新できます。

   注: パーティション・データベース環境では、すべてのデータベース・パーティションでこのステップを実行する必要があります。データベース・パーティション上でこのステップを実行する順序は、オンライン・バックアップを実行するか、それともオフライン・バックアップを実行するかによって違います。詳しくは、 320 ページの『データのバックアップ』を参照してください。

6. 以下のコマンドを使用して、zample データベースに接続します。

   ```
   db2 connect to zample
   ```

7. データベースに関する新しいトランザクション・ログを生成します。そうするには、以下のコマンドを使って表を作成し、TSM サーバーにデータをロードします。

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader
    replace into employee copy yes use tsmwhere
   ```

   この例では、表の名前は employee であり、区切り文字で区切られている mr という名前の ASCII ファイルからデータをロードします。**COPY** YES オプションを指定することにより、ロードされるデータのコピーを作成します。**USE TSM** オプションは、TSM サーバーにデータのコピーを保管することを指定します。

   注: COPY YES オプションを指定できるのは、データベースのロールフォワード・リカバリーが使用可能になっている場合だけです。つまり、**logarchmeth1** データベース構成パラメーターが USEREXIT、 LOGRETAIN、DISK、または TSM に設定されている必要があります。

   進行状況を示すために、ロード・ユーティリティーは次のような一連のメッセージを戻します。

   ```
   SQL3109N  The utility is beginning to load data from file "/home/roecken/mr".

   SQL3500W  The utility is beginning the "LOAD" phase at time "02/16/2009
   15:12:13.392633".
   ```

```
SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3110N  The utility has completed processing.  "1" rows were read from the
input file.

SQL3519W  Begin Load Consistency Point. Input record count = "1".

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the "LOAD" phase at time "02/16/2009
15:12:13.445718".


Number of rows read         = 1
Number of rows skipped      = 0
Number of rows loaded       = 1
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 1
```

8. データが表にロードされた後、zample データベースに対して以下のような照会を実行して、TSM サーバー上に 1 つのバックアップ・イメージ、1 つのロード・コピー・イメージ、および 1 つのログ・ファイルが存在することを確認します。

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
```

次の情報が戻されます。

```
Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
    Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
    1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

9. この例では、コンピューター 2 (つまり dps) は zample データベースのノード間リカバリー用にまだセットアップされていません。以下のコマンドを使用して、このユーザーおよびコンピューターに関連付けられたデータが存在しないことを確認します。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
```

次の情報が戻されます。

```
  --- Database directory is empty ---
Warning: There are no file spaces created by DB2 on the ADSM server
Warning: No DB2 backup images found in ADSM for any alias.
```

10. 以下のコマンドを使用して、プロキシー・ノード clusternode に関連付けられた zample データベースのオブジェクトのリストを TSM サーバーに照会します。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
   options="-asnodename=clusternode"
```

次の情報が戻されます。

```
--- Database directory is empty ---

Query for database ZAMPLE

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
    Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
    1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

この情報は既に生成された TSM 情報と一致するため、このイメージを dps コンピューターにリストアできることが確認されます。

11. 以下のコマンドを使用して、zample データベースを TSM サーバーから dps コンピューターにリストアします。

```
dps:/home/regress9> db2 restore db zample use tsm options
   "'-asnodename=clusternode'" without prompting
```

次の情報が戻されます。

```
DB20000I  The RESTORE DATABASE command completed successfully.
```

注: dps に zample データベースがすでに存在している場合は、 **OPTIONS** パラメーターを省略し、データベース構成パラメーター **vendoropt** を使用することになります。この構成パラメーターは、バックアップまたはリストア操作の **OPTIONS** パラメーターをオーバーライドします。

12. 新しい表を作成して新しいデータをロードしたときに zample データベース・ログ・ファイルに記録されたトランザクションを適用するために、ロールフォワード操作を実行します。この例では、次のようなロールフォワード操作の試行は失敗します。ユーザーとコンピューターの情報が指定されないため、ロールフォワード・ユーティリティーはログ・ファイルを見つけることができません。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次のようなエラーがコマンドによって戻されます。

```
SQL4970N  Roll-forward recovery on database "ZAMPLE" cannot reach the
specified stop point (end-of-log or point-in-time) because of missing log
file(s) on node(s) "0".
```

適切な **logarchopt** 値を使用して、別のコンピューター上のログ・ファイルをロールフォワード・ユーティリティーに強制的に検索させます。この例では、次のようなコマンドを使って **logarchopt1** データベース構成パラメーターを設定し、ユーザー roecken およびコンピューター bar に関連付けられたログ・ファイルを検索します。

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-asnodename=clusternode'"
```

13. 以下のコマンドを使って **vendoropt** データベース構成パラメーターを設定することにより、ロールフォワード・ユーティリティーがバックアップおよびロード・コピー・イメージを使用できるようにします。

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-asnodename=clusternode'"
```

14. 以下のコマンドを使用して、zample データベース・ログ・ファイルに記録されたトランザクションを適用することにより、ノード間のデータ・リカバリーを完了できます。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次の情報が戻されます。

```
                                  Rollforward Status

 Input database alias                   = zample
 Number of members have returned status = 1


 Member number  Rollforward  Next log to  Log files processed        Last committed transaction
                status       be read
 -------------  -----------  -----------  -------------------------  ------------------------------
         0      not pending               S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC


DB20000I  The ROLLFORWARD command completed successfully.
```

ユーザー regress9 の下でのコンピューター dps 上のデータベース zample が、ユーザー roecken の下でのコンピューター bar 上のデータベースと同じポイントにリカバリーされました。

例 4: DB2 pureScale環境でクライアント・プロキシー・ノードを使用するよう TSM サーバーを構成する

この例では、ログ・アーカイブとバックアップが TSM サーバー上に保管され、PASSWORDACCESS=GENERATE オプションを使ってパスワードが管理されている場合に、一方のメンバーから、もう一方のメンバーにデータをリカバリーできるように、 2 つのメンバーをプロキシー・ノードとしてセットアップする方法を示します。

注: データベース構成を更新した後に、データベースのオフライン・バックアップを取っておく必要があるかもしれません。

この例では、メンバー member1 および member2 がプロキシー名 clusternode の下で登録されます。 DB2 pureScale環境では、任意のメンバーからバックアップまたはデータ・リカバリー操作を実行できます。この例では、member2 からデータをリカバリーします。

1. 以下のコマンドを使用して、TSM サーバー上でメンバー member1 および member2 をプロキシー・ノードとして登録します。

   ```
   REGISTER NODE clusternode mypassword
   GRANT PROXYNODE TARGET=clusternode AGENT=member1,member2
   ```

2. データベースのログを TSM サーバーにアーカイブできるようにするには、以下のコマンドを使用して、zample データベースに関するデータベース構成パラメーター **logarchmeth1** を更新します。

   ```
   member1:/home/roecken> db2 update db cfg for zample using
      LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
   ```

   注: DB2 pureScale環境では、グローバル **logarchmeth1** データベース構成パラメーターを、任意のメンバーから一度設定するだけで済みます。

   次の情報が戻されます。

   ```
   DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
   ```

3. 以下のコマンドを使用して、すべてのユーザーとアプリケーションをデータベースから切断します。

   ```
   db2 force applications all
   ```

4. 以下のコマンドを使用して、データベースに接続しているアプリケーションが存在しないことを確認します。

   ```
   db2 list applications global
   ```

   データが戻されなかったことを示すメッセージを受け取るはずです。

5. 以下のコマンドを使用して、TSM サーバー上にデータベースのバックアップを作成します。

   ```
   db2 backup db zample use tsm options '-asnodename=clusternode'
   ```

   次のような情報が戻されます。

   ```
   Backup successful. The timestamp for this backup image is : 20090216151025
   ```

   BACKUP DATABASE コマンドで **-asnodename** オプションを指定する代わりに、**vendoropt** データベース構成パラメーターを更新できます。

   注: DB2 pureScale環境では、このコマンドを任意のメンバーから実行して、データベースの全データをバックアップできます。

6. 以下のコマンドを使用して、zample データベースに接続します。

   ```
   db2 connect to zample
   ```

7. データベースに関する新しいトランザクション・ログを生成します。そうするには、以下のコマンドを使って表を作成し、TSM サーバーにデータをロードします。

   ```
   member1:/home/roecken> db2 load from mr of del modified by noheader replace
         into employee copy yes use tsmwhere
   ```

   この例では、表の名前は employee であり、区切り文字で区切られている mr という名前の ASCII ファイルからデータをロードします。**COPY** YES オプショ

ンを指定することにより、ロードされるデータのコピーを作成します。**USE TSM** オプションは、TSM サーバーにデータのコピーを保管することを指定します。

注: COPY YES オプションを指定できるのは、データベースのロールフォワード・リカバリーが使用可能になっている場合だけです。つまり、**logarchmeth1** データベース構成パラメーターが USEREXIT、 LOGRETAIN、DISK、または TSM に設定されている必要があります。

進行状況を示すために、ロード・ユーティリティーは次のような一連のメッセージを戻します。

```
SQL3109N  The utility is beginning to load data from file "/home/roecken/mr".

SQL3500W  The utility is beginning the "LOAD" phase at time "02/16/2009
15:12:13.392633".

SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3110N  The utility has completed processing.  "1" rows were read from the
input file.

SQL3519W  Begin Load Consistency Point. Input record count = "1".

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the "LOAD" phase at time "02/16/2009
15:12:13.445718".


Number of rows read         = 1
Number of rows skipped      = 0
Number of rows loaded       = 1
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 1
```

8. データが表にロードされた後、zample データベースに対して以下のような照会を実行して、TSM サーバー上に 1 つのバックアップ・イメージ、1 つのロード・コピー・イメージ、および 1 つのログ・ファイルが存在することを確認します。

```
member1:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
```

次の情報が戻されます。

```
Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
    Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
    1 Time: 20090216151213
```

```
Retrieving LOG ARCHIVE information.

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.01.10

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.01.11

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.01.19

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.02.49

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.03.15

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.03.15

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.03.16
```

9. 以下のコマンドを使用して、プロキシー・ノード clusternode に関連付けられた zample データベースのオブジェクトのリストを TSM サーバーに照会します。

```
member2:/home/regress9/sqllib/adsm> db2adutl query db zample
   options="-asnodename=clusternode"
```

次の情報が戻されます。

```
   --- Database directory is empty ---

   Query for database ZAMPLE

   Retrieving FULL DATABASE BACKUP information.
       1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
       Sessions: 1

   Retrieving INCREMENTAL DATABASE BACKUP information.
     No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

   Retrieving DELTA DATABASE BACKUP information.
     No DELTA DATABASE BACKUP images found for ZAMPLE

   Retrieving TABLESPACE BACKUP information.
     No TABLESPACE BACKUP images found for ZAMPLE

   Retrieving INCREMENTAL TABLESPACE BACKUP information.
     No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

   Retrieving DELTA TABLESPACE BACKUP information.
     No DELTA TABLESPACE BACKUP images found for ZAMPLE

   Retrieving LOAD COPY information.
       1 Time: 20090216151213

Retrieving LOG ARCHIVE information.

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.01.10

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.01.11

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.01.19

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.02.49

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.03.15
```

```
Log file: S0000002.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.03.15

Log file: S0000002.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.03.16
```

この情報は既に生成された TSM 情報と一致するため、このイメージを member2 メンバーにリストアできることが確認されます。

10. 以下のコマンドを使用して、TSM サーバー上の zample データベースを member2 メンバーからリストアします。

```
member2:/home/regress9> db2 restore db zample use tsm options
   '-asnodename=clusternode' without prompting
```

次の情報が戻されます。

```
DB20000I  The RESTORE DATABASE command completed successfully.
```

注: member2 に zample データベースがすでに存在している場合は、 **OPTIONS** パラメーターを省略し、データベース構成パラメーター **vendoropt** を使用することになります。この構成パラメーターは、バックアップまたはリストア操作の **OPTIONS** パラメーターをオーバーライドします。

11. 以下のコマンドを使って **vendoropt** データベース構成パラメーターを設定することにより、ロールフォワード・ユーティリティーがバックアップおよびロード・コピー・イメージを使用できるようにします。

```
member2:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-asnodename=clusternode'"
```

注: DB2 pureScale環境では、グローバル **vendoropt** データベース構成パラメーターを、任意のメンバーから一度設定するだけで済みます。

12. 以下のコマンドを使用して、zample データベース・ログ・ファイルに記録されたトランザクションを適用することにより、メンバー間のデータ・リカバリーを完了できます。

```
member2:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次の情報が戻されます。

```
                                  Rollforward Status

 Input database alias                   = zample
 Number of members have returned status = 3


 Member number  Rollforward  Next log to  Log files processed        Last committed transaction
                status       be read
 -------------  -----------  -----------  -------------------------  ------------------------------
             0  not pending               S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
             1  not pending               S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
             2  not pending               S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC

DB20000I  The ROLLFORWARD command completed successfully.
```

ユーザー regress9 の下でのメンバー member2 上のデータベース zample が、ユーザー roecken の下でのメンバー member1 上のデータベースと同じポイントにリカバリーされました。

ドロップされた表のリカバリー

ときに、まだ必要なデータを含む表をドロップしてしまうことがあります。そのような場合は、表をドロップした後でリカバリー可能な表を考慮するとよいでしょう。

表データを、データベース・リストア操作によってリカバリーし、その後、表がドロップされる前のポイント・イン・タイムまでのデータベース・ロールフォワード操作を実行できます。リストアおよびロールフォワード操作はデータベースが大きいと時間がかかり、そのリカバリーの間はデータは使用できなくなります。

ドロップされた表をリカバリーするフィーチャーにより、表スペース・レベルのリストアおよびロールフォワード操作を使用して、ドロップされた表データをリカバリーすることができます。

この表スペース・レベルのリカバリーはデータベース・レベルのリカバリーより速く、ユーザーもデータベースをそのまま使用できます。

始める前に

ドロップされた表がリカバリー可能になるには、表が存在する表スペースで 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。このオプションの有効化は、表スペース作成の間に行うことができます。または、ALTER TABLESPACE ステートメントを起動することによっても行えます。DROPPED TABLE RECOVERY オプションは表スペースに固有で、REGULAR 表スペースに限定されます。ある表スペースで、ドロップされた表のリカバリーが可能かどうかを判別するには、 SYSCAT.TABLESPACES カタログ表にある DROP_RECOVERY 列を照会することができます。

ドロップされた表のリカバリー・オプションは、表スペースの作成時にはデフォルトでオンとなっています。ドロップされた表のリカバリー用の表スペースを有効にしない場合は、CREATE TABLESPACE ステートメントの発行時に DROPPED TABLE RECOVERY オプションを明示的に OFF に設定するか、または ALTER TABLESPACE ステートメントを使用して、既存の表スペースのドロップされた表リカバリーを無効にすることができます。ドロップされた表のリカバリー操作が多数の場合、または履歴ファイルが大規模である場合、ドロップされた表のリカバリー・フィーチャーは、順方向リカバリーのパフォーマンスに影響を与える可能性があります。

DROP TABLE ステートメントが、ドロップされた表のドロップが可能になっている表スペースを持つ表に対して実行されると、追加の項目 (ドロップされた表を識別する) がログ・ファイル内に作成されます。項目はリカバリー履歴ファイルにも作成され、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。

パーティション表では、ドロップされた表のリカバリーは常にオンになります。パーティション表がまたは表スペースの内の 1 つ以上の表スペースの「ドロップされた表のリカバリー」が、含まれる非パーティション表のためにオフにされても、パーティション表についてはオンです。パーティション表では、ドロップされた表のログ・レコードが 1 つだけ作成されます。表のすべてのデータ・パーティションをリカバリーするには、このログ・レコードだけで十分です。

このタスクについて

ドロップ時に、表が REORG 保留状態であった場合、ヒストリー・ファイルの CREATE TABLE DDL はインポート・ファイルの CREATE TABLE DDL と完全には一致しません。最初の **REORG** 推奨の ALTER が実行される前に、インポート・

ファイルは表のフォーマットになりますが、履歴ファイルの CREATE TABLE ステートメントは、ALTER TABLE ステートメントの結果を含む表の状態と一致します。

LOAD または IMPORT に使用するファイル・タイプ修飾子

ロードまたはインポートによって表をリカバリーする際は、以下のファイル・タイプ修飾子を指定してください。

- リカバリーされるデータが GRAPHIC または VARGRAPHIC データ・タイプである場合、 **IMPORT** または **LOAD** コマンドに **usegraphiccodepage** ファイル・タイプ修飾子を使用する必要があります。複数のコード・ページが含まれている可能性があるためです。
- **IMPORT** または **LOAD** コマンドに **delprioritychar** ファイル・タイプ修飾子を使用する必要があります。これによって、**LOAD** および **IMPORT** は、CHARACTER または GRAPHIC の列データ中に改行文字が含まれている行を解析できます。

制約事項

一度でリカバリーできるドロップされた表は 1 つだけです。

ドロップされた表からリカバリー可能なデータのタイプについて、いくつかの制限事項があります。以下のものは、リカバリーすることはできません。

- DROPPED TABLE RECOVERY オプションは一時表には使用できません。
- 行タイプと関連したメタデータ。 (データはリカバリーされますが、メタデータはリカバリーされません。)型付き表の階層表にあるデータはリカバリーされます。このデータには、ドロップされた型付き表に現れたよりも詳細にわたる情報が含まれる場合があります。
- XML データ。 XML データを含むドロップされた表をリカバリーしようとすると、対応する列データは空になります。

手順

以下のようにして、ドロップされた表をリカバリーできます。

1. **LIST HISTORY DROPPED TABLE** コマンドを起動して、ドロップされた表を識別します。 ドロップされた表の ID は、Backup ID 列にリストされます。
2. 表がドロップされる前に作成されたデータベース・レベルまたは表スペース・レベルのバックアップ・イメージをリストアします。
3. 表データが含まれているファイルを書き込むエクスポート・ディレクトリーを作成します。 このディレクトリーは、すべてのデータベース・パーティションからアクセスできるものか、またはそれぞれのデータベース・パーティションに存在するかのいずれかでなければなりません。このエクスポート・ディレクトリーの下のサブディレクトリーは、各データベース・パーティションごとに自動的に作成されます。これらのサブディレクトリーの名前は NODE*nnnn* です。ここで、*nnnn* はデータベース・パーティションまたはノード番号を表します。それぞれのデータベース・パーティションにあったときと同じ、ドロップされた表データを含むデータ・ファイルは、 data という下位のサブディレクトリーにエクスポートされます。 例えば、以下のようにします。

   ```
   ¥export_directory¥NODE0000¥data.
   ```

4. **ROLLFORWARD DATABASE** コマンドの **RECOVER DROPPED TABLE** パラメーターを使用して、表がドロップされた後の特定のポイント・イン・タイムまでロールフォワードします。または、ログの最後までロールフォワードします。こうすると、表スペースまたはデータベース内の他の表への更新は失われません。
5. リカバリー履歴ファイルから CREATE TABLE ステートメントを使用して表を再作成します。
6. ロールフォワード操作中にエクスポートされた表データを、表にインポートします。 ドロップの発生時に表が REORG 保留状態にあった場合、CREATE TABLE DDL の内容はデータ・ファイルの内容と一致するように変更されなければならない場合があります。

クラッシュ・リカバリー

データベースに対するトランザクション (つまり作業単位) は、予期しない割り込みを受けることがあります。例えば、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、ディスクに書き込まれる前に障害が発生すると、データベースは矛盾した、または使用不能な状態のままになっています。*クラッシュ・リカバリー* とは、データベースを整合した使用可能な状態に戻すプロセスのことです。

これは、未完了のトランザクションをロールバックし、破損発生時にメモリーに残っていたコミット済みトランザクションを完了することによって行われます (図 18)。データベースが整合性があり使用可能な状態の場合には、これは「*整合点*」にあることになります。

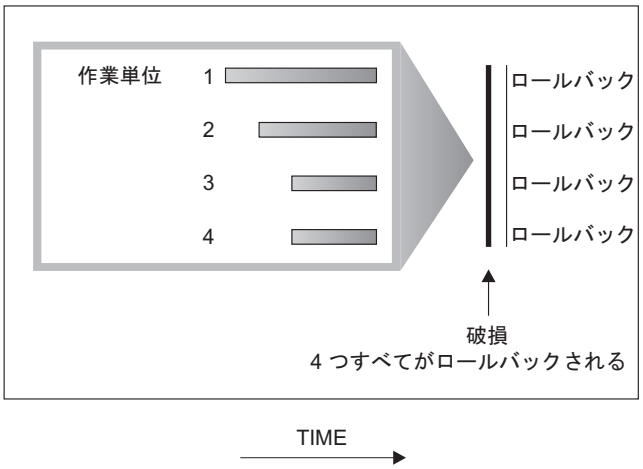


図 18. 作業単位のロールバック (クラッシュ・リカバリー)

IBM DB2 pureScale Feature を使用している場合は、*メンバー・クラッシュ・リカバリー* および*グループ・クラッシュ・リカバリー* という、2 種類の固有のクラッシュ・リカバリー方法があることに注意してください。メンバー・クラッシュ・リカバリーとは、メンバーに障害が発生した場合に、単一のメンバーのログ・ストリームを使用して、データベースの一部をリカバリーするプロセスのことです。メンバー・クラッシュ・リカバリーは、通常、メンバー再始動の一環として自動的に開始されるオンライン操作です。つまり、この操作中、他のメンバーは、データベースにアクセスできます。複数のメンバーで、同時にメンバー・クラッシュ・リカバリ

ーが行われる場合もあります。グループ・クラッシュ・リカバリーとは、障害によって、クラスター内に使用可能なクラスター・キャッシング・ファシリティーが存在しなくなった場合に、複数メンバーのログ・ストリームを使用して、データベースをリカバリーするプロセスのことです。グループ・クラッシュ・リカバリーも、通常は (グループ再始動の一環として) 自動的に開始されるものであり、 DB2 pureScale 環境以外の DB2 クラッシュ・リカバリー操作と同様に、このリカバリーの進行中は、データベースにアクセスできません。

データベースまたはデータベース・マネージャーに障害が発生すると、データベースは不整合な状態のままとなる可能性があります。障害発生時に未完了であったトランザクションによる変更が、データベースの内容に含まれている可能性があります。また、障害前に完了したトランザクションによって行われた変更で、まだディスクにフラッシュされていなかったものが、データベースから失われている可能性もあります。クラッシュ・リカバリー操作を実行して、部分的にしか完了しなかったトランザクションをロールバックし、メモリーにしか書き込まれていなかった完了トランザクションによる変更をディスクに書き込む必要があります。

クラッシュ・リカバリーが必要となる状況には、次のようなものがあります。

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

データベース・マネージャーにクラッシュ・リカバリーを自動的に実行させるには、データベース構成パラメーター (**autorestart**) を ON に設定することによって、自動再始動を有効にします。 (これはデフォルト値です。) 自動再始動を動作したくない場合、 **autorestart** データベース構成パラメーターを「OFF」に設定してください。結果として、データベース障害の発生時に **RESTART DATABASE** コマンドを発行する必要があります。破損が発生する前にデータベース入出力が SUSPEND された場合には、クラッシュ・リカバリーが継続するように、**RESTART DATABASE** コマンドの **WRITE RESUME** オプションを指定する必要があります。データベースが再始動操作を開始すると、管理通知ログに記録されます。

ロールフォワード・リカバリーが使用可能になっている (つまり **logarchmeth1** 構成パラメーターが OFF に設定されていない) データベースで、クラッシュ・リカバリーが発生し、そのクラッシュ・リカバリー中に個別の表スペースが原因のエラーが発生した場合、その表スペースはオフラインになり、修復されるまでアクセスできなくなります。クラッシュ・リカバリーは他の表スペースで続行します。クラッシュ・リカバリーが完了した時点で、データベースに入っている他の表スペースはアクセス可能であり、データベースへの接続は確立できます。しかしながら、オフラインになった表スペースがシステム・カタログを含む表スペースである場合には、その表スペースは、いずれかの接続が許可される前に修復されなければなりません。この動作は、 DB2 pureScale 環境 には当てはまりません。メンバー・クラッシュ・リカバリーまたはグループ・クラッシュ・リカバリー中にエラーが発生した場合、そのクラッシュ・リカバリー操作は失敗します。

損傷を受けた表スペースのリカバリー

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

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

- データベースをリストアする
- カタログ表スペースをリストアする。

注:

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

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

損傷を受けた表スペースが唯一の TEMPORARY 表スペースである場合には、データベースに接続できたらすぐに新しい TEMPORARY 表スペースを作成しなければなりません。一度作成されると、新しい TEMPORARY 表スペースが使用できるようになり、 TEMPORARY 表スペースが必要な通常のデータベース操作を再開できます。それから、任意でオフライン状態の TEMPORARY 表スペースをドロップします。 SYSTEM TEMPORARY 表スペースを使用する表の再編成については、特別な考慮事項があります。

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

リカバリー可能データベースの表スペースのリカバリー

クラッシュ・リカバリーが必要な場合、損傷のある表スペースはオフラインになりアクセス不能になります。その表スペースはロールフォワード・ペンディング状態になります。追加の問題がない場合、損傷のある表スペースが存在していても、再始動操作によってデータベースはオンラインになります。オンラインになると、損傷のある表スペースは使用できませんが、データベースの残りの部分は使用可能です。損傷のある表スペースを修正してそれを使用可能にするには、以下の手順に従ってください。

手順

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

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

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

リカバリー不能データベースの表スペースのリカバリー

クラッシュ・リカバリーが必要であるものの、損傷のある表スペースが存在する場合は、その表スペースをドロップした場合にのみデータベースを正常に再始動できます。リカバリー不能データベースでは、損傷のある表スペースをリカバリーするのに必要なログは保持されません。したがって、そのような表スペースで有効なアクションは、これらをドロップすることだけです。

手順

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

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

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

   注: DROP PENDING TABLESPACES リストに表スペース名を入れても、この表スペースが DROP PENDING 状態になったことにはなりません。このような状態になるのは、表スペースが再始動操作中に損傷を受けた場合だけです。

3. データベース再始動操作が成功したら、**LIST TABLESPACES** コマンドを呼び出して、どの表スペースがドロップ・ペンディング状態であるかを調べてください。
4. DROP TABLESPACE ステートメントを発行してドロップ・ペンディング状態にある各表スペースをドロップします。 損傷を受けた表スペースをドロップしたら、その表スペースが使用していたスペースを再利用することも、表スペースを再作成することもできます。
5. これらの表スペースをドロップして損傷を受けた表スペースのデータを失うことを望まない場合は、以下を実行します。
   - 損傷を受けているコンテナーを修正します (元のデータを失わないようにする)。
   - **RESTART DATABASE** コマンドを再発行します。
   - データベースのリストア操作を実行します。

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リカバリー処理を簡単に実行できるようにするためには、以下の操作を実行します。

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

ディスク障害に対する保護

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

ディスク・アレイは、単に RAID (Redundant Array of Independent Disks) と呼ばれることがあります。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・コントローラーにもアタッチされています (2 つの SCSI アダプターと同じ)。データの保護機能は良好です。つまり、どちらのディスクに障害が発生しても、データは他のディスクからアクセス可能です。ディスク・デュプレキシングでは、一方のディスク・コントローラーに障害が発生したとしても、データ保護が保たれます。パフォーマンスは良好ですが、これをインプリメントすると通常の 2 倍のディスクが必要になります。

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

RAID レベル 1+0 (10) では、少なくとも 2 つのディスク間でデータのミラーリングおよびストライピングが行われます。ミラーリングによって、2 つ以上のディスクに同時にデータが書き込まれるため、RAID レベル 1 と同等の耐障害性が得られます。ストライピングによって、データがブロックに分割され、各ブロックは別々のディスク・ドライブに書き込まれます。多数のチャネルおよびドライブに I/O 負荷が分散されるため、高い I/O パフォーマンスが実現します。ただし、RAID レベル 1+0 では、すべてのデータがミラーリングされるため、有効なディスク・スペースは半分に減ります。RAID レベル 10 を実装するには、最小 4 つのドライブが必須です。

RAID レベル 0+1 は、セグメントが RAID 0 アレイであるミラーリングされたアレイとして実装され、RAID レベル 5 と同等の耐障害性を備えます。多数のチャネルおよびドライブに I/O 負荷を分散することによって、高い I/O 率が得られます。RAID レベル 0+1 を RAID レベル 1+0 と混同しないように注意してください。単一ドライブ障害が発生すると、アレイ全体は、実質的に RAID レベル 0 アレイとなります。

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

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

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

ソフトウェア・ディスク・アレイ

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

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

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

トランザクション障害の影響の緩和

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

- 各 DB2 サーバーでの中断されない電源供給
- すべてのデータベース・パーティションでデータベース・ログに十分なディスク・スペース

- パーティション・データベース環境においては、データベース・パーティション・サーバー間の高信頼性通信リンク
- パーティション・データベース環境では、システム・クロックの同期

パーティション・データベース環境におけるトランザクション障害のリカバリー

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

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

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

READ-ONLY
サーバーではデータの変更は行われなかった。

YES サーバーではデータの変更が行われた。

NO エラーが発生したため、サーバーはコミットの準備ができない。

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

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

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

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

- アクティブ・データベース・パーティション・サーバーがアプリケーションのコーディネーター・パーティションで、障害データベース・パーティション・サーバー (ただし COMMIT の準備はできていない) でそのアプリケーションが実行されていた場合は、コーディネーター・エージェントは障害リカバリーを実行するための割り込みが行われます。コーディネーター・エージェントが COMMIT 処理の 2 番目のフェーズにある場合は、アプリケーションに SQL0279N が戻されてから、データベース接続が切断されます。そうでない場合は、コーディネーター・エージェントはトランザクションに関係する他のすべてのサーバーに対して ROLLBACK 要求を配布し、 SQL1229N がアプリケーションに戻されます。
- 障害データベース・パーティション・サーバーがアプリケーションのコーディネーター・パーティションであった場合は、アクティブ・サーバーでそのアプリケーションに対し現在でも作業を実行しているエージェントは、障害リカバリーを実行するための割り込みが行われます。トランザクションが準備済みの状態にない各データベース・パーティションでは、トランザクションはローカルにロールバックされます。トランザクションが準備済みの状態にあるデータベース・パーティションでは、トランザクションは未確定になります。コーディネーター・データベース・パーティションが使用可能でないため、コーディネーター・データベース・パーティションは、いくつかのデータベース・パーティションでトランザクションが未確定であることを認識しません。
- 障害データベース・パーティション・サーバーにアプリケーションが接続されていて (障害発生前)、ローカル・データベース・パーティション・サーバーも障害データベース・パーティション・サーバーもコーディネーター・パーティションでない場合は、このアプリケーションの処理を実行しているエージェントは割り込みが行われます。コーディネーター・パーティションは、ROLLBACK または切断メッセージを他のデータベース・パーティション・サーバーに送信します。コーディネーター・パーティションが SQL0279 を戻す場合、トランザクションは依然としてアクティブなデータベース・パーティション・サーバーで単に未確定になるだけです。

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

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

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

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

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

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

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

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

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

クラッシュ・リカバリーですべての未確定トランザクションが解決できない場合もあります。例えば、一部のデータベース・パーティション・サーバーが使用不能の場合などがそうです。コーディネーター・パーティションが、トランザクションにかかわっている他のデータベース・パーティションよりも前にクラッシュ・リカバリーを完了すると、クラッシュ・リカバリーで未確定トランザクションを解決できなくなります。そのような動作になるのは、クラッシュ・リカバリーがデータベース・パーティションごとに独立して実行されるからです。この場合、SQL 警告メッセージ SQL1061W が戻されます。未確定トランザクションはロックおよびアクティブ・ログ・スペースなどのリソースを保留するので、アクティブ・ログ・スペースが未確定トランザクションにより使用されたままになるため、データベースに対して変更を加えられなくなる場合があります。このため、クラッシュ・リカバリー後に未確定トランザクションが残っているかどうかを判別し、未確定トランザクシ

ョンを解決しなければならないすべてのデータベース・パーティション・サーバーを、できるだけ早期にリカバリーする必要があります。

注: パーティション・データベース・サーバー環境では、RESTART データベース・コマンドはノードごとに実行されます。すべてのノードでデータベースが再始動されるように、以下のコマンドを使用することをお勧めします。

```
db2_all "db2 restart database <database_name>"
```

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

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

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

障害のあるデータベース・パーティション・サーバーの識別

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

SQL0279N
この SQLCODE は、トランザクションに関係するデータベース・パーティション・サーバーが COMMIT 処理中に終了すると受け取られます。

SQL1224N
この SQLCODE は、障害が発生したデータベース・パーティション・サーバーがトランザクションのコーディネーター・パーティションであるときに受け取られます。

SQL1229N
この SQLCODE は、障害が発生したデータベース・パーティション・サーバーがトランザクションのコーディネーター・パーティションでないときに受け取られます。

どのデータベース・パーティション・サーバーに障害が発生したかの判別は、2 つのステップで構成されます。

1. SQLCA を調査することにより、障害を検出したパーティション・サーバーを見つけます。 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。)

2. 手順 1 で見つかったサーバーの管理通知ログで、障害が発生したサーバーのノード番号を調べます。

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することがあります。

データベース・パーティション・サーバーの障害からのリカバリー

手順

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

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

メインフレームまたはミッドレンジ・サーバー上の未確定トランザクションのリカバリー

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

トランザクションでアプリケーションがホストまたは System i データベース・サーバーにアクセスした場合には、未確定トランザクションがリカバリーされる方法は多少異なります。ホストまたは System i データベース・サーバーにアクセスするのに、 DB2 Connect が使用されます。 DB2 Connect に DB2 Syncpoint Manager が構成されている場合、リカバリー・ステップは異なります。

このタスクについて

ホストまたは System i サーバーにおける未確定トランザクションのリカバリーは、通常、トランザクション・マネージャー (TM) および DB2 Syncpoint Manager (SPM) によって自動的に行われます。ホストまたは System i 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 System iサーバーで未確定である間はその位置にリソースを保持します。ヒューリスティックな決定を行う必要があるとホストまたは System i の管理者が判断すると、ホストまたは System i 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 (例えば、電話で)。これが行われると、**LIST DRDA INDOUBT TRANSACTIONS** コマンドを使用して、ローカル DB2 Connect インスタンスでのトランザクションの状態を判別することができます。

手順

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. SPM に接続します。 以下に例を示します。

```
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。

   次の例は、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 System i サーバーのローカル別名です。 partner_lu がホストまたは System i サーバーの完全修飾 LU 名です。これはホストまたは System i サーバーを最もよく識別できるもので、そのトランザクションの呼び出し側で、ホストあるいは System i サーバーから得て提供されるものです。 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 System i サーバーで使用可能です。問題のトランザクションが表示されている場合には、uow_status フィールドを用いて、値が C (コミット) か R (ロールバック) の場合のトランザクションの結果を判別することができます。 **WITH PROMPTING** パラメーターを指定して **LIST DRDA INDOUBT TRANSACTIONS** コマンドを出す場合は、トランザクションのコミット、ロールバック、無視を対話式に行えます。

```
db2 => list drda indoubt transactions
DRDA Indoubt Transactions:
1.db_name: DBAS3    db_alias: DBAS3    role: AR
  uow_status: C  partner_status: I  partner_lu: USIBMSY.SY12DQA
corr_tok: USIBMST.STB3327L
luwid: USIBMST.STB3327.305DFDA5DC00.0001
xid: 53514C2000000017 00000000544D4442 0000000000305DFD A63055E962000000
     00035F
```

3. partner_lu および luwid の未確定トランザクションが表示されない場合、または **LIST DRDA INDOUBT TRANSACTIONS** コマンドが次のように戻る場合、トランザクションはロールバックされています。

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

次のタスク

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

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

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

多少異なります。ホストまたは System i データベース・サーバーにアクセスするのに、 DB2 Connect が使用されます。 DB2 Connect に DB2 同期点マネージャーが構成されている場合、リカバリー手順は異なります。

このタスクについて

DB2 Connect Personal Edition または DB2 Connect Enterprise Edition のいずれかからのマルチサイト更新で、DB2 (z/OS 版) を更新するために TCP/IP 接続が使用されていて、 DB2 Syncpoint Manager が使用されない場合は、このセクションの情報を活用してください。この状態での未確定トランザクションのリカバリーは、 DB2 Syncpoint Manager が関係する未確定トランザクションのリカバリーとは異なります。この環境で未確定トランザクションが発生すると、その問題の検出元に従い、クライアント、データベース・サーバー、またはトランザクション・マネージャー (TM) データベース (あるいはそれらの複数の組み合わせ) でアラート項目が生成されます。アラート項目は、`db2alert.log` ファイルに保管されます。

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

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

手順

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

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

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

災害時リカバリー

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

災害時リカバリーの計画には、以下の 1 つまたは複数が含まれます。

- 非常事態発生時に使用されるサイト
- データベースをリカバリーするための別のマシン

- データベース・バックアップ、表スペース・バックアップ、またはその両方とアーカイブ・ログのオフサイト・ストレージ

災害時リカバリー計画において別のマシンでのデータベース全体のリストアが行われる場合は、少なくとも 1 つのフル・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・ログを保持することをお勧めします。データベース内の表スペースごとに表スペースのフル・バックアップがある場合は、データベースを再構築できますが、この方式には非常に多くのバックアップ・イメージが関係するので、データベースのフル・バックアップを使用するリカバリーより時間がかかります。

ログをアーカイブする際にそれらをスタンバイ・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、データベースまたは表スペースのバックアップとログ・アーカイブをスタンバイ・サイトに維持し、災害発生後にだけリストアおよびロールフォワードを実行するようにもできます。(後者の場合は、最新のバックアップ・イメージが望ましい)。しかし、災害時の状況では、すべてのトランザクションを災害発生時までリカバリーすることは通常は不可能です。

災害時リカバリーに表スペースのバックアップが役に立つかどうかは、障害の範囲によって決まります。通常、データベース全体をリストアすれば災害時リカバリーは複雑でなく時間がかからないので、スタンバイ・サイトにデータベースのフル・バックアップを保持すべきです。その災害がディスクの損傷である場合には、表スペースのバックアップをそのディスクにある表スペースごとにリカバリーに使用できます。ディスク障害 (または他の理由) によりコンテナーにアクセスできない場合は、コンテナーを別の場所にリストアできます。

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

複製を使用して、部分的または完全なサイト障害からデータを保護することもできます。レプリケーションによって、複数のリモート・データベースに定期的にデータをコピーすることができます。 DB2 データベースは多数の複製ツールを提供しており、これらのツールを使用すると、コピーする必要のあるデータ、データのコピー先のデータベース表、および更新情報をコピーする頻度を指定できます。

ピアツーピア・リモート・コピー (PPRC) などのストレージ・ミラーリングを使用してデータを保護することもできます。PPRC はボリュームまたはディスクの同期コピーを提供して、災害に対する保護を行います。

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

バージョン・リカバリー

バージョン・リカバリーは、以前のバージョンのデータベースの修復であり、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。

このリカバリー方式は、リカバリー不能データベース (つまり、管理者がアーカイブ・ログを持っていないデータベース) に使用します。リカバリー可能データベースでも、**RESTORE DATABASE** コマンドで **WITHOUT ROLLING FORWARD** オプションを使用して、この方式を使用することもできます。

データベース・リストア操作では、以前に作成されたバックアップ・イメージを使用して、データベース全体がリストアされます。データベースのバックアップにより、データベースを、バックアップをとった時点と同じ状態にリストアすることができます。しかし、バックアップ時点から障害発生時点までのすべての作業単位は失われています (図 19 を参照)。

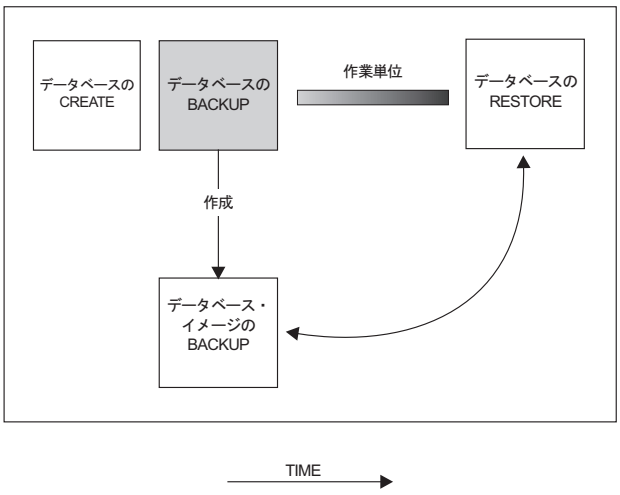


図 19. バージョン・リカバリー: バックアップ時点から障害発生時点までのすべての作業単位は失われることを示します。

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

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

ロールフォワード・リカバリー

データベースまたは表スペースに対して、ロールフォワード・リカバリーを実行できます。

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

注: 以下の場合は、**WITHOUT ROLLING FORWARD** パラメーターは使用できません。

- オンライン・バックアップ・イメージからリストアする場合
- 表スペース・レベルのリストアを発行する場合

リカバリー時に、アーカイブ・ログ・ファイルはアーカイブからリトリーブされます。アーカイブ・ログ・ファイルは、圧縮されている場合、自動的に圧縮解除されて使用されます。アーカイブ・ログ・ファイルは、アクティブ・ログ・パスまたはオーバーフロー・ログ・パス (そこにファイルを手動でコピーした場合) で検出されたときも自動的に圧縮解除されます。

考慮する 2 つのロールフォワード・リカバリーは次のとおりです。

- *データベースのロールフォワード・リカバリー*。このタイプのロールフォワード・リカバリーでは、データベース・ログに記録されているトランザクションがデータベース・リストア操作の後に適用されます (377 ページの図 20 を参照)。データベース・ログには、データベースへの変更がすべて記録されています。この方式では、データベースが特定の時点の状態に、または障害が生じる直前 (アクティブ・ログの最後) の状態にリカバリーされます。

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

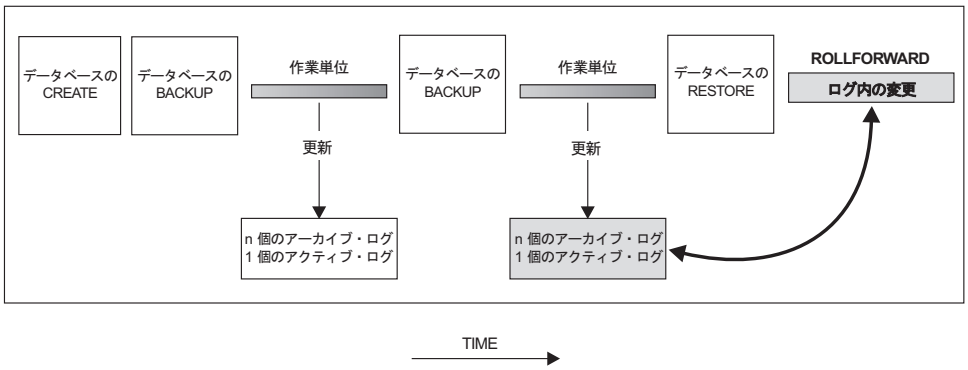


図 20. データベースのロールフォワード・リカバリー: 長時間実行しているトランザクションの場合には、複数のアクティブ・ログが生じる可能性があります。

- *表スペースのロールフォワード・リカバリー*。データベースの順方向リカバリーが可能であれば、表スペースのバックアップ、リストア、およびロールフォワードも可能です (378 ページの図 21 を参照)。表スペースのリストアおよびロールフォワードを行う場合、データベース全体 (つまり、すべての表スペース)、または 1 つ以上の個別表スペースのバックアップ・イメージが必要です。さらに、リカバリーする表スペースに影響を与えるログ・レコードが必要です。以下の 2 つのポイントのうちの 1 つにログをロールフォワードできます。
  - ログの終わった時点。または、
  - 特定の時点 (*ポイント・イン・タイム指定リカバリー*と呼ばれる)。

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

- 表スペース・リストア操作の後、表スペースは常にロールフォワード・ペンディング状態で、ロールフォワードする必要があります。 **ROLLFORWARD DATABASE** コマンドを呼び出し、特定の時点またはログの最後まで表スペースにログを適用してください。
- クラッシュ・リカバリーの後で 1 つまたは複数の表スペースが*ロールフォワード・ペンディング* 状態の場合、まず表スペースの問題を解決します。場合によっては、表スペースの問題を解決するのにデータベース・リストア操作が関係しない場合もあります。例えば、電源が切れると、表スペースはロールフォワード・ペンディング状態になります。この場合にはデータベース・リストア操作は必要ありません。表スペースの問題が解決したら、**ROLLFORWARD DATABASE** コマンドを使用して、ログの最後までのログを、表スペースに適用することができます。クラッシュ・リカバリーの前にこの問題が解決された場合、データベースを整合性のある使用可能状態にするのにクラッシュ・リカバリーで十分な場合があります。

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

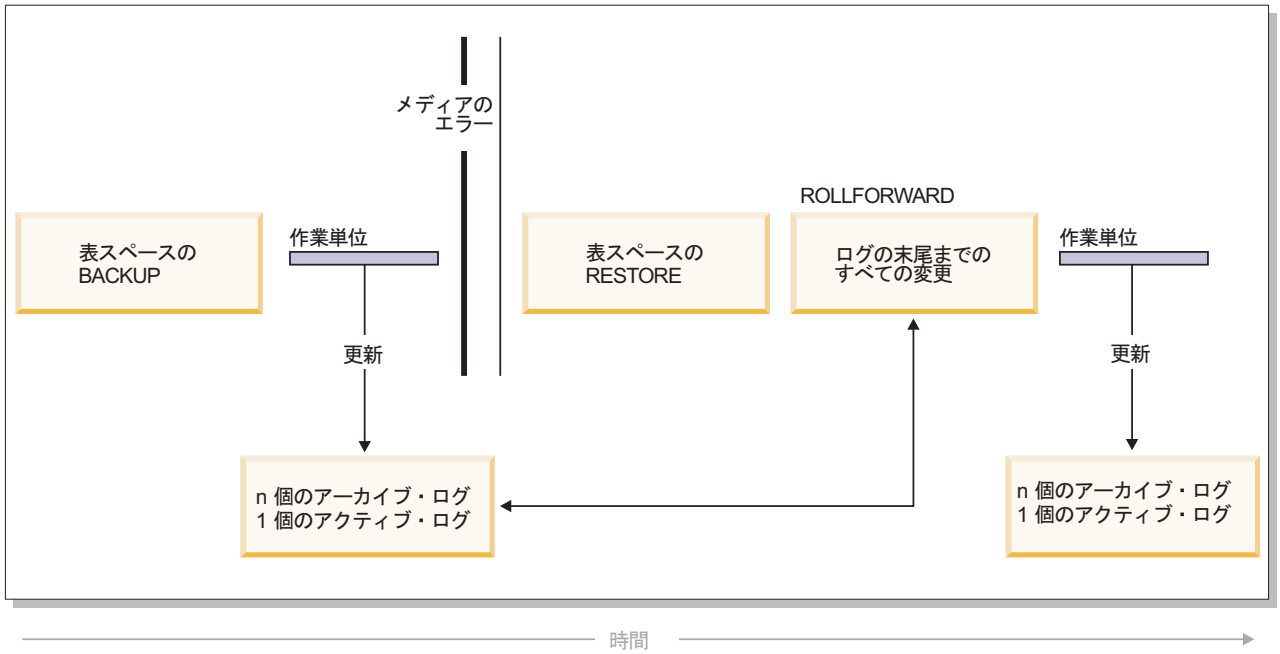


図 21. 表スペースのロールフォワード・リカバリー： 長時間実行しているトランザクションの場合には、複数のアクティブ・ログが生じる可能性があります。

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

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

表スペースのロールフォワード操作は、DB2 pureScale環境では異なる動作をします。詳しくは、 528 ページの『DB2 pureScale環境におけるログ・ストリーム・マージおよびログ・ファイル管理』および 533 ページの『DB2 pureScale環境のログ・シーケンス番号』を参照してください。

パーティション表の一部を含む表スペースをロールフォワードする際に、特定の時点までロールフォワードする場合は、その表を含む他のすべての表スペースも同じ時点までロールフォワードしなければなりません。ただし、ログの最後までロールフォワードする場合は、パーティション表の一部を含む単一の表スペースのみのロールフォワードができます。

パーティション表がデータ・パーティションをアタッチしたり、デタッチしたり、ドロップしたりした場合、ポイント・イン・タイム指定ロールフォワードにそれらのデータ・パーティションに関するすべての表スペースも含めなければなりません。パーティション表がデータ・パーティションをアタッチしたり、デタッチしたり、ドロップしたりしたかどうかを判別するには、SYSDATAPARTITIONS カタログ表を照会してください。

ロールフォワード・リカバリー中のストレージ・グループの変更

ロールフォワード操作中にストレージ・グループのパスの変更が再実行されるかどうかは、リストア・プロセス中にストレージ・グループがリダイレクトされたかどうかに依存します。データベースのリストア操作中にストレージ・グループを再定義していない場合、ストレージ・グループまたはそのパスに影響を与えるログ・レコードが、ロールフォワード・リカバリー中に適用されます。ログ・レコードに記述されているストレージ・パスの更新、ストレージ・グループのリネーム操作、および表スペースとストレージ・グループの関連付けの更新が、ロールフォワード操作中に適用されます。ロールフォワード操作が、ストレージ・パスの追加またはストレージ・グループの作成に関するログ・レコードの適用を試行しているときに、ストレージ・パスが見つからないと、エラー SQL1051N が返されます。

リストア操作中にストレージ・パスを再定義した場合、ロールフォワード操作は、ストレージ・パスに対する変更や、パスがリダイレクトされたストレージ・グループのメディア属性に対する変更を再実行しません。ただし、ストレージ・グループのデータ・タグや名前への変更は再実行されます。また、その他の操作 (DROP STOGROUP 操作を含む) のログ・レコードは適用されます。明示的に指定されているすべてのストレージ・グループのパスは、その必要な最終的パスに設定されていることが想定されています。

ログ内にリバランス操作が出現すると、ロールフォワード・リカバリー中に、表スペースのリバランス操作が開始されます。リバランス操作は、ロールフォワード操作の進行中に完了しない可能性があります。その場合、ロールフォワード操作の完了時にリバランス処理が中断され、次にデータベースがアクティブになったときに再開されます。

ロールフォワード操作中、ログ内に CREATE STOGROUP ステートメントが出現すると、CREATE STOGROUP ステートメントを発行したときに指定したパスに、ストレージ・グループが作成されます。

増分バックアップおよびリカバリー

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

以下の問題を考慮に入れてください。

- ウェアハウス中のデータ変更のパーセンテージが低い場合は、データベース全体のバックアップを取るべきではない。

- 既存のデータベースに表スペースを付加した後に表スペースのバックアップしか行わないのは危険。なぜなら、表スペースのバックアップを行っている間に、バックアップ対象の表スペース以外に変更が加えられなかった保証はないからです。

この問題に取り組むため、DB2 には増分バックアップおよびリカバリーが用意されています。

増分バックアップ とは、前回のバックアップ以降に更新されたページだけを含むバックアップ・イメージのことです。個々の増分バックアップ・イメージには、更新されたデータ・ページと索引ページに加えて、通常は全バックアップ・イメージに保管されるすべての初期データベース・メタデータ (データベースの構成、表スペースの定義、データベースの履歴など) も含まれます。

注:

1. 表スペースにロング・フィールドまたはラージ・オブジェクト・データが含まれていて、増分バックアップが行われる場合に、その表スペースの任意のページが前のバックアップから変更されていると、ロング・フィールドまたはラージ・オブジェクト・データはすべて、バックアップ・イメージにコピーされます。
2. ダーティー・ページ (変更が加えられたがまだディスクに書き込まれていないデータを含むページ) を含む表スペースの増分バックアップを取る場合は、すべてのラージ・オブジェクト・データがバックアップされます。通常のデータは、変更が加えられた場合のみバックアップされます。
3. **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドで **ADD DBPARTITIONNUMS** パラメーターが指定されていると、データ再配分によってすべての新規データベース・パーティションに表スペースが作成される可能性があります。その場合、増分バックアップ操作に影響を与える可能性があります。

2 種類の増分バックアップがサポートされています。

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

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

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

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

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

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

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

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

増分バックアップ・イメージからのリストア

このタスクについて

- 増分バックアップ・イメージからのリストア操作は、常に以下のステップから成ります。
  1. 増分ターゲット・イメージを識別します。

     リストアする最終イメージを決定し、DB2 リストア・ユーティリティーによる増分リストア操作を要求しなければなりません。このイメージは、リストアされる最終イメージになるので、増分リストアのターゲット・イメージとして認識されます。増分ターゲット・イメージは **RESTORE DATABASE** コマンドの **TAKEN AT** パラメーターを使用して指定します。
  2. 最新の全データベースまたは表スペースのイメージをリストアして、以後の増分バックアップ・イメージの適用対象となるベースラインを確立します。

3. ステップ 2 でリストアしたベースライン・イメージの上部に、個々の必要な全データベースまたは表スペース増分バックアップ・イメージを、作成順にリストアします。
4. ステップ 1 のターゲット・イメージが 2 度目に読み取られるまで、ステップ 3 を繰り返します。増分リストア操作が完了するまでの間に、ターゲット・イメージは 2 度アクセスされます。最初のアクセスの際には、初期データだけがイメージから読み取られます。ユーザー・データは読み取られません。イメージが完全に読み取られて処理されるのは 2 度目のアクセス時だけです。

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

- 増分バックアップ・イメージをリストアするには、2 つの方法があります。
  - 自動増分リストアでは、**RESTORE DATABASE** コマンドを 1 回だけ発行して、使用するターゲット・イメージを指定します。その後 DB2 Database for Linux, UNIX, and Windows は、データベース履歴を使用して残っている必要なバックアップ・イメージを判別し、それらをリストアします。
  - 手動増分リストアでは、リストアする必要があるバックアップ・イメージごとに、 **RESTORE DATABASE** コマンドを 1 回発行する必要があります (上記でリストされたステップで説明されています)。

手順

- **自動増分リストアの例**

  自動増分リストアを使用して増分バックアップ・イメージの集合をリストアするには、 **RESTORE DATABASE** コマンドに **TAKEN AT** timestamp パラメーターを指定してください。リストアする最終イメージのタイム・スタンプを使用してください。以下に例を示します。

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

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

  処理の 2 番目の段階で、データベース履歴が照会され、要求されたリストア操作を実行するのに必要なバックアップ・イメージのチェーンが構築されます。何らかの理由で照会できず、DB2 Database for Linux, UNIX, and Windows によって必要なイメージのチェーンが完全に構築できない場合、リストア操作は終了し、エラー・メッセージが戻されます。このような場合は、自動増分リストアは行えないので、 **RESTORE DATABASE** コマンドを **INCREMENTAL ABORT** パラメーターとともに発行する必要があります。このようにするとすべての残っているリソースをクリーンアップしますので、手動増分リストアを行うことができます。

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

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

- **手動増分リストアの例**

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

  1.
     ```
     db2 restore database sample incremental taken at ts
     ```
     ここで、*ts* はリストア対象の最後の増分バックアップ・イメージ (ターゲット・イメージ) を指します。
  2.
     ```
     db2 restore database sample incremental taken at ts1
     ```
     ここで、*ts1* は最初の全データベース (または表スペース) イメージを指します。
  3.
     ```
     db2 restore database sample incremental taken at  tsX
     ```
     ここで、*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、イメージ *ts* まで (このイメージも含む) 各増分バックアップ・イメージをリストアしていきます。

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

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

自動増分リストアの制限

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

   例:

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

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

   推奨されている対処策: 手動増分リストアを使用してください。

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

   例:

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

   推奨されている対処策:

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

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

   例:

   ```
   db2 create db a
   db2 create db b

   db2 update db cfg for a using trackmod on

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

   db2 backup db b incremental -> ts2

   db2 restore db b incremental automatic taken at ts2
   ```

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

推奨されている対処策:

- 以下のようにして手動増分リストアを使用してください。

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

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

リカバリー・パフォーマンスの最適化

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

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

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

  DB2 データベース製品は、バッファー数、バッファー・サイズ、および並列処理設定の最適値を選択することで、バックアップまたはリストア操作の完了にかかる時間を自動的に最小限に抑えようとします。この値は、使用可能なユーティリティー・ヒープ・メモリーの大きさ、使用可能なプロセッサー数、およびデータベース構成に基づきます。

- リストア操作を完了するために必要な時間を短縮するためには、複数のソース装置を使用します。

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

のロールフォワードを希望する場合は、ログの最後までロールフォワードを実行し、表データが含まれるすべての表スペース間で整合性を保証するようにしてください。

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

- バックアップ操作とリストア操作には、以下の事柄が適用されます。
  - 複数の装置を使用してください。
  - 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。
- DB2 データベース製品では、複数のエージェントを使用して、 クラッシュ・リカバリーとデータベースのロールフォワード・リカバリーの両方を実行するようになりました。特に、対称マルチプロセッサー (SMP) マシンの場合、これらの操作の際にパフォーマンスが向上することを期待できます。データベースをリカバリーする際に複数のエージェントを使用すると、 SMP マシン上で使用可能な CPU が増えるという利点があります。

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

  DB2 データベース・マネージャー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。例えば、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。
- リカバリー操作を実行すると、DB2 データベース・マネージャーは、バッファー数、バッファー・サイズ、および並列処理設定の最適値を自動的に選択します。この値は、使用可能なユーティリティー・ヒープ・メモリーの大きさ、使用可能なプロセッサー数、およびデータベース構成に基づきます。したがって、システムに使用可能なストレージの量によっては、**util_heap_sz** 構成パラメーターを増やして、より多くのメモリーを割り振ることを検討する必要があります。

リカバリーの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャーの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。

RECOVER 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。

第 13 章 リストアの概要

DB2 **RESTORE DATABASE** コマンドの最も単純な形式の場合、必要な操作は、リストアするデータベースの別名を指定することだけです。例えば、以下のようにします。

```
db2 restore db sample
```

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

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

y を指定すると、リストア操作は正常に完了するはずです。

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

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

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

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

32 ビット・レベルでとったバックアップ・イメージから 64 ビット・レベルにデータベースをリストアすることはできますが、その逆はできません。

データベースのバックアップおよびリストアには、DB2 のバックアップおよびリストア・ユーティリティーを使用しなければなりません。ファイル・セットを 1 つのマシンから別のマシンに移動することは、データベースの保全性を失わせる可能性があるので、推奨されません。

特定の条件のもとでは、転送可能セットを **RESTORE DATABASE** コマンドで使用してデータベースを移動することができます。

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます: データベース・バックアップのリストア. タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、お

よびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

リストアの使用

メディアまたはストレージの障害、またはアプリケーションの障害などの問題の後、**RESTORE DATABASE** コマンドを使用して、データベースまたは表スペースをリカバリーします。データベースまたは個々の表スペースがバックアップされてある場合、何らかの理由でそれらが損傷を受けたり破損した場合に再作成することができます。

始める前に

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

このタスクについて

データベースは、ローカルとリモートのいずれかです。

リストア・ユーティリティーには、以下の制限が適用されます。

- リストア・ユーティリティーが使用できるのは、事前に DB2 バックアップ・ユーティリティーを使って、データベースのバックアップをとってある場合に限ります。
- インスタンスの所有者以外のユーザー (UNIX の場合)、または DB2ADMNS か管理者グループのメンバー以外のユーザー (Windows の場合) が、バックアップ・イメージをリストアしようとすると、エラー (SQL2061N) になります。他のユーザーがバックアップ・イメージにアクセスする必要がある場合、バックアップが生成された後にファイル許可を変更する必要があります。
- ロールフォワード処理の実行中にデータベースのリストア操作を開始することはできません。
- **TRANSPORT** オプションを指定しない場合、表スペースが既に存在して、しかもそれが同一の表スペースである場合に限り、表スペースを既存のデータベースにリストアできます。ここで「同一」とは、バックアップからリストア操作までの間に表スペースがドロップ/再作成されていないという意味です。ディスク上のデータベースとバックアップ・イメージ中のデータベースは同じでなければなりません。
- 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の表スペース・レベル・リストアを発行することはできません。
- システム・カタログ表が関係している表スペース・レベルのリストア操作をオンラインで実行することはできません。

- 単一データベース・パーティション環境で取られたバックアップを、既存のパーティション・データベース環境にリストアすることはできません。代わりに、バックアップは単一データベース・パーティション環境にリストアしなければならず、その後必要に応じてデータベース・パーティションを追加します。
- 1 つのコード・ページのバックアップ・イメージを、異なるコード・ページのシステムにリストアすると、システムのコード・ページは、バックアップ・イメージのコード・ページにより上書きされます。
- **RESTORE DATABASE** コマンドを使用して、非自動ストレージが有効である表スペースを自動ストレージが有効である表スペースに変換することはできません。
- **TRANSPORT** オプションを指定するときには、以下のような制約事項が適用されます。
  - バックアップ・イメージがリストア操作によってリストア可能で、しかもアップグレードがサポートされているならば、それを転送できます。
  - オンライン・バックアップが使われる場合、ソースおよびターゲット・データ・サーバーはどちらも同じ DB2 バージョンを実行している必要があります。
  - ターゲット・データベースに対して **RESTORE DATABASE** コマンドを発行する必要があります。リモート・クライアントのプラットフォームがサーバーと同じである場合、スキーマ転送をサーバーでローカルに実行することも、リモート・インスタンス接続を介して実行することもできます。ターゲット・データベースが、転送のローカル実行場所のインスタンスでカタログされたリモート・データベースである場合には、そのリモート・ターゲット・データベースに対するスキーマ転送はサポートされません。
  - 既存のデータベースへの表スペースとスキーマの転送だけが可能です。転送操作では、新しいデータベースは作成されません。データベースを新しいデータベースにリストアするには、**TRANSPORT** オプションを指定せずに **RESTORE DATABASE** コマンドを使用することができます。
  - DB2 のセキュリティー設定や権限によって、ソース・データベース内のスキーマが保護されている場合、ターゲット・データベースに転送されたスキーマもその同じ設定または権限を保持します。
  - パーティション・データベース環境では、転送はサポートされていません。
  - スキーマ内のいずれかの表に XML 列が含まれる場合、転送は失敗します。
  - **TRANSPORT** オプションは REBUILD オプションと両立しません。
  - スナップショット・バックアップ・イメージからのリストアでは **TRANSPORT** オプションがサポートされません。
  - ターゲット・データベースでは、データベース・リカバリーが使用可能になっている必要があります。
  - 転送用のステージング・データベースが作成されます。他の操作のためにこれを使用することはできません。
  - ステージング表とターゲット表のデータベース構成パラメーターを同じにする必要があります。同じでない場合は、非互換性エラーで転送操作が失敗します。
  - 無効としてリストされたオブジェクトを転送するには、ターゲット・データベースで **auto_reval** 構成パラメーターを deferred_force に設定する必要があります。これを行わない場合、転送が失敗します。

- オンライン・バックアップ・イメージが使われる場合、アクティブ・ログが含まれないと、転送操作が失敗します。
- 以前のリリースのバックアップ・イメージを使用する場合、それは完全なオフライン・データベース・レベルのバックアップ・イメージでなければなりません。
- ステージング・データベースまたはターゲット・データベースでエラーが発生した場合、リストア操作の全体を再発行する必要があります。発生したすべての障害はターゲット・サーバーの **db2diag** ログ・ファイルに記録されます。**RESTORE** コマンドを再発行する前に、これを確認してください。
- 転送クライアントに障害が発生した場合、ステージング・データベースが適切にクリーンアップされない可能性があります。この場合、ステージング・データベースをドロップする必要があります。ステージング・データベースのコンテナーによって後続の転送が妨げられるのを防ぐために、**RESTORE** コマンドを再発行する前にすべてのステージング・データベースをドロップしてください。
- 同じターゲット・データベースに対して同時に転送を実行することはサポートされていません。
- リダイレクト・リストア・スクリプトの生成は、表スペース転送ではサポートされません。

- ストレージ・グループが更新されている場合は、表スペースをリストアできます。表スペースのリストア中のターゲット・ストレージ・グループは、**RESTORE** が実行されたときにその表スペースが関連付けられていたストレージ・グループになります。
- 以前のストレージ・グループの関連付けへのポイント・イン・タイム・リカバリーは実行できません。

手順

リストア・ユーティリティーを起動するには、次のようにします。

- **RESTORE DATABASE** コマンドを実行します。
- db2Restore アプリケーション・プログラミング・インターフェース (API) を呼び出します。
- **RESTORE DATABASE** コマンドに関する IBM Data Studio のタスク・アシストを開きます。

例

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
```

関連情報:

スナップショットのバックアップ・イメージからのリストア

スナップショットのバックアップからのリストアは、ストレージ・デバイスの高速コピー・テクノロジーを使用して、リストアのデータのコピー部分を実行します。

始める前に

スナップショット・バックアップ操作およびリストア操作を実行するには、ご使用のストレージ・デバイス用の DB2 ACS API ドライバーが必要です。 IBM Data Server には、以下のストレージ・ハードウェアのための DB2 ACS API ドライバーが組み込まれています。

- IBM TotalStorage SAN ボリューム・コントローラー
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N シリーズ
- NetApp V-series
- NetApp FAS series

スナップショットのバックアップからリストアする前に、スナップショットのバックアップを実行する必要があります。 322 ページの『スナップショットのバックアップの実行』を参照してください。

手順

USE SNAPSHOT パラメーターを指定した **RESTORE DATABASE** コマンド、または SQLU_SNAPSHOT_MEDIA メディア・タイプを指定した db2Restore API を使用して、スナップショットのバックアップからリストアすることができます。

- **RESTORE DATABASE** コマンド:

  ```
  db2 restore db sample use snapshot
  ```

- db2Restore API:

  ```
  int sampleRestoreFunction( char dbAlias[],
                             char restoredDbAlias[],
                             char user[],
                             char pswd[],
                             char workingPath[] )
  {
    db2MediaListStruct mediaListStruct = { 0 };

    rmediaListStruct.locations = &workingPath;
    rmediaListStruct.numLocations = 1;
    rmediaListStruct.locationType = SQLU_SNAPSHOT_MEDIA;


    db2RestoreStruct restoreStruct = { 0 };

    restoreStruct.piSourceDBAlias = dbAlias;
    restoreStruct.piTargetDBAlias = restoredDbAlias;
    restoreStruct.piMediaList = &mediaListStruct;
    restoreStruct.piUsername = user;
    restoreStruct.piPassword = pswd;
    restoreStruct.iCallerAction = DB2RESTORE_STORDEF_NOINTERRUPT;


    struct sqlca sqlca = { 0 };
  ```

```
  db2Restore(db2Version900, &restoreStruct, &sqlca);

  return 0;
}
```

既存データベースへのリストア

データベース・レベルのリストアの場合、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。データベース・マネージャーは、データベースが作成されるときにシードを割り当てます。DB2 は、常にバックアップ・イメージからのシードを使用します。

表スペースを既存のデータベースにリストアできるのは、データベースに表スペースが存在していて、しかも両者が同じ表スペースである場合 (すなわち、バックアップを取ってからリストア操作を行うまでの間に表スペースをドロップして再作成していない場合) に限られます。

ディスク上のデータベースとバックアップ・イメージ中のデータベースは同じでなければなりません。表スペースをリストアする際に、現在定義されているストレージ・グループを変更したり、新規ストレージ・グループを明示的に作成したりすることはできません。

既存データベースにリストアする場合、リストア・ユーティリティーは以下のアクションを実行します。

- 既存のデータベースから表、索引、およびロング・フィールドのデータを削除し、バックアップのデータに置き換える。
- リストアしている表スペースごとに表項目を置き換える。
- リカバリー履歴ファイルを保持する (損傷していたり、項目がない場合を除く)。リカバリー履歴ファイルが損傷を受けているか、その中に項目がない場合は、データベース・マネージャーはバックアップ・イメージからファイルをコピーします。リカバリー履歴ファイルを置き換える場合は、**REPLACE HISTORY FILE** パラメーターを指定して **RESTORE DATABASE** コマンドを発行できます。
- 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。ディレクトリーは、データベースの位置、およびカタログされる方法を定義します。
- データベース・シードを比較する。シードが異なる場合、ユーティリティーは以下のアクションを実行します。
  - 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。
  - **NEWLOGPATH** パラメーターが指定されている場合は、**RESTORE DATABASE** コマンドの **NEWLOGPATH** パラメーターを、**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、ユーティリティーは以下のアクションを実行します。

– イメージがリカバリー不能データベースのものである場合はログを削除する。
– 現在のデータベース構成ファイルを保持する。
– **NEWLOGPATH** パラメーターが指定されている場合は、**RESTORE DATABASE** コマンドの **NEWLOGPATH** パラメーターを、**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、ユーティリティーは現在のログ・パスをデータベース構成ファイルにコピーする。ログのパスを確認する。データベースがそのパスを使用できない場合、ユーティリティーはデフォルトのログ・パスを使用するようにデータベース構成を変更する。

新規データベースへのリストア

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

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

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

テストおよび実稼働環境における増分リストアの使用

プロダクション・データベースの増分バックアップとリカバリーを使用可能にした後は、増分または差分バックアップ・イメージを使用してテスト用データベースを作成またはリフレッシュできます。

手動または自動増分リストアを使用してこのことを行えます。

バックアップ・イメージをプロダクション・データベースからテスト用データベースにリストアするには、**RESTORE DATABASE** コマンド上で INTO *target-database-alias* オプションを使用してください。例えば、以下のバックアップ・イメージを持つプロダクション・データベースがあるとします。

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

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

以下は、手動増分リストアの 1 例です。

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

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

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

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

自動増分リストア操作はデータベース履歴に依存しているため、テスト用データベースがあるかどうかによってリストアのステップは多少変わります。データベース TEST への自動増分リストアを実行するためには、その履歴がデータベース PROD のバックアップ・イメージ履歴を含んでいなくてはなりません。バックアップ・イメージのデータベース履歴は、以下のいずれかが当てはまるときに、既に存在しているデータベース TEST 用の任意のデータベース履歴を置き換えます。

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

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

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

リストア・ユーティリティーは、TEST データベースを作成し、それにデータを設定します。

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

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

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

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

```
connect to test
Database Connection Information

Database server            = server_id
SQL authorization ID       = id
Local database alias       = TEST

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

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

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

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

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

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

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

リダイレクト・リストア操作の実行

データベースのリストア操作では、データベースを再作成するために、データベースのバックアップ・イメージを使用します。

次の状況では、リダイレクト・リストア操作を使用します。

- ソース・マシンとは別のターゲット・マシンにバックアップ・イメージをリストアする場合
- 表スペース・コンテナーを異なる物理ロケーションにリストアする場合
- 1 つ以上のコンテナーがアクセス不能であるためにリストア操作に失敗した場合

- 定義済みのストレージ・グループのパスを再定義する場合

制約事項:

リダイレクト・リストアは、データを 1 つのオペレーティング・システムから別のオペレーティング・システムに移動するのに使うことはできません。

リストア処理中にストレージ・グループを作成したりドロップしたりすることはできません。

ストレージ・グループに関連付けられているすべての表スペースをリストアする場合でも、表スペースのリストア処理中にそのストレージ・グループのパスを変更することはできません。

増分バックアップ・イメージを使用したリダイレクト・リストアを実行するためのプロセスは、非増分バックアップ・イメージを使用したリダイレクト・リストアを実行するためのプロセスに類似しています。以下のいずれかの方法を使用します。

- **REDIRECT** パラメーターを指定して **RESTORE DATABASE** コマンドを発行し、データベースの増分リストアに使用するバックアップ・イメージを指定します。
- バックアップ・イメージからリダイレクト・リストア・スクリプトを生成してから、必要に応じてこのスクリプトを変更します。

RESTORE DATABASE コマンドのアプローチは、2 段階のデータベース・リストア・プロセスで、表スペース・コンテナーまたはストレージ・グループのパスを定義する段階が途中にあります。リダイレクト・リストアを実行するには、以下のようにします。

1. 次のように、**REDIRECT** パラメーターを指定して **RESTORE DATABASE** コマンドを発行する。
2. 以下のいずれかのステップを実行します。
   - **SET TABLESPACE CONTAINERS** コマンドを発行して、表スペース・コンテナーを定義する。
   - **SET STOGROUP PATHS** コマンドを発行して、リストアするデータベースのストレージ・グループ・パスを定義する。
3. 再び **RESTORE DATABASE** コマンドを発行します。今回は **CONTINUE** パラメーターを指定します。

RESTORE CONTINUE コマンドを発行すると、新しいパスが、関連するすべての表スペース用の表スペース・コンテナー・パスになります。**LIST TABLESPACE CONTAINERS** コマンドまたは **GET SNAPSHOT FOR TABLESPACES** コマンドを、**SET STOGROUP PATHS** の後、かつ **RESTORE CONTINUE** コマンドの前に発行すると、表スペース・コンテナー・パスの出力は、**SET STOGROUP PATHS** コマンドを使用して指定した新規パスを反映しません。

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

DB2 データベース製品は、表スペース・コンテナーの非自動ストレージ DMS 表スペース、および自動ストレージ表スペースのストレージ・グループ・パスを追加、

変更、または削除するための SQL ステートメントを提供します。リダイレクト・リストアは、非自動ストレージ SMS 表スペース・コンテナー構成を変更する唯一の方法です。

表スペース・コンテナーを再定義したり、またはストレージ・グループ・パスを変更したりするには、**RESTORE DATABASE** コマンドに **REDIRECT** パラメーターを指定して実行します。

表スペース・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。表スペース・コンテナー・パスをリダイレクトするのと似た方法で、バックアップ・イメージからデータ・ページをリストアする前に、データベースのストレージ・グループ構成を変更できます。バックアップ・イメージの生成後に、ストレージ・グループの名前が変更されている場合、**SET STOGROUP PATHS** コマンドに指定するストレージ・グループ名には、その新しい名前ではなく、バックアップ・イメージに含まれているストレージ・グループ名を使用します。

パーティション・データベース環境におけるリダイレクト・リストア操作の実行

パーティション・データベース環境では、リダイレクト・データベース・リストア時にストレージ・グループ・パスを新規ストレージ・パスにリダイレクトできるのは、カタログ・データベース・パーティションからリダイレクトする場合のみです。他のすべてのデータベース・パーティションの場合は、ストレージ・グループ・パスをカタログ・パーティションと同期させる必要があります。

カタログ・パーティション上でストレージ・グループ・パスを変更すると、すべての非カタログ・パーティションが RESTORE_PENDING 状態になります。ストレージ・グループ・パスをリダイレクトすると、他のすべてのデータベース・パーティションより先にカタログ・パーティションをリストアしなければなりません。カタログ・データベース・パーティションをリストアした後で、カタログ以外のデータベース・パーティションを、ストレージ・パスのリダイレクトなしで、並行してリストアできます。カタログ以外のデータベース・パーティションは、カタログ・データベース・パーティションに指定された新規のストレージ・グループ・パスを自動的に取得します。異なるデータベース (名前、インスタンス、またはシードが異なる) をリストアするときに、ストレージ・グループ・パスが暗黙的に変更される場合にも、新規ストレージ・グループ・パスが自動的に取得されます。

最後にバックアップを取った後にストレージ・グループ・パスを変更した場合も、そのバックアップ・イメージを使用して (ただし、ストレージ・グループ・パスは異なる)、任意のデータベース・パーティション上でリストアを実行できます。このリストアは、リダイレクト・リストアとはみなされません。このバックアップ・イメージからリストアすると、データベース・パーティションは、バックアップが作成されたときに定義されたストレージ・グループ・パスを一時的に使用するようになります。ロールフォワード・リカバリーを実行して、ストレージ・グループ・パスの変更を適用し、すべてのデータベース・パーティションを再同期します。

例

例 1

SET TABLESPACE CONTAINERS コマンドで表スペース・コンテナーを定義することにより、データベース SAMPLE に対して、表スペース・コンテナーのリダイレクト・リストアを実行できます。

```
db2 restore db sample redirect without prompting
SQL1277W A redirected restore operation is being performed.
During a table space restore, only table spaces being restored can
have their paths reconfigured. During a database restore, storage
group storage paths and DMS table space containers can be reconfigured.

DB20000I The RESTORE DATABASE command completed successfully.

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

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

例 2

SET STOGROUP PATHS コマンドを使用して、以下のように定義済みストレージ・グループのパスを再定義できます。

```
RESTORE DB SAMPLE REDIRECT

SET STOGROUP PATHS FOR sg_hot ON '/ssd/fs1', '/ssd/fs2'
SET STOGROUP PATHS FOR sg_cold ON '/hdd/path1', '/hdd/path2'

RESTORE DB SAMPLE CONTINUE
```

例 3

以下は、別名が MYDB であるデータベースの典型的な非増分リダイレクト・リストアのシナリオです。

1. 次のように、REDIRECT オプションを指定して RESTORE DATABASE コマンドを発行します。

   ```
   db2 restore db mydb replace existing redirect
   ```

2. 再定義するコンテナーを持つ表スペースごとに、SET TABLESPACE CONTAINERS コマンドを発行します。例えば、Windows 環境では次のようにします。

   ```
   db2 set tablespace containers for 5 using
          (file 'f:¥ts3con1'20000, file 'f:¥ts3con2'20000)
   ```

   リストアしたデータベースのコンテナーが、このステップで指定したものであることを検査するために、コンテナーの場所が再定義されているすべての表スペースに対して LIST TABLESPACE CONTAINERS コマンドを発行します。

3. ステップ 1 および 2 が正常終了した後、次を発行します。

   ```
   db2 restore db mydb continue
   ```

   これはリダイレクト・リストア操作の最終ステップです。

4. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアはステップ 1 から再始動できます。

注:

1. ステップ 1 が正常終了した後でステップ 3 が完了する前に、次を発行してリストア操作を打ち切ることができます。

   ```
   db2 restore db mydb abort
   ```

2. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアはステップ 1 から再始動できます。

例 4

以下は、別名が MYDB であり、以下のバックアップ・イメージを持つデータベースの、典型的な手動増分リダイレクト・リストアのシナリオです。

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

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

1. 次のように、INCREMENTAL および REDIRECT オプションを指定して RESTORE DATABASE コマンドを発行します。

   ```
   db2 restore db mydb incremental taken at <ts2> replace existing redirect
   ```

2. 再定義する必要があるコンテナーを持つ表スペースごとに、 SET TABLESPACE CONTAINERS コマンドを発行します。例えば、Windows 環境では次のようにします。

   ```
   db2 set tablespace containers for 5 using
          (file 'f:¥ts3con1'20000, file 'f:¥ts3con2'20000)
   ```

   リストアしたデータベースのコンテナーが、このステップで指定したものであることを検査するために、 LIST TABLESPACE CONTAINERS コマンドを発行します。

3. ステップ 1 および 2 が正常終了した後、次を発行します。

   ```
   db2 restore db mydb continue
   ```

4. 残りの増分リストア・コマンドは、以下のように発行することができるようになります。

   ```
   db2 restore db mydb incremental taken at <ts1>
   db2 restore db mydb incremental taken at <ts2>
   ```

   これはリダイレクト・リストア操作の最終ステップです。

注:

1. ステップ 1 が正常終了した後でステップ 3 が完了する前に、次を発行してリストア操作を打ち切ることができます。

   ```
   db2 restore db mydb abort
   ```

2. ステップ 3 が正常終了した後でステップ 4 のすべての必要なコマンドを実行する前に、次を発行してリストア操作を打ち切ることができます。

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

3. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアはステップ 1 から再始動できます。
4. いずれかのリストア・コマンドがステップ 4 で失敗した場合、失敗したコマンドを再実行してリストア・プロセスを継続できます。

例 5

以下は、同じデータベースでの典型的な自動増分リダイレクト・リストア・シナリオです。

1. 次のように、INCREMENTAL AUTOMATIC および REDIRECT オプションを指定して RESTORE DATABASE コマンドを発行します。

   ```
   db2 restore db mydb incremental automatic taken at <ts2>
          replace existing redirect
   ```

2. 再定義する必要があるコンテナーを持つ表スペースごとに、 SET TABLESPACE CONTAINERS コマンドを発行します。例えば、Windows 環境では次のようにします。

   ```
   db2 set tablespace containers for 5 using
          (file 'f:¥ts3con1'20000, file 'f:¥ts3con2'20000)
   ```

   リストアしたデータベースのコンテナーが、このステップで指定したものであることを検査するために、 LIST TABLESPACE CONTAINERS コマンドを発行します。

3. ステップ 1 および 2 が正常終了した後、次を発行します。

   ```
   db2 restore db mydb continue
   ```

   これはリダイレクト・リストア操作の最終ステップです。

注:

1. ステップ 1 が正常終了した後でステップ 3 が完了する前に、次を発行してリストア操作を打ち切ることができます。

   ```
   db2 restore db mydb abort
   ```

2. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアは以下を実行した後ステップ 1 から再始動できます。

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

自動生成スクリプトを使用してデータベースをリストアすることにより表スペース・コンテナーを再定義する

データベースをリストアする場合、リストア・ユーティリティーは、物理的コンテナーのレイアウトがデータベースがバックアップされた際のレイアウトと同一であると想定します。いずれかの物理的コンテナーのロケーションまたはサイズを変更する必要がある場合、**REDIRECT** オプションを指定して **RESTORE DATABASE** コマンドを発行しなければなりません。

このオプションを使用する際には、バックアップ・イメージに保管されている物理的コンテナーのロケーションを指定して、変更する非自動表スペースごとにコンテナーの完全セットを提供する必要があります。バックアップ時のコンテナー情報をキャプチャーすることはできますが、これは厄介である場合があります。

リダイレクト・リストアの実行を容易にするために、リストア・ユーティリティーでは、**REDIRECT** パラメーターおよび **GENERATE SCRIPT** パラメーターを指定して **RESTORE DATABASE** コマンドを発行することにより、既存のバックアップ・イメージからリダイレクト・リストア・スクリプトを生成することができます。リストア・ユーティリティーは、バックアップ・イメージを検討し、バックアップ・イメージからコンテナー情報を抽出し、詳細なコンテナー情報のすべてを組み込む CLP ス

クリプトを生成します。これで、スクリプト内のパスまたはコンテナー・サイズをどれでも変更し、CLP スクリプトを実行して新規のコンテナー・セットを伴うデータベースを再作成することができます。バックアップ・イメージしか持っていなくてコンテナーのレイアウトが不明な場合でも、生成したスクリプトを使用してデータベースをリストアできます。スクリプトはクライアントで作成されます。スクリプトをベースとして使用して、リストアされたデータベースがログ・ファイルおよびコンテナーのスペースを必要とする場所を判別し、それに従ってログ・ファイルおよびコンテナーのパスを変更することができます。

生成されたスクリプトは、以下の 4 つのセクションで構成されます。

初期化 最初のセクションは、コマンド・オプションを設定し、コマンドが実行されるデータベース・パーティションを指定します。以下に、最初のセクションの例を示します。

```
UPDATE COMMAND OPTIONS USING S ON Z ON SAMPLE_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM 0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
```

説明

- S ON は、コマンド・エラーが発生した場合にコマンドの実行を停止することを指定します。
- Z ON SAMPLE_NODE0000.out は、出力先が *dbalias*_NODE*dbpartitionnum*.out というファイルであることを指定します。
- V ON は、現在のコマンドが標準出力に表示されることを指定します。

パーティション・データベース環境でスクリプトを実行する場合、スクリプト・コマンドが実行されるデータベース・パーティションを指定することは重要です。

RESTORE DATABASE コマンド (REDIRECT パラメーターを指定)

2 番目のセクションは、**RESTORE DATABASE** コマンドを開始し、**REDIRECT** パラメーターを使用します。このセクションでは、**REDIRECT** パラメーターとともに使用できないパラメーターを除く、すべての **RESTORE DATABASE** コマンド・パラメーターを使用することができます。以下に、2 番目のセクションの例を示します。

```
RESTORE DATABASE SAMPLE
-- USER 'username'
-- USING 'password'
FROM '/home/jseifert/backups'
TAKEN AT 20050906194027
-- DBPATH ON 'target-directory'
INTO SAMPLE
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00001/LOGSTREAM0000/'
-- WITH num-buff BUFFERS
-- BUFFER buffer-size
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM n
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
```

表スペース定義

このセクションには、バックアップ・イメージ内の表スペースまたはコマンド行で指定された表スペースごとの表スペース定義が含まれます。表スペースごとに、表スペースの名前、タイプ、およびサービスに関する情報を含むコメント・ブロックで構成されるセクションが存在します。この情報は、表スペース・スナップショットと同じ形式で提供されます。提供される情報を使用して、表スペースに必要なサイズを決定することができます。自動ストレージを使用して作成された表スペースの出力を表示する場合、SET TABLESPACE CONTAINERS 節は表示されません。以下に、表スペース定義セクションの例を示します。

```
-- *********************************************************************
-- ** Tablespace name                         = SYSCATSPACE
-- **   Tablespace ID                         = 0
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Total number of pages                 = 5572
-- *********************************************************************
SET TABLESPACE CONTAINERS FOR 0
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0000.0'
);
-- *********************************************************************
-- ** Tablespace name                         = TEMPSPACE1
-- **   Tablespace ID                         = 1
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = System Temporary data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Total number of pages                 = 0
-- *********************************************************************
SET TABLESPACE CONTAINERS FOR 1
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0001.0'
);
-- *********************************************************************
-- ** Tablespace name                         = DMS
-- **   Tablespace ID                         = 2
-- **   Tablespace Type                       = Database managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Auto-resize enabled                   = No
-- **   Total number of pages                 = 2000
-- **   Number of usable pages                = 1960
-- **   High water mark (pages)               = 96
-- *********************************************************************
SET TABLESPACE CONTAINERS FOR 2
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   '/tmp/dms1'                                          1000
, FILE   '/tmp/dms2'                                          1000
);
```

RESTORE DATABASE コマンド (CONTINUE パラメーターを指定)

最終セクションでは、**CONTINUE** パラメーターを指定して **RESTORE DATABASE** コマンドを発行し、リダイレクト・リストアを完了します。以下に、最終セクションの例を示します。

```
RESTORE DATABASE SAMPLE CONTINUE;
```

自動生成スクリプトを使用したリダイレクト・リストアの実行

リダイレクト・リストア操作を実行するときは、バックアップ・イメージに保管されている物理コンテナーの場所を指定し、変更される各表スペースのすべてのコンテナーを提供する必要があります。

始める前に

リダイレクト・リストアを実行できるのは、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。

このタスクについて

- データベースが存在する場合、スクリプトを生成するには、データベースに接続できなければなりません。従って、データベースでアップグレードまたはクラッシュ・リカバリーが必要な場合は、リダイレクトした復元スクリプトを生成する前に、これらの操作を行う必要があります。
- パーティション・データベース環境で作業しており、ターゲット・データベースが存在しない場合、リダイレクトされた復元スクリプトをすべてのデータベース・パーティションで同時に生成するコマンドを実行することはできません。その代わり、カタログ・パーティションからはじめて、一度に 1 つのデータベース・パーティションで、リダイレクトされた復元スクリプトを生成するコマンドを実行する必要があります。

  別の方法として、ターゲット・データベースと同じ名前を持つダミーのデータベースを最初に作成することもできます。ダミーのデータベースを作成した後、すべてのデータベース・パーティションに、リダイレクトされた復元スクリプトを同時に生成することができます。
- スクリプト生成のための **RESTORE DATABASE** コマンドの発行時に **REPLACE EXISTING** パラメーターを指定しても、その **REPLACE EXISTING** パラメーターはスクリプトではコメント化されます。
- セキュリティー上の理由により、パスワードは、生成されたスクリプトに現れません。パスワードは手動で入力する必要があります。
- このリストア・スクリプトには、リストアするすべての表スペースに関するストレージ・グループの関連付け情報が入っています。

手順

スクリプトを使用してリダイレクト・リストアを実行するには、以下のようにします。

1. リストア・ユーティリティーを使用してリダイレクト・リストア・スクリプトを生成する。 リストア・ユーティリティーは、コマンド行プロセッサー (CLP)、または db2Restore アプリケーション・プログラミング・インターフェース

(API) を通して起動できます。 以下は、**REDIRECT** パラメーターと **GENERATE SCRIPT** パラメーターを指定した **RESTORE DATABASE** コマンドの例です。

```
db2 restore db test from /home/jseifert/backups taken at 20050304090733
      redirect generate script test_node0000.clp
```

これはクライアント上に test_node0000.clp というリダイレクト・リストア・スクリプトを作成します。

2. 必要な変更を行うために、テキスト・エディターでリダイレクト・リストア・スクリプトをオープンする。 変更できるのは、次のとおりです。
   - リストア・オプション
   - 自動ストレージ・パス
   - コンテナー・レイアウトおよびパス
3. 変更されたリダイレクト・リストア・スクリプトを実行します。 例えば、以下のようにします。

   ```
   db2 -tvf test_node0000.clp
   ```

異なるストレージ・グループ・パスを使用した実動データベースのクローン作成

実動データベースのクローンを、別のマシンを使用するテスト・データベース上に作成しなければならない場合があります。テスト・マシンと実動サーバーのストレージ・グループ・パスは異なっている場合が少なくありません。テスト・システムのパスには、最新で最も高速のストレージ・ディスクによる支援がない場合があります。

このタスクについて

実動データベース proddb のデータの一部がストレージ・グループ sg_hot にあり、そのパスは SSD ドライブ上にあるとします。このデータを、より低コストで、よりパフォーマンスの低いテスト・データベース testdb にリストアしようと考えています。テスト・システムには SSD ドライブがありませんが、それ以外のパスは同等です。リダイレクト・リストアを実行すると、その他のストレージ・グループを変更することなく、テスト・システムで sg_hot のパスを変更することができます。

手順

実動データベースからテスト・データベースにデータをリストアするには、次のようにします。

1. 実動データベースをバックアップします。 以下のコマンドを発行します。

   ```
   BACKUP DATABASE production_db TO /backup
   ```

   ここで、*production_db* は実動データベースです。
2. リダイレクト・リストアをテスト・データベースにセットアップします。 以下のコマンドを発行します。

   ```
   RESTORE DATABASE testdb REDIRECT
   ```

   ここで、*testdb* はテスト・データベースです。

3. テスト・データベースには使用可能な高速ストレージがないため、sg_hot のストレージ・パスを変更します。 以下のコマンドを発行します。

   ```
   SET STOGROUP PATHS FOR sg_hot ON '/hdd/path1', '/hdd/path2'
   ```

   ここで、*sg_hot* は、sg_hot ストレージ・グループです。
4. テスト・データベースのリストアを続行します。 以下のコマンドを発行します。

   ```
   RESTORE DATABASE testdb CONTINUE
   ```
5. 新しいパスに対応するようにストレージ・グループ名を更新します。 次のコマンドを使用します。

   ```
   CONNECT TO testdb
   RENAME STOGROUP sg_hot TO sg_cold
   ```

データベースの再ビルド

データベースの再ビルドは、一連のリストア操作を使用して、データベースまたはその表スペースのサブセットをリストアするプロセスです。データベースの再ビルドを使用して提供される機能により、DB2 データベース製品は用途が広がるとともにより堅固になり、一層完全なリカバリー・ソリューションを提供します。

表スペースのバックアップ・イメージからデータベースを再ビルドする機能により、大量のデータベース全体のバックアップを取る必要がなくなります。データベースのサイズが大きくなるにつれ、データベース全体のバックアップを取る機会は限られてきます。代わりに表スペース・バックアップを使用すると、データベース全体のバックアップを頻繁に取る必要がなくなります。表スペース・バックアップをより頻繁に取り、災害時にはログ・ファイルとともに、そうしたバックアップを使用するよう計画できます。

リカバリーの際、表スペースのサブセットを他の部分よりも早くオンラインにする必要がある場合、再ビルドを使用するとそのようにできます。表スペースのサブセットのみをオンラインにする機能は、テストや実稼働環境で特に有用です。

データベースの再ビルドには、実行する可能性のある一連の数多くのリストア操作が含まれています。再ビルド操作では、データベース・イメージ、または表スペース・イメージのいずれか、あるいはその両方を用いることが可能です。フル・バックアップまたは増分バックアップ、あるいはその両方を使用できます。最初のリストア操作により、リストア可能なデータベース構造 (表スペース・セット、ストレージ・グループ、データベース構成など) を定義するターゲット・イメージがリストアされます。リカバリー可能なデータベースの場合、再ビルドにより、接続可能で、オンラインにする必要のある表スペースのサブセットが含まれるデータベースを作成することができ、後ほどオフラインで表スペースをリカバリー可能な状態に維持することができます。

データベースを再ビルドするのに使用する方法は、リカバリー可能か、またはリカバリー不能かによって異なります。

- データベースがリカバリー可能な場合、以下のいずれかの方法を使用してください。
  - ターゲットとして、データベース全体のバックアップ・イメージ、または増分データベース・バックアップ・イメージ、あるいは表スペース・バックアッ

プ・イメージを使用し、**REBUILD** オプションを用いて、ターゲット・イメージのみから SYSCATSPACE と他の表スペースをリストアすることにより、データベースを再ビルドします。その後、データベースを特定の時点にロールフォワードできます。

- ターゲットとして、データベース全体のバックアップ・イメージ、または増分データベース・バックアップ・イメージ、あるいは表スペース・バックアップ・イメージを使用し、**REBUILD** オプションを用いて、リストアされるターゲット・イメージのその時点のデータベースで定義されている表スペースの集合を指定することにより、データベースを再ビルドします。SYSCATSPACE は、この集合の一部でなければなりません。この操作により、ターゲット・イメージで定義されている指定の表スペースがリストアされ、その後リカバリー履歴ファイルを使用して、ターゲット・イメージに含まれていない残りの表スペースで必要な他のバックアップ・イメージを自動的に検出してリストアします。リストアが完了したなら、データベースを特定の時点にロールフォワードします。

- データベースがリカバリー不能である場合、次のようにします。
  - ターゲットとして、データベース全体のバックアップ・イメージ、または増分データベース・バックアップ・イメージを使用し、適切な **REBUILD** 構文を用いて、ターゲット・イメージから SYSCATSPACE および他の表スペースをリストアすることにより、データベースを再ビルドします。リストアが完了すると、データベースに接続できます。

ターゲット・イメージの指定

データベースの再ビルドを実行するには、リストア操作のターゲットとして使用する最新のバックアップ・イメージを指定して **RESTORE** コマンドを発行することから始めます。このイメージは、再ビルド操作のターゲット・イメージとして認識されます。それは、リストア可能な表スペース、データベース構成、およびログ・シーケンスを含む、リストアされるデータベースの構造を定義するからです。再ビルドのターゲット・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。ターゲット・イメージは、すべてのタイプのバックアップ (全体、表スペース、増分、オンラインまたはオフライン) が可能です。ターゲット・イメージが作成された時点でデータベースに定義されていた表スペースは、データベースを再ビルドするのに使用できる表スペースとなります。

以下のいずれかの方法を使用して、リストアする表スペースを指定する必要があります。

- データベース内で定義されているリストア対象のすべての表スペースを指定して、除外する表スペースがある場合には例外リストを提供します。
- ターゲット・イメージ内にユーザー・データがあるリストア対象のすべての表スペースを指定して、除外する表スペースがある場合には例外リストを提供します。
- データベース内に定義されているリストア対象の表スペースのリストを指定します。

再ビルド・データベースに含める表スペースを判別したなら、**RESTORE** コマンドを適切な **REBUILD** オプションを使用して発行し、使用するターゲット・イメージを指定します。

再ビルド・フェーズ

適切な **REBUILD** オプションを使用して **RESTORE** コマンドを発行し、ターゲット・イメージが正常にリストアされた後、データベースは再ビルド・フェーズにあると見なされます。ターゲット・イメージのリストア後に生じる追加の表スペースのリストアにより、再ビルド・データベースで定義されているように既存の表スペースにデータがリストアされます。その後こうした表スペースは、再ビルド操作の完了時にデータベースにロールフォワードされます。

適切な **REBUILD** オプションを使用して **RESTORE** コマンドを発行し、データベースが存在しない場合には、ターゲット・イメージの属性に基づいて新しいデータベースが作成されます。データベースが存在する場合には、再ビルド・フェーズが開始されていることを通知する警告メッセージを受け取ります。再ビルド操作を継続するかどうかを尋ねられます。

再ビルド操作により、ターゲット・イメージからすべての初期メタデータがリストアされます。これには、データベースに属していて、表スペース・データまたはログ・ファイルには属さないすべてのデータが含まれます。初期メタデータの例は次のとおりです。

- 表スペース定義
- 履歴ファイル (管理操作を記録するデータベース・ファイルです)

再ビルド操作では、データベース構成もリストアされます。ターゲット・イメージはログ・チェーンを設定し、このログ・チェーンは再ビルド・フェーズで残りのリストアに使用できるイメージを判別します。同じログ・チェーンにあるイメージのみを使用できます。

ディスク上にデータベースがすでに存在し、ターゲット・イメージから履歴ファイルが生成されるようにするには、**REPLACE HISTORY FILE** オプションを指定してください。この時点でのディスク上の履歴ファイルは、データベースを再ビルドするのに必要な残りのイメージを検出するために自動ロジックによって使用されます。

ターゲット・イメージがリストアされると、以下のようになります。

- データベースがリカバリー可能な場合には、データベースはロールフォワード・ペンディング状態に置かれ、リストアするすべての表スペースもロールフォワード・ペンディング状態になります。データベースで定義済みであるもののリストアされていない表スペースは、リストア・ペンディング状態になります。
- データベースがリカバリー可能ではない場合には、リストアされたデータベースと表スペースは、通常の状態になります。リストアされない表スペースは、もはやリカバリーできないのでドロップ・ペンディング状態にされます。このタイプのデータベースの場合、再ビルド・フェーズが完了します。

リカバリー可能データベースの場合、最初の **ROLLFORWARD DATABASE** コマンドが発行される際に再ビルド・フェーズが終了し、ロールフォワード・ユーティリティーがログ・レコードの処理を開始します。ログ・レコードの処理の開始後にロールフォワード操作が失敗して、次にリストア操作が発行されると、そのリストアは再ビルド・フェーズの一部とは見なされません。こうしたリストアは、再ビルド・フェーズの一部ではない通常の表スペース・リストアと見なされます。

自動処理

ターゲット・イメージがリストアされた後、リストア・ユーティリティーは、リストアすることが必要な残りの表スペースが存在するかどうかを判別します。そうした表スペースがある場合、**REBUILD** オプションを使用して **RESTORE DATABASE** コマンドを実行するために使用されたのと同じ接続を使用して、それらはリストアされます。このユーティリティーはディスク上の履歴ファイルを使用して、リストアすることが必要な残りの各表スペースが含まれるターゲット・イメージの前に取得された最新のバックアップ・イメージを検出します。リストア・ユーティリティーは、履歴ファイルに保管されているバックアップ・イメージのロケーション・データを使用して、こうした各イメージを自動的にリストアします。表スペース・レベルのリストアであるこのような後続のリストアは、オフラインに限り実行できます。選択したイメージが現行のログ・チェーンに属していない場合には、エラーが戻されます。そのイメージからリストアされる各表スペースは、ロールフォワード・ペンディング状態になります。

リストア・ユーティリティーは、必要なすべての表スペースを自動的にリストアしようとします。場合によっては、履歴ファイルでの問題により一部の表スペースをリストアできないことや、いずれかの必要なイメージをリストアする際にエラーが生じることがあります。そのような場合、再ビルドを手動で完了するか、問題を訂正してから再ビルドを再び実行します。

自動再ビルドが正常に完了しない場合、リストア・ユーティリティーは残りのリストア・ステップのために集められた情報を診断ログ (**db2diag** ログ・ファイル) に書き込みます。この情報を使用して、再ビルドを手動で完了できます。

データベースが再ビルド中の場合、再ビルド・プロセスの一部である表スペースに属するコンテナーのみが獲得されます。

リダイレクトされたリストアによりコンテナーを再定義する必要がある場合、残りのリストアおよび後続のロールフォワード操作のために、新しいコンテナーの新しいパスとサイズを設定する必要があります。

残りのイメージのいずれかからリストアした表スペースのデータが新しいコンテナー定義に適合しない場合、その表スペースはリストア・ペンディング状態に置かれ、リストア終了時に警告メッセージが戻されます。この問題に関する追加情報は診断ログにあります。

再ビルド・フェーズの完了

対象の表スペースすべてがリストアされると、データベースの構成に基づいて 2 つのオプションがあります。データベースがリカバリー不能である場合、データベースは接続可能で、リストアされた表スペースはオンラインになります。ドロップ・ペンディング状態の表スペースはもはやリカバリーできず、その後にデータベースでバックアップを実行するとドロップされます。

データベースがリカバリー可能な場合には、ロールフォワード・コマンドを発行して、リストアされた表スペースをオンラインにできます。SYSCATSPACE がリストアされなかった場合、ロールフォワードは失敗し、この表スペースをリストアしなければロールフォワード操作を開始できません。つまり、再ビルド・フェーズで、SYSCATSPACE をリストアする必要があります。

注: パーティション・データベース環境では、SYSCATSPACE は非カタログ・パーティションには存在しないので、そうしたパーティションでは再ビルドできません。しかし、カタログ・パーティションでは、SYSCATSPACE は再ビルドされる表スペースの 1 つでなければならず、そうではない場合にはロールフォワード操作は成功しません。

データベースをロールフォワードすると、データベースはロールフォワード・ペンディング状態ではなくなり、ロールフォワード・ペンディング状態にある表スペースがロールフォワードされます。ロールフォワード・ユーティリティーは、リストア・ペンディング状態にある表スペースでは作動しません。

ロールフォワード操作の停止時刻は、再ビルド・フェーズ中にリストアされる最新のバックアップ・イメージの終了時刻より遅くなければなりません。それより前の時刻が設定されると、エラーが生じます。ロールフォワード操作がリストアされた最も古いイメージのバックアップ時刻に達することができない場合、ロールフォワード・ユーティリティーがデータベースを整合ポイントに持ち込むことはできず、ロールフォワードは失敗します。

ロールフォワード・ユーティリティーで使用可能な最も古いバックアップ・イメージと最新のバックアップ・イメージ間の時間フレームにわたるすべてのログ・ファイルがなければなりません。必要なログは、ターゲット・イメージ内の短縮形配列によって定義されているとおりの、最も古いバックアップ・イメージからターゲット・バックアップ・イメージまでのログ・チェーンに従うログで、そうでない場合にはロールフォワード操作は失敗します。ターゲット・イメージよりも新しいバックアップ・イメージが再ビルド・フェーズでリストアされた場合には、ターゲット・イメージから、リストアされた最新のバックアップ・イメージまでの追加ログが必要です。ログが使用可能になっていない場合、ログが網羅していないこうした表スペースを、ロールフォワード操作はリストア・ペンディング状態にします。**LIST HISTORY** コマンドを発行して、ロールフォワードで必要なログ範囲のリストア再ビルド項目を表示できます。

適正なログ・ファイルが使用可能でなければなりません。ロールフォワード・ユーティリティーに依存してログを取得する場合、DB2 ログ・マネージャーがログ・ファイルを取得するロケーションを示すよう構成されていることを確認しなければなりません。ログ・パスまたはアーカイブ・パスが変更された場合、**ROLLFORWARD DATABASE** コマンドの **OVERFLOW LOG PATH** オプションを使用する必要があります。

ROLLFORWARD DATABASE コマンドの **AND STOP** オプションを使用して、ロールフォワード・コマンドが正常に完了する際にデータベースを使用可能にします。この時点で、データベースはもうロールフォワード・ペンディング状態にはありません。ロールフォワード操作が開始し、正常に完了する前にエラーが発生する場合、失敗した時点でロールフォワード操作は停止して、エラーが戻ります。データベースは、ロールフォワード・ペンディング状態に留まります。問題を正すステップ (例えば、ログ・ファイルを修正する) を実行する必要があり、その後に別のロールフォワード操作を実行して処理を続けます。

エラーを修正できない場合には、**ROLLFORWARD STOP** コマンドを発行して、失敗した時点までデータベースを戻すことができます。ログ内のその時点より後のログ・データは、**STOP** オプションを一度使用するともう使用できなくなります。データベースはその時点に戻り、リカバリーされた表スペースはオンラインになります。まだ

リカバリーされていない表スペースは、リストア・ペンディング状態になります。データベースは、通常の状態になります。

リストア・ペンディング状態にある残りの表スペースをリカバリーするための最善の方法を決定しなければなりません。新しいリストアを実行してから表スペースのロールフォワードを行うか、再ビルド操作全体を再び実行することができます。それは、直面する問題のタイプにより異なります。SYSCATSPACE がリストア・ペンディング状態にある表スペースの 1 つである場合には、データベースは接続できません。

再ビルドと表スペース・コンテナー

再ビルドの際、再ビルド・プロセスの一部である表スペースに限ってコンテナーを獲得できます。表スペース・ユーザー・データがイメージからリストアされる際に、各表スペースに属するコンテナーが獲得されます。

ターゲット・イメージがリストアされると、バックアップ時にデータベースに認識される各表スペースの定義のみがリストアされます。つまり、再ビルドによって作成されたデータベースは、バックアップ時と同じ表スペースに関する情報を持ちます。ターゲット・イメージからリストアされたユーザー・データも有している必要のある表スペースの場合、そのコンテナーもその時点で獲得されます。

中間の表スペース・リストアでリストアされる残りの表スペースは、表スペース・データが含まれるイメージがリストアされる際に、それぞれのコンテナーを獲得します。

リダイレクトされたリストアを使用した再ビルド

リダイレクトされたリストアの場合、ターゲット・イメージのリストアの際にすべての表スペース・コンテナーが定義される必要があります。**REDIRECT** オプションを指定すると、ご使用の表スペース・コンテナーを再定義するために制御がユーザーに戻ります。**SET TABLESPACE CONTAINERS** コマンドを使用して表スペース・コンテナーを再定義した場合、そうした新しいコンテナーはその時点で獲得されます。再定義されていない表スペース・コンテナーは、表スペース・ユーザー・データがイメージからリストアされる際に、通常の方法で獲得されます。

リストアされる表スペース・データが新しいコンテナー定義に適合できない場合には、表スペースがリストア・ペンディング状態に置かれ、リストア終了時に警告 (SQL2563W) が戻されます。DB2 診断ログに問題について詳述したメッセージが生成されます。

再ビルドと TEMPORARY 表スペース

通常、DB2 バックアップ・イメージは、以下のコンポーネントから成っています。

- 表スペース定義、データベース構成ファイル、および履歴ファイルなどの、初期データベース・メタデータ
- **BACKUP** ユーティリティーに対して指定された非一時表スペース用データ
- ログ・ファイル・ヘッダーなどの最終データベース・メタデータ
- ログ・ファイル (**INCLUDE LOGS** オプションが指定された場合)

データベース・バックアップまたは表スペース・バックアップ、全体バックアップあるいは増分 (差分) バックアップであろうと、すべてのバックアップ・イメージにはこうしたコア・コンポーネントが必ずあります。

データベース・バックアップ・イメージには、以前にリストしたコンポーネントすべて、およびバックアップ時にデータベースに定義されるすべての表スペース用データが含まれます。

表スペース・バックアップ・イメージには必ず上記でリストされたデータベース・メタデータが含まれますが、バックアップ・ユーティリティーに対して指定される対象の表スペース用データに限り含まれます。

TEMPORARY 表スペースは、非一時表スペースとは別の方法で処理されます。TEMPORARY 表スペース・データは決してバックアップされませんが、データベースのフレームワークにとってこの存在はとても重要です。TEMPORARY 表スペース・データは決してバックアップされませんが、TEMPORARY 表スペースはデータベースの一部と見なされるので、バックアップ・イメージとともに保管されるメタデータの中で特別にマークされます。これにより、バックアップ・イメージ内にあるかのように見えます。加えて、表スペース定義には、TEMPORARY 表スペースの存在に関する情報が保持されます。

バックアップ・イメージには TEMPORARY 表スペースのデータは決して含まれませんが、(イメージのタイプに関わらず) ターゲット・イメージがリストアされるデータベース再ビルド操作の際、TEMPORARY 表スペースのコンテナーが取得されて割り振られるという意味でのみ、TEMPORARY 表スペースもリストアされます。コンテナーの取得および割り振りは、再ビルド・プロセスの一部として自動的に実行されます。ですから、データベースの再ビルド時に、TEMPORARY 表スペースを除外することはできません。

データベース再ビルド用ターゲット・イメージの選択

再ビルドのターゲット・イメージは、リストア操作の開始点として使用する最新のバックアップ・イメージでなければなりません。このイメージは、再ビルド操作のターゲット・イメージとして認識されます。それは、リストア可能な表スペース、データベース構成、およびログ・シーケンスを含む、リストアされるデータベースの構造を定義するからです。

すべてのタイプのバックアップ (全体、表スペース、増分、オンラインまたはオフライン) が可能です。

ターゲット・イメージはログ・シーケンス (ログ・チェーン) を設定し、このログ・シーケンスは再ビルド・フェーズの間に残りのリストアに使用できるイメージを判別します。同じログ・チェーンにあるイメージのみを使用できます。

以下の例は、再ビルド操作のターゲット・イメージとして使用するイメージを選択する方法を示しています。

以下の表スペースが含まれる SAMPLE という名前のデータベースがあるとします。

- SYSCATSPACE (システム・カタログ)
- USERSP1 (ユーザー・データ表スペース)

- USERSP2 (ユーザー・データ表スペース)
- USERSP3 (ユーザー・データ表スペース)

415 ページの図 22 には、取得された以下のデータベース・レベルのバックアップおよび表スペース・レベルのバックアップが発生順に示されています。

1. データベースのフル・バックアップ DB1
2. 表スペースのフル・バックアップ TS1
3. 表スペースのフル・バックアップ TS2
4. 表スペースのフル・バックアップ TS3
5. データベース・リストア、および TS1 と TS2 の間のポイントへのロールフォワード
6. 表スペースのフル・バックアップ TS4
7. 表スペースのフル・バックアップ TS5

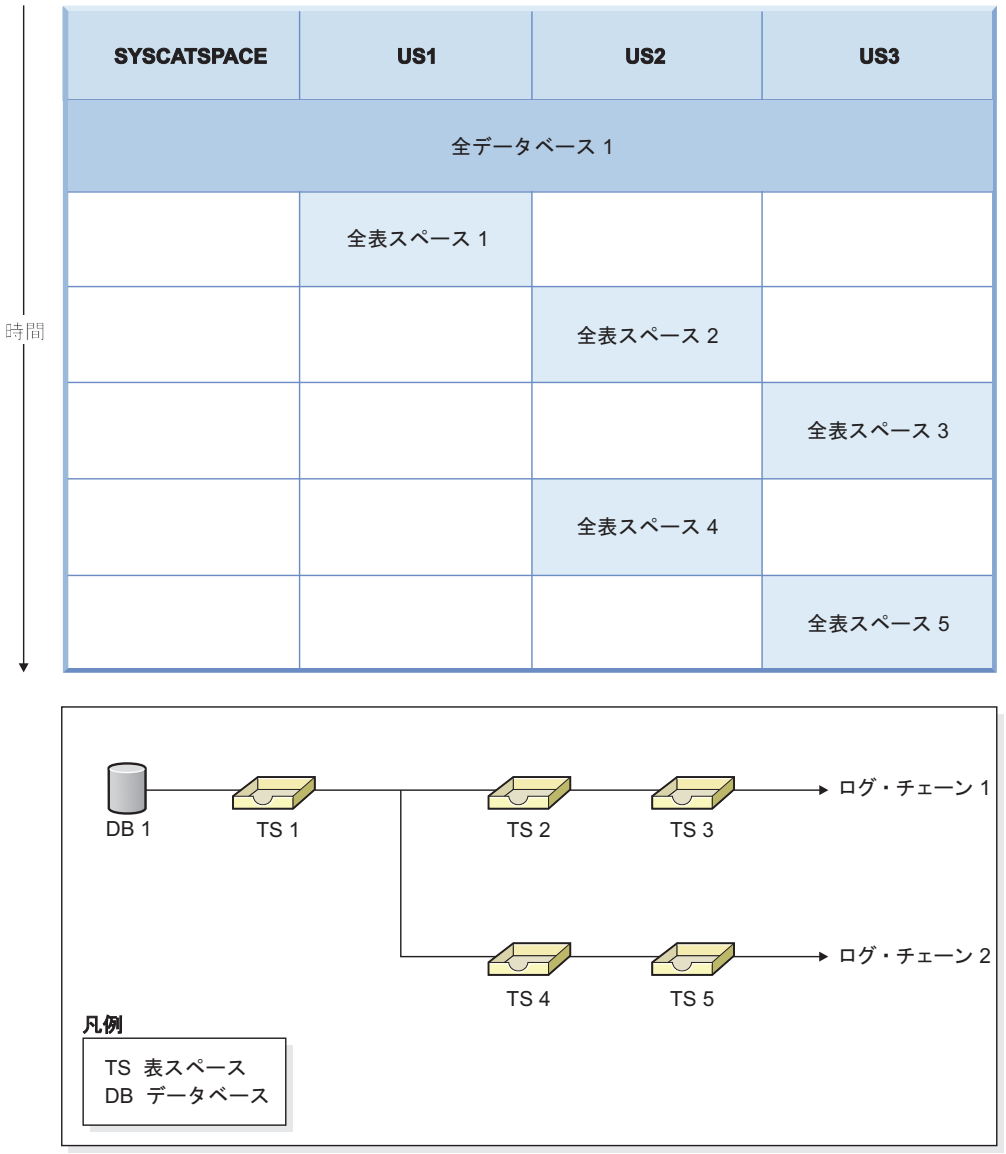


図 22. データベース SAMPLE のデータベース・レベルおよび表スペース・レベルのバックアップ

例 1

以下の例は、データベース SAMPLE を現行時点に再ビルドするために発行する必要のある CLP コマンドを示しています。まず、再ビルドする表スペースを選択する必要があります。データベースを現行時点に再ビルドすることが目的ですから、最新のバックアップ・イメージをターゲット・イメージとして選択する必要があります。最新のバックアップ・イメージは、ログ・チェーン 2 にあるイメージ TS5 です。

```
db2 restore db sample rebuild with all tablespaces in database taken at
   TS5 without prompting
db2 rollforward db sample to end of logs
db2 rollforward db sample stop
```

これにより、バックアップ・イメージ TS5、TS4、TS1 および DB1 が自動的にリストアされ、データベースはログ・チェーン 2 の終わりにロールフォワードされます。

注: ログ・チェーン 2 に属するすべてのログは、ロールフォワード操作を完了するためにアクセス可能でなければなりません。

例 2

2 番目の例は、ログ・チェーン 1 の終わりにデータベース SAMPLE を再ビルドするために発行する必要のある CLP コマンドを示しています。選択するターゲット・イメージはログ・チェーン 1 の最新のバックアップ・イメージである TS3 でなければなりません。

```
db2 restore db sample rebuild with all tablespaces in database
   taken at TS3 without prompting
db2 rollforward db sample to end of logs
db2 rollforward db sample stop
```

これにより、バックアップ・イメージ TS3、TS2、TS1 および DB1 が自動的にリストアされ、データベースはログ・チェーン 1 の終わりにロールフォワードされます。

注: ログ・チェーン 1 に属するすべてのログは、ロールフォワード操作を完了するためにアクセス可能でなければなりません。

再ビルド用の間違ったターゲット・イメージの選択

以下の表スペースが含まれる SAMPLE2 という名前のデータベースがあるとします。

- SYSCATSPACE (システム・カタログ)
- USERSP1 (ユーザー・データ表スペース)
- USERSP2 (ユーザー・データ表スペース)

417 ページの図 23 は、以下のバックアップで構成されている SAMPLE2 のバックアップ・ログ・チェーンを示しています。

1. BK1 は、すべての表スペースが含まれるデータベースのフル・バックアップ
2. BK2 は、USERSP1 の表スペースのフル・バックアップ
3. BK3 は、USERSP2 の表スペースのフル・バックアップ

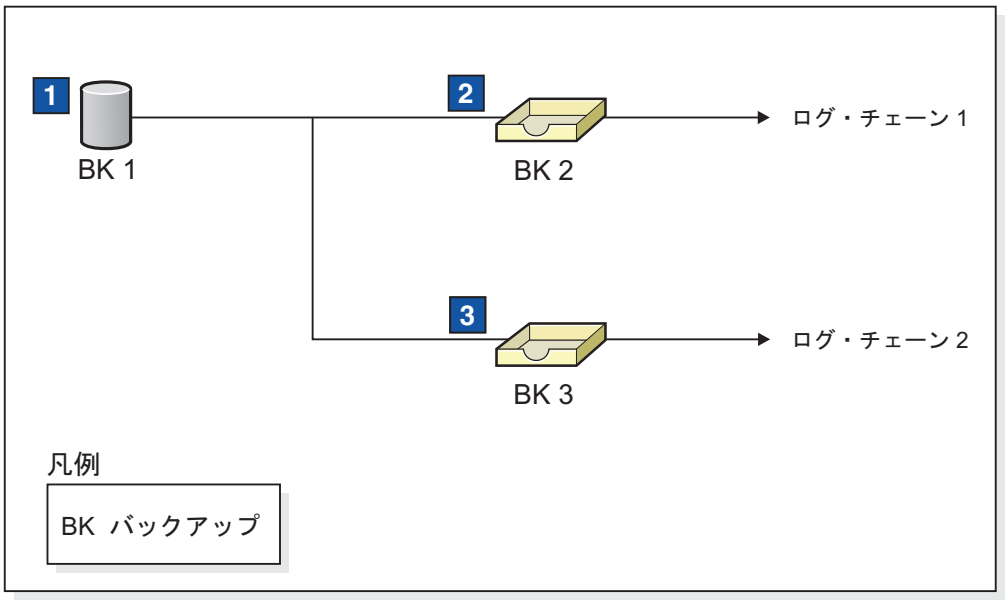


図 23. データベース SAMPLE2 のバックアップ・ログ・チェーン

以下の例は、表スペース SYSCATSPACE と USERSP2 を使用して、BK3 からデータベースを再ビルドするために発行する必要のある CLP コマンドを示しています。

```
db2 restore db sample2 rebuild with tablespace (SYSCATSPACE,
   USERSP2) taken at BK3 without prompting
```

このリストアの完了後、USERSP1 もリストアすることにしたとします。その場合は以下のコマンドを発行します。

```
db2 restore db sample2 tablespace (USERSP1) taken at BK2
```

このリストアは失敗し、BK2 が間違ったログ・チェーンから提供されていることを示すメッセージ (SQL2154N) が表示されます。 図 23 に表示されているように、USERSP1 のリストアに使用できるイメージは BK1 だけです。ですから、以下のコマンドを入力する必要があります。

```
db2 restore db sample2 tablespace (USERSP1) taken at BK1
```

これは成功し、それに従ってデータベースをロールフォワードすることができます。

選択済み表スペースを再ビルドする

データベースを再ビルドすることによって、元のデータベースを構成する表スペースのサブセットを含むデータベースをビルドすることができます。

このタスクについて

以下のような場合には、データベース内の表スペースのサブセットのみを再ビルドすることが役に立つ可能性があります。

- 表スペースのサブセットのみで作業を行うテスト環境および開発環境。

- 重要な表スペースを他よりも早くオンラインに戻す必要があるリカバリーの状況では、最初に表スペースのサブセットをリストアしてから、他の表スペースを後でリストアすることができます。

元のデータベースを構成する表スペースのサブセットを含むデータベースを再ビルドするには、以下の例を考慮してください。

この例では、以下の表スペースを含む SAMPLE というデータベースが存在します。

- SYSCATSPACE (システム・カタログ)
- USERSP1 (ユーザー・データ表スペース)
- USERSP2 (ユーザー・データ表スペース)
- USERSP3 (ユーザー・データ表スペース)

以下のバックアップが取られています。

- BK1 は SYSCATSPACE および USERSP1 のバックアップ
- BK2 は USERSP2 および USERSP3 のバックアップ
- BK3 は USERSP3 のバックアップ

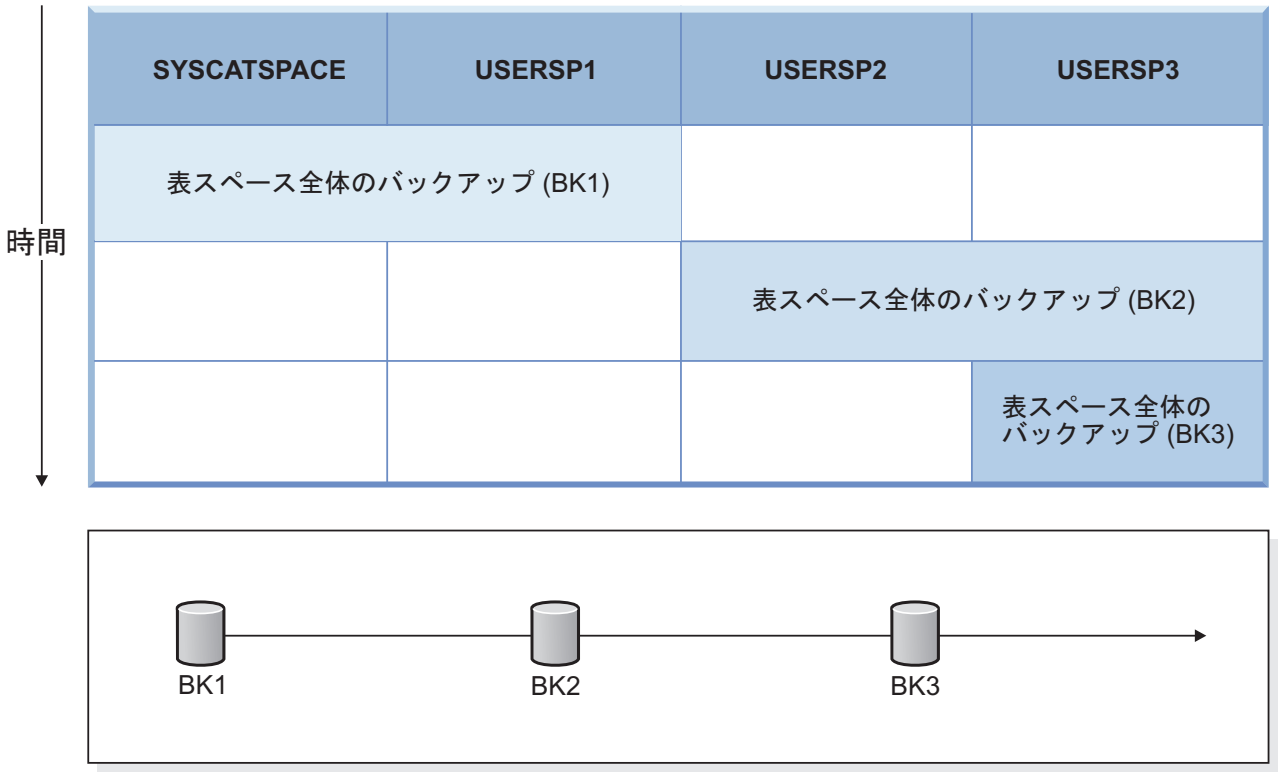


図 24. データベース SAMPLE に使用できるバックアップ・イメージ

以下は、 CLP を介して発行される **RESTORE DATABASE** コマンドおよび **ROLLFORWARD DATABASE** コマンドを使用して、SYSCATSPACE および USERSP1 のみをログの最後まで再ビルドする手順です。

```
db2 restore db mydb rebuild with all tablespaces in image
       taken at BK1 without prompting
db2 rollforward db mydb to end of logs
db2 rollforward db mydb stop
```

この時点で、データベースは接続可能であり、SYSCATSPACE および USERSP1 だけが NORMAL 状態になっています。 USERSP2 および USERSP3 はリストア・ペンディング状態になります。USERSP2 および USERSP3 は後でリストアすることができます。

再ビルドと増分バックアップ・イメージ

増分イメージを使用してデータベースを再ビルドできます。

デフォルトでは、リストア・ユーティリティーはすべての増分イメージに自動増分リストアを使用しようとします。つまり、**RESTORE DATABASE** コマンドの **INCREMENTAL** オプションを使用しなくても、ターゲット・イメージが増分バックアップ・イメージの場合、リストア・ユーティリティーは自動増分リストアを使用して再ビルド操作を実行します。ターゲット・イメージが増分イメージではなく、別の必要なイメージが増分イメージの場合には、リストア・ユーティリティーはこうした増分イメージが自動増分リストアを使用してリストアされるようにします。リストア・ユーティリティーは、**INCREMENTAL** オプションを **AUTOMATIC** オプションとともに指定するかどうかに関わらず、同じ方法で動作します。

INCREMENTAL オプションを指定して **AUTOMATIC** オプションを指定しない場合には、再ビルド・プロセス全体を手動で実行する必要があります。リストア・ユーティリティーは、通常の手動増分リストアで行うのと同じように、ターゲット・イメージから初期メタデータのみをリストアします。その後、必要な増分リストア・チェーンを使用して、ターゲット・イメージのリストアを完了する必要があります。それから、残りのイメージをリストアしてデータベースを再ビルドしなければなりません。

データベースを再ビルドするには、自動増分リストアを使用することが推奨されています。リストアが失敗した場合にのみ、手動の方法でデータベースの再ビルドを試行してください。

パーティション・データベースの再ビルド

パーティション・データベースを再ビルドするには、各データベース・パーティションを個別に再ビルドします。各データベース・パーティションごとに、カタログ・パーティションを初めとして、必要とするすべての表スペースをまずリストアします。リストアされない表スペースはすべてリストア・ペンディング状態になります。すべてのデータベース・パーティションがリストアされたら、カタログ・パーティションで **ROLLFORWARD DATABASE** コマンドを発行し、すべてのデータベース・パーティションをロールフォワードします。

このタスクについて

注: 再ビルド・フェーズに最初から組み込まれていなかった表スペースを後でリストアする必要がある場合、表スペースを以後ロールフォワードする際に、ロールフォワード・ユーティリティーによってデータベース・パーティション間のすべてのデータが同期化されていることを確認する必要があります。最初のリストアおよび

ロールフォワード操作時に表スペースが欠落している場合、データにアクセスしようとしてデータ・アクセス・エラーが発生するまで、表スペースが検出されない可能性があります。その場合、欠落している表スペースをリストアしてからロールフォワードし、パーティションの残りの部分と同期するようにしなければなりません。

表スペース・レベルのバックアップ・イメージを使用してパーティション・データベースを再ビルドするには、以下の例を考慮してください。

この例では、以下の 3 つのデータベース・パーティションを含む SAMPLE というリカバリー可能なデータベースが存在します。

- データベース・パーティション 1 には、表スペース SYSCATSPACE、USERSP1、および USERSP2 が含まれます。これはカタログ・パーティションです。
- データベース・パーティション 2 には、表スペース USERSP1 および USERSP3 が含まれます。
- データベース・パーティション 3 には、表スペース USERSP1、USERSP2、および USERSP3 が含まれます。

次のバックアップが取られています。BK*xy* は、パーティション *y* のバックアップ番号 *x* を表します。

- BK11 は、SYSCATSPACE、USERSP1、および USERSP2 のバックアップ
- BK12 は、USERSP2 および USERSP3 のバックアップ
- BK13 は、USERSP1、USERSP2、および USERSP3 のバックアップ
- BK21 は、USERSP1 のバックアップ
- BK22 は、USERSP1 のバックアップ
- BK23 は、USERSP1 のバックアップ
- BK31 は、USERSP2 のバックアップ
- BK33 は、USERSP2 のバックアップ
- BK42 は、USERSP3 のバックアップ
- BK43 は、USERSP3 のバックアップ

以下の手順では、CLP を介して発行される **RESTORE DATABASE** コマンドおよび **ROLLFORWARD DATABASE** コマンドを使用し、ログの末尾にデータベース全体を再ビルドすることについて例示しています。

手順

1. データベース・パーティション 1 で、次のように **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行します。

   ```
   db2 restore db sample rebuild with all tablespaces in database
      taken at BK31 without prompting
   ```

2. データベース・パーティション 2 で、次のように **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行します。

   ```
   db2 restore db sample rebuild with tablespaces in database
      taken at BK42 without prompting
   ```

3. データベース・パーティション 3 で、次のように **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行します。

```
db2 restore db sample rebuild with all tablespaces in database
   taken at BK43 without prompting
```

4. カタログ・パーティションで、**TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

```
db2 rollforward db sample to end of logs
```

5. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

```
db2 rollforward db sample stop
```

次のタスク

この時点で、データベースはすべてのデータベース・パーティションで接続可能であり、すべての表スペースは NORMAL 状態になっています。

データベース再ビルドの制約事項

次のリストは、データベース再ビルドの制約事項を要約しています。

- 再ビルドする表スペースの 1 つは、カタログ・パーティション上の SYSCATSPACE でなければなりません。
- コマンド行プロセッサー (CLP) を使用してコマンドを発行するか、対応するアプリケーション・プログラミング・インターフェース (API) を使用して、再ビルド操作を実行してください。
- オフライン・データベース・バックアップのイメージではない限り、**REBUILD** オプションをバージョン 9.1 より前のターゲット・イメージには使用できません。ターゲット・イメージがオフライン・データベース・バックアップの場合には、そのイメージ内の表スペースのみを再ビルドに使用できます。再ビルド操作が正常に完了した後に、そのデータベースをマイグレーションする必要があります。バージョン 9.1 より前の他のタイプのターゲット・イメージを使用して再ビルドを試行すると、エラーが生じます。
- ターゲット・イメージがデータベースのフル・バックアップではない限り、リストアしているのとは異なるオペレーティング・システムから **REBUILD** オプションをターゲット・イメージに対して発行することはできません。ターゲット・イメージがデータベースのフル・バックアップの場合には、そのイメージ内の表スペースのみを再ビルドに使用できます。リストアしているのとは異なるオペレーティング・システムからその他のタイプのターゲット・イメージを使用して再ビルドを試行すると、エラーが生じます。
- TRANSPORT オプションは REBUILD オプションと両立しません。

再ビルド・セッション - CLP の例

このトピックでは、再ビルド操作の例をいくつか示します。

シナリオ 1

以下の例では、MYDB という名前のリカバリー可能データベースがあり、それには次の表スペースが含まれています。

- SYSCATSPACE (システム・カタログ)

- USERSP1 (ユーザー・データ表スペース)
- USERSP2 (ユーザー・データ表スペース)
- USERSP3 (ユーザー・データ表スペース)

以下のバックアップが取られています。

- BK1 は SYSCATSPACE および USERSP1 のバックアップ
- BK2 は USERSP2 および USERSP3 のバックアップ
- BK3 は USERSP3 のバックアップ

例 1

以下は、データベース全体を最新の特定時点にまで再ビルドします。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行する。

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      taken at BK3 without prompting
   ```

2. **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to end of logs
   ```

3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

この時点で、データベースは接続可能であり、すべての表スペースは NORMAL 状態になっています。

例 2

以下は、SYSCATSPACE および USERSP2 だけを特定時点にまで再ビルドします (BK3 の終了点はその特定時点よりも前で、その特定時点はログの終了点よりも前です)。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行し、含める表スペースを指定します。

   ```
   db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP2)
      taken at BK2 without prompting
   ```

2. **TO PIT** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to PIT
   ```

3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

この時点で、データベースは接続可能であり、SYSCATSPACE および USERSP2 だけが通常状態になっています。 USERSP1 および USERSP3 は、RESTORE_PENDING 状態です。

後に、USERSP1 および USERSP3 を通常の表スペースのリストア (**REBUILD** オプションを指定しない) でリストアする方法は、次のとおりです。

1. 次のように、**REBUILD** オプションを*指定しないで* **RESTORE DATABASE** コマンドを発行し、リストアする表スペースを指定します。最初に、次のようにして USERSPI をリストアします。

   ```
   db2 restore db mydb tablespace (USERSP1) taken at BK1 without prompting
   ```

2. その後、次のように USERSP3 をリストアします。

   ```
   db2 restore db mydb tablespace taken at BK3 without prompting
   ```

3. **END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行し、リストアする表スペースを指定します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to end of logs tablespace (USERSP1, USERSP3)
   ```

   ロールフォワードは PIT までのすべてのログを再生し、これら 2 つの表スペースに対しては最初のロールフォワードの後に行われた作業がないため、そこで停止します。

4. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

例 3

以下は、SYSCATSPACE および USERSP1 だけをログの最後まで再ビルドします。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行する。

   ```
   db2 restore db mydb rebuild with all tablespaces in image
      taken at BK1 without prompting
   ```

2. **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to end of logs
   ```

3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

この時点で、データベースは接続可能であり、SYSCATSPACE および USERSP1 だけが NORMAL 状態になっています。 USERSP2 および USERSP3 は、RESTORE_PENDING 状態です。

例 4

以下の例では、バックアップの BK1 および BK2 はヒストリー・ファイルに示されている場所と同じ場所には存在しなくなっていますが、このことは再ビルドが発行されるときに知られていません。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行し、データベース全体を最新の特定時点に再ビルドするように指定します。

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      taken at BK3 without prompting
   ```

この時点で、ターゲット・イメージは正常にリストアされますが、必要なイメージが見つからなかったことを示すエラーがリストア・ユーティリティーから戻されます。

2. ここで、再ビルドを手動で完了させる必要があります。データベースは再ビルド・フェーズにあるので、これは以下のように行うことができます。

   a. 以下のように、**RESTORE DATABASE** コマンドを発行して BK1 バックアップ・イメージの場所を指定します。

      ```
      db2 restore db mydb tablespace taken at BK1 from location
         without prompting
      ```

   b. 以下のように、**RESTORE DATABASE** コマンドを発行して BK2 バックアップ・イメージの場所を指定します。

      ```
      db2 restore db mydb tablespace (USERSP2) taken at BK2 from
         location without prompting
      ```

   c. **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

      ```
      db2 rollforward db mydb to end of logs
      ```

   d. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

      ```
      db2 rollforward db mydb stop
      ```

この時点で、データベースは接続可能であり、すべての表スペースは NORMAL 状態になっています。

例 5

この例では、表スペース USERSP3 に特定のレポートを生成するために必要な独立データが含まれていますが、レポート生成により元のデータベースが妨げられないように希望しています。データにはアクセスしながら元のデータベースには影響を与えないようにするために、**REBUILD** を使用して、この表スペースと SYSCATSPACE だけで新規データベースを生成することができます。 SYSCATSPACE は、リストアおよびロールフォワード操作後にデータベースを接続可能にするためにも必要です。

SYSCATSPACE および USERSP3 内の最新データで新規データベースを構築するには、以下のようにします。

1. **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行し、表スペース SYSCATSPACE および USERSP3 が新規データベース NEWDB にリストアされるように指定します。

   ```
   db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP3)
      taken at BK3 into newdb without prompting
   ```

2. NEWDB に対して **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db newdb to end of logs
   ```

3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db newdb stop
   ```

この時点で、新規データベースは接続可能であり、SYSCATSPACE および USERSP3 だけが NORMAL 状態になっています。 USERSP1 および USERSP2 は、RESTORE_PENDING 状態です。

注: コンテナー・パスが現行のデータベースと新規データベースとの間に問題がある場合 (例えば、ファイル・システムが存在していないかまたはコンテナーが元のデータベースによりすでに使用中であるなどの理由で、元のデータベースのコンテナーを変更する必要がある場合)、リダイレクト・リストアを実行する必要があります。この例は、デフォルトの自動ストレージ・データベース・パスが表スペースに使用されることを前提としています。

シナリオ 2

以下の例では、SYSCATSPACE および T*xxxx* という名前のユーザー表スペースが 1000 個ある、MYDB と呼ばれるリカバリー可能データベースがあります。*xxxx* は表スペース番号を表します (T0001 など)。 1 つのフル・データベース・バックアップ・イメージ (BK1) があります。

例 6

以下は、T0999 および T1000 を除くすべての表スペースをリストアします。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行する。

```
db2 restore db mydb rebuild with all tablespaces in image except
   tablespace (T0999, T1000) taken at BK1 without prompting
```

2. **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

```
db2 rollforward db mydb to end of logs
```

3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

```
db2 rollforward db mydb stop
```

この時点で、データベースは接続可能となり、T0999 および T1000 を除くすべての表スペースは通常状態となります。 T0999 および T1000 はリストア・ペンディング状態となります。

シナリオ 3

このシナリオの例は、増分バックアップを使用してリカバリー可能データベースを再ビルドする方法を示しています。以下の例では、MYDB という名前のデータベースがあり、それには次の表スペースが含まれています。

- SYSCATSPACE (システム・カタログ)
- USERSP1 (データ表スペース)
- USERSP2 (ユーザー・データ表スペース)
- USERSP3 (ユーザー・データ表スペース)

以下のバックアップが取られています。

- FULL1 は、SYSCATSPACE、USERSP1、USERSP2、および USERSP3 の全バックアップ
- DELTA1 は、SYSCATSPACE および USERSP1 の差分バックアップ
- INCR1 は、USERSP2 および USERSP3 の増分バックアップ
- DELTA2 は、SYSCATSPACE、USERSP1、USERSP2、および USERSP3 の差分バックアップ
- DELTA3 は、USERSP2 の差分バックアップ
- FULL2 は、USERSP1 の全バックアップ

例 7

以下は、増分自動リストアを使用して、SYSCATSPACE および USERSP2 だけを最新の特定時点にまで再ビルドします。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行する。 **INCREMENTAL AUTO** オプションは、任意指定です。リストア・ユーティリティーは、イメージの細分度を検出し、必要であれば自動増分リストアを使用します。

   ```
   db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP2)
      incremental auto taken at DELTA3 without prompting
   ```

2. **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to end of logs
   ```

3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

この時点で、データベースは接続可能であり、SYSCATSPACE および USERSP2 だけが通常状態になっています。 USERSP1 および USERSP3 は、RESTORE_PENDING 状態です。

例 8

以下は、増分自動リストアを使用して、データベース全体を最新の特定時点にまで再ビルドします。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行する。 **INCREMENTAL AUTO** オプションは、任意指定です。リストア・ユーティリティーは、イメージの細分度を検出し、必要であれば自動増分リストアを使用します。

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      incremental auto taken at DELTA3 without prompting
   ```

2. **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to end of logs
   ```

3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

この時点で、データベースは接続可能であり、すべての表スペースは NORMAL 状態になっています。

例 9

以下は、USERSP3 を除くデータベース全体を最新の特定時点にまで再ビルドします。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行する。ターゲット・イメージは非増分イメージですが、リストア・ユーティリティーは必要な再ビルド・チェーンに増分イメージが含まれることを検出し、それらのイメージを自動的に増分的にリストアします。

   ```
   db2 restore db mydb rebuild with all tablespaces in database except
      tablespace (USERSP3) taken at FULL2 without prompting
   ```

2. **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to end of logs
   ```

3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

シナリオ 4

このシナリオの例は、ログ・ファイルを含むバックアップ・イメージを使用してリカバリー可能データベースを再ビルドする方法を示しています。以下の例では、MYDB という名前のデータベースがあり、それには次の表スペースが含まれています。

- SYSCATSPACE (システム・カタログ)
- USERSP1 (ユーザー・データ表スペース)
- USERSP2 (ユーザー・データ表スペース)

例 10

以下は、SYSCATSPACE および USERSP2 だけを持つデータベースを、最新の特定時点にまで再ビルドします。ログ・ファイルを含むフル・オンライン・データベース・バックアップ・イメージ (BK1) があります。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行する。

   ```
   db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP2)
      taken at BK1 logtarget /logs without prompting
   ```

2. **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (BK1 の終了後のすべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to end of logs overflow log path (/logs)
   ```

3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

この時点で、データベースは接続可能であり、SYSCATSPACE および USERSP2 だけが通常状態になっています。 USERSP1 は、RESTORE_PENDING 状態です。

例 11

以下は、データベースを最新の特定時点にまで再ビルドします。次の 2 つの、ログ・ファイルを含むフル・オンライン表スペース・バックアップ・イメージがあります。

- BK1 は、ログ・ファイル 10 から 45 を使用する SYSCATSPACE のバックアップ
- BK2 は、ログ・ファイル 64 から 80 を使用する、USERSP1 および USERSP2 のバックアップ

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行する。

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      taken at BK2 logtarget /logs without prompting
   ```

   ロールフォワード操作はログ・ファイル 10 から開始します。このログ・ファイルは、1 次ログ・ファイル・パスになければ、常にオーバーフロー・ログ・パスにあります。ログ範囲 46 から 63 は、どのバックアップ・イメージにも含まれていないので、ロールフォワードのために使用可能にする必要があります。
2. ログ・ファイル 64 から 80 のオーバーフロー・ログ・パスを使用し、**TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb to end of logs overflow log path (/logs)
   ```
3. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

この時点で、データベースは接続可能であり、すべての表スペースは NORMAL 状態になっています。

シナリオ 5

以下の例では、MYDB という名前のリカバリー可能データベースがあり、それには次の表スペースが含まれています。

- SYSCATSPACE (0)、SMS システム・カタログ (相対コンテナー)
- USERSP1 (1) DMS ユーザー・データ表スペース (絶対コンテナー /usersp2)
- USERSP2 (2) DMS ユーザー・データ表スペース (絶対コンテナー /usersp3)

以下のバックアップが取られています。

- BK1 は SYSCATSPACE のバックアップ
- BK2 は USERSP1 および USERSP2 のバックアップ
- BK3 は、USERSP2 のバックアップ

例 12

以下は、リダイレクトされたリストアを使用して、データベース全体を最新の特定時点にまで再ビルドします。

1. 次のように、**REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行する。

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      taken at BK3 redirect without prompting
   ```

2. 再定義するコンテナーを持つ表スペースごとに、**SET TABLESPACE CONTAINERS** コマンドを発行します。例えば、以下のようにします。

   ```
   db2 set tablespace containers for 3 using (file '/newusersp1' 10000)
   ```

3.

   ```
   db2 set tablespace containers for 4 using (file '/newusersp2' 15000)
   ```

4. 次のように、**CONTINUE** オプションを指定して **RESTORE DATABASE** コマンドを発行する。

   ```
   db2 restore db mydb continue
   ```

5. **TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、アクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to end of logs
   ```

6. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

この時点で、データベースは接続可能であり、すべての表スペースは NORMAL 状態になっています。

シナリオ 6

以下の例では、3 つのデータベース・パーティションを持つ MYDB という名前のデータベースがあります。

- データベース・パーティション 1 には、表スペース SYSCATSPACE、USERSP1、および USERSP2 が含まれます。これはカタログ・パーティションです。
- データベース・パーティション 2 には、表スペース USERSP1 および USERSP3 が含まれます。
- データベース・パーティション 3 には、表スペース USERSP1、USERSP2、および USERSP3 が含まれます。

次のバックアップが取られています。BK*xy* は、パーティション *y* のバックアップ番号 *x* を表します。

- BK11 は、SYSCATSPACE、USERSP1、および USERSP2 のバックアップ
- BK12 は、USERSP2 および USERSP3 のバックアップ
- BK13 は、USERSP1、USERSP2、および USERSP3 のバックアップ
- BK21 は、USERSP1 のバックアップ
- BK22 は、USERSP1 のバックアップ
- BK23 は、USERSP1 のバックアップ
- BK31 は、USERSP2 のバックアップ

- BK33 は、USERSP2 のバックアップ
- BK42 は、USERSP3 のバックアップ
- BK43 は、USERSP3 のバックアップ

例 13

以下は、データベース全体をログの最後にまで再ビルドします。

1. データベース・パーティション 1 で、次のように **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行します。

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      taken at BK31 without prompting
   ```

2. データベース・パーティション 2 で、次のように **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行します。

   ```
   db2 restore db mydb rebuild with tablespaces in database taken at
      BK42 without prompting
   ```

3. データベース・パーティション 3 で、次のように **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行します。

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      taken at BK43 without prompting
   ```

4. カタログ・パーティションで、**TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します (すべてのログが保管済みで、すべてのデータベース・パーティションからアクセス可能であることを前提としています)。

   ```
   db2 rollforward db mydb to end of logs
   ```

5. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

この時点で、データベースはすべてのデータベース・パーティションで接続可能であり、すべての表スペースは NORMAL 状態になっています。

例 14

以下は、SYSCATSPACE、USERSP1、および USERSP2 を最新の特定時点にまで再ビルドします。

1. データベース・パーティション 1 で、次のように **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行します。

   ```
   db2 restore db mydb rebuild with all tablespaces in database
      taken at BK31 without prompting
   ```

2. データベース・パーティション 2 で、次のように **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行します。

   ```
   db2 restore db mydb rebuild with all tablespaces in image taken at
      BK22 without prompting
   ```

3. データベース・パーティション 3 で、次のように **REBUILD** オプションを指定して **RESTORE DATABASE** コマンドを発行します。

   ```
   db2 restore db mydb rebuild with all tablespaces in image taken at
      BK33 without prompting
   ```

   注: このコマンドでは、再ビルド操作を完了するために必要な USERSP1 が省略されています。

4. カタログ・パーティションで、**TO END OF LOGS** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb to end of logs
   ```

5. 次のように、**STOP** オプションを指定して **ROLLFORWARD DATABASE** コマンドを発行します。

   ```
   db2 rollforward db mydb stop
   ```

   ロールフォワードは正常に終了し、データベースはすべてのデータベース・パーティションで接続可能です。 USERSP3 はそれが存在するすべてのデータベース・パーティションで RESTORE PENDING 状態であり、 USERSP1 はデータベース・パーティション 3 で RESTORE PENDING 状態ですが、これら以外のすべての表スペースは NORMAL 状態となります。

   データベース・パーティション 3 の USERSP1 にあるデータに対するアクセスを試行すると、データ・アクセス・エラーが生じます。これを修正するには、USERSP1 をリカバリーする必要があります。

   a. データベース・パーティション 3 で、次のように USERSP1 を含むバックアップ・イメージを指定して、**RESTORE DATABASE** コマンドを発行します。

      ```
      db2 restore db mydb tablespace taken at BK23 without prompting
      ```

   b. カタログ・パーティションで、次のように **TO END OF LOGS** オプションおよび **AND STOP** オプションを指定して、**ROLLFORWARD DATABASE** コマンドを発行します。

      ```
      db2 rollforward db mydb to end of logs on dbpartitionnum (3) and stop
      ```

   この時点で、データベース・パーティション 3 の USERSP1 が通常状態となったため、そのデータにアクセスすることができます。

シナリオ 7

以下の例では、MYDB という名前の*リカバリー不能*データベースがあり、それには次の表スペースが含まれています。

- SYSCATSPACE (0)、SMS システム・カタログ
- USERSP1 (1)、DMS ユーザー・データ表スペース
- USERSP2 (2)、DMS ユーザー・データ表スペース

データベースのバックアップは、BK1 の 1 つだけです。

例 15

以下は、リカバリー不能データベースで再ビルドを使用する方法を示しています。

次のようにして、SYSCATSPACE および USERSP1 だけを使用してデータベースを再ビルドします。

```
db2 restore db mydb rebuild with tablespace (SYSCATSPACE, USERSP1)
   taken at BK1 without prompting
```

リストアの後に、データベースは接続可能になります。 **LIST TABLESPACES** コマンドまたは MON_GET_TABLESPACE 表関数を実行する場合、

SYSCATSPACE および USERSP1 が NORMAL 状態となるのに対して、USERSP2 は DELETE_PENDING/OFFLINE 状態になります。これで、NORMAL 状態の 2 つの表スペースで作業を行えるようになります。

データベースのバックアップを取る場合、まず DROP TABLESPACE ステートメントを使用して USERSP2 をドロップする必要があります。そのようにしない場合、バックアップは失敗します。

後に USERSP2 をリストアするには、BK1 からデータベース・リストアを再発行する必要があります。

リストア操作の進行状況をモニターする

LIST UTILITIES コマンドを使用して、データベースでのリストア操作をモニターできます。

手順

LIST UTILITIES コマンドを発行して、**SHOW DETAIL** パラメーターを指定します。

```
LIST UTILITIES SHOW DETAIL
```

タスクの結果

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

例

次に示すのは、リストア操作でのパフォーマンスをモニターするときの出力例です。

```
ID                                   = 6
Type                                 = RESTORE
Database Name                        = SAMPLE
Partition Number                     = 0
Description                          = db
Start Time                           = 08/04/2011 12:24:47.494191
State                                = Executing
Invocation Type                      = User
Progress Monitoring:
      Completed Work                 = 4096 bytes
      Start Time                     = 08/04/2011 12:24:47.494197
```

リストアのパフォーマンスの最適化

リストア操作を実行すると、DB2 データベース製品は、バッファー数、バッファー・サイズ、および並列処理設定の最適値を自動的に選択します。この値は、使用可能なユーティリティー・ヒープ・メモリーの大きさ、使用可能なプロセッサー数、およびデータベース構成に基づきます。

したがって、システムに使用可能なストレージの量によっては、**util_heap_sz** 構成パラメーターを増やして、より多くのメモリーを割り振ることを検討する必要があります。目的は、リストア操作の完了にかかる時間を最小限に抑えることです。次の **RESTORE DATABASE** コマンド・パラメーターの値を明示的に入力しないと、DB2 データベース製品側が値を選択します。

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

リストア操作の場合、バックアップ操作によって使用されるバッファー・サイズの倍数が常に使用されます。 **RESTORE DATABASE** コマンドを発行するときにバッファー・サイズを指定できますが、それは必ずバックアップ・バッファー・サイズの倍数にする必要があります。

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

- リストア・バッファー・サイズを大きくする。

  リストア・バッファー・サイズは、バックアップ操作中に指定したバックアップ・バッファー・サイズに正の整数を乗算したサイズでなければなりません。誤ったバッファー・サイズを指定すると、割り振られるバッファーは、許容可能な最小のサイズになります。
- バッファー数を増やします。

  指定する値は、バックアップ・バッファーに指定したページ数の倍数でなければなりません。ページ数の最小値は 8 です。
- **PARALLELISM** パラメーターの値を増やす。

  これにより、リストア操作中にデータベースへの書き込みのために使用されるバッファー・マニピュレーター (BM) の番号が増加します。
- ユーティリティー・ヒープ・サイズを大きくする。

  これにより、他のユーティリティーが同時に使用できるメモリーが増加します。

リストアの使用に必要な特権、権限、および許可

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャーの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。

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

データベース・スキーマ転送

データベース・スキーマの転送操作には、データベースのバックアップ・イメージを取得すること、および既存の別のデータベースにデータベース・スキーマをリストアすることが含まれます。データベース・スキーマを転送すると、転送されるスキーマ内のデータベース・オブジェクトは、新しいデータベースを参照するように再作成されて、新しいデータベースにデータがリストアされます。

データベース・スキーマは、全体として転送する必要があります。 1 つの表スペースの中に、転送したいスキーマと、もう 1 つ別のスキーマが含まれる場合、両方のスキーマのすべてのデータ・オブジェクトを転送する必要があります。他のデータベース・スキーマを参照しないこれらのスキーマのセットを、*転送可能セット* といいます。転送可能セットに含まれる表スペース内のデータとスキーマ内の論理オブジェクトは、その転送可能セットの中の表スペースとスキーマだけを参照します。例えば、表には、その転送可能セットに含まれる他の表に対する表従属関係だけが存在します。

以下の図は、いくつかの表スペースとスキーマを持つデータベースを示しています。図の中で、スキーマによって参照される表スペースはスキーマの上に示されています。複数の表スペースを参照するスキーマや、複数のスキーマによって参照される表スペースもあります。

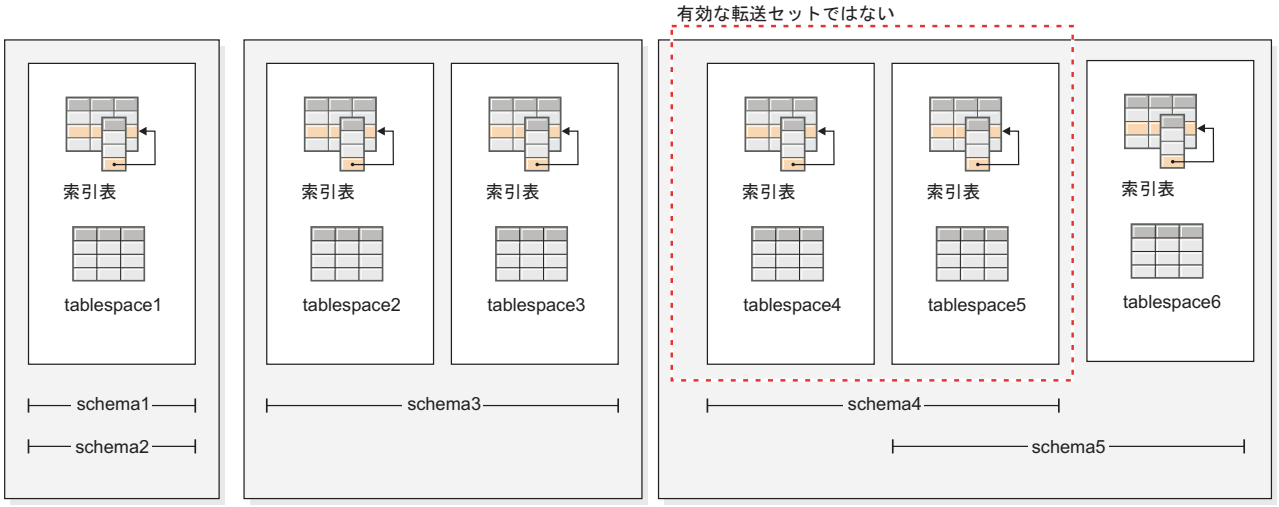


図 25. 表スペースとスキーマのセット

以下に示す表スペースとスキーマの組み合わせは、有効な転送可能セットです。

- tablespace1 と schema1 および schema2
- tablespace2 および tablespace3 と schema3
- tablespace4、tablespace5、tablespace6 と schema4 および schema5
- 次のように、複数の有効な転送可能セットの組み合わせもまた、1 つの有効な転送可能セットとなります。
  - tablespace1、tablespace2、tablespace3 と schema1、schema2、schema3

tablespace4 および tablespace5 と schema4 から成るセットは有効な転送可能セットではありません。tablespace5 と schema5 の間、および schema5 と tablespace6 の

間に参照が存在するためです。このセットが有効な転送可能セットになるためには、tablespace6 と schema5 が含まれる必要があります。

データベース・スキーマを転送するには、**TRANSPORT** パラメーターを指定する **RESTORE** コマンドを使用できます。

データベース・スキーマを転送するときには、転送操作の一環として一時データベースが作成され、それに名前が付けられます。ターゲット・データベース上に再作成する目的で論理オブジェクトをバックアップ・イメージから抽出するために、転送ステージング・データベースが使用されます。バックアップ・イメージにログが含まれている場合、ステージング・データベースのトランザクション整合性をとるために、そのログも使用されます。その後、転送される表スペースの所有権がターゲット・データベースに移されます。

データベース・スキーマの転送によって再作成されるデータベース・オブジェクトに関する考慮事項

データベース・スキーマの転送によるデータベース・オブジェクトの再作成に関する以下の情報について、検討してください。

table blah

データベース・オブジェクト	スキーマ転送時の考慮事項
SQL ルーチン (SQL を使用する外部ルーチンではない)	SQL ルーチンの新規コピーがターゲット・データベースに作成されます。 SQL ストアード・プロシージャーの場合、ストアード・プロシージャー・バイトコードの追加コピーが新しいデータベースに作成されるため、追加のカタログ・スペースが消費されます。
外部ルーチン	ルーチンごとに新しいカタログ項目が作成されます。このカタログ項目は、元のソース・ルーチンと同じバイナリー・ファイルを参照します。 **RESTORE** コマンドは、ソース・システムから外部ルーチン・バイナリー・ファイルをコピーしません。
アクセスの問題を発生させる状態にあるソース表	バックアップ・イメージ生成時に正常な状態ではなかった表 (例えばチェック・ペンディング状態の表やロード・ペンディング状態の表) のデータは、ターゲット・データベースでアクセス不能になる場合があります。これを避けるには、スキーマ転送の前に、ソース・データベース内で表を正常な状態に移すことができます。
データ・キャプチャー属性を含む表	データ・キャプチャーが使用可能になったソース表はデータ・キャプチャー属性と共にターゲット・データベースに転送されて、データベース間のデータ複製情報を引き続きログに記録します。ただし、複製される表はこの表から情報を抽出しません。ユーザーのオプションとして、**RESTORE** コマンド完了後にレプリケーション・ソースとして動作する新しいターゲット表を登録することができます。
ラベル・ベースのアクセス制御 (LBAC) を使用する表	LBAC によって保護されたデータを転送するときには、転送操作によってターゲット・データベース上に LBAC オブジェクトが再作成されます。同じ名前の LBAC オブジェクトがターゲット・データベースに存在する場合、転送操作が失敗します。制限付きデータ・アクセスを損なわないようにするために、転送操作では、ターゲット・データベース上の既存の LBAC オブジェクトを使用しません。

表スペースを転送すると、特殊なフォーマットのログ・レコードが、ターゲット・データベース上に作成されます。このフォーマットは、従来の DB2 バージョンでは読み取ることができません。表スペースを転送した後に、DB2 バージョン 9.7 フィックスパック 2 より前のバージョンにダウングレードすると、転送された表スペースを含むターゲット・データベースは、リカバリー不能になります。このようなターゲット・データベースを古いバージョンの DB2 に対応させるには、転送操作前の時点に、ターゲット・データベースをロールフォワードします。

重要: データベースのロールフォワードで表スペース・スキーマ転送ログ・レコードが検出された場合、対応する転送済み表スペースはオフラインになってドロップ・ペンディング状態に変わります。転送された表スペースとその内容を再作成するための転送済み表スペースの完全なログが、データベースにないためです。転送完了後にターゲット・データベースのフルバックアップを取ることができるので、その後のロールフォワードがログ・ストリーム内のスキーマ転送のポイントを通過することはありません。

転送可能オブジェクト

バックアップ・イメージからターゲット・データベースにデータを転送する場合、結果として生じるアクションは主に 2 つあります。リストア対象の表スペースの物理オブジェクトと論理オブジェクトがターゲット・データベース内に再作成されて、表スペース定義とコンテナーがターゲット・データベースに追加されます。

以下の論理オブジェクトが再作成されます。

- 表、作成済みグローバル一時表、およびマテリアライズ照会表
- 通常ビューおよび統計ビュー
- 以下の種類の生成済み列:
  - 式
  - ID
  - 行変更タイム・スタンプ
  - 行変更トークン
- ユーザー定義関数、および生成済み関数
- 関数とプロシージャー (外部ルーチン実行可能ファイルを除く)
- ユーザー定義タイプ
- 以下の種類の制約:
  - チェック
  - 外部キー
  - 機能の従属関係
  - 1 次
  - ユニーク
- 索引
- トリガー
- 順序
- オブジェクトの許可、特権、セキュリティー、アクセス制御、および監査構成
- 表の統計、プロファイル、およびヒント

- パッケージ

スキーマの以下のコンポーネントは、ターゲット・データベース上に作成されません。

- 別名
- 作成済みグローバル変数
- 外部ルーチン実行可能ファイル
- 関数マッピングとテンプレート
- 階層表
- 索引拡張
- ジョブ
- メソッド
- ニックネーム
- OLE DB 外部関数
- 範囲パーティション表
- サーバー
- ソース派生プロシージャー
- 構造化タイプ
- システム・カタログ
- 型付き表および型付きビュー
- 使用量リスト
- ラッパー

転送の例

TRANSPORT オプションを指定した **RESTORE DATABASE** コマンドを使用すると、表スペースと SQL スキーマから成るセットを 1 つのデータベースから別のデータベースにコピーできます。

以下の例では、バックアップ・イメージのソースとして ORIGINALDB という名前のデータベース、およびターゲット・データベースとして TARGETDB を使用します。

以下の図は、ORIGINALDB の表スペースとスキーマを示しています。

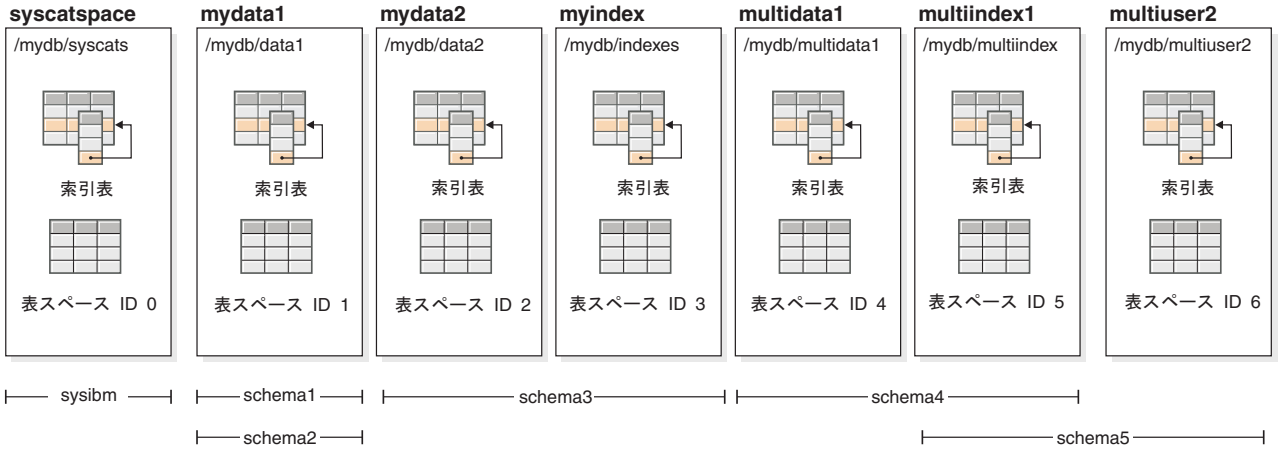


図 26. ORIGINALDB データベース

originalDB データベースには、次のような有効な転送可能セットが含まれています。

- mydata1; schema1 + schema2
- mydata2 + myindex; schema3
- multidata1 + multiindex1 + multiuser2; schema4 + schema5
- 次のように、複数の有効な転送可能セットの組み合わせもまた、1 つの有効な転送可能セットとなります。
  - mydata1 + mydata2 + myindex; schema1 + schema + schema3

以下の図は、TARGETDB の表スペースとスキーマを示しています。

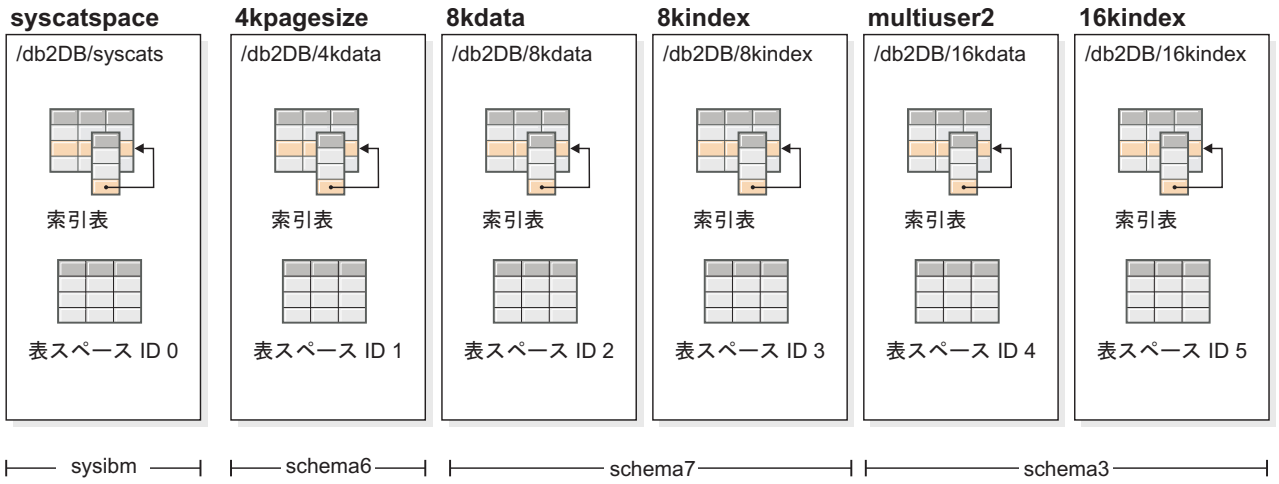


図 27. TARGETDB データベース

同じスキーマ名を持つスキーマ、または表スペース名を持つ表スペースがソースおよびターゲット・データベースに含まれる場合、そのスキーマまたは表スペースをターゲット・データベースに転送することはできません。ターゲット・データベース上のスキーマまたは表スペースと同じ名前のスキーマまたは表スペースを含む転

送操作を発行した場合、その転送操作は失敗します。例えば、以下のようなグループは有効な転送可能セットですが、これをターゲット・データベースに直接転送することはできません。

- mydata2 + myindex; schema3 (schema3 はソースおよびターゲット・データベースの両方に存在する)

データベース内のすべての表スペースを含んでいる ORIGINALDB の単一のオンライン・バックアップ・イメージが存在する場合、これが転送のソースとなります。また、表スペース・レベルのバックアップ・イメージにも、これが該当します。

転送される表スペースのコンテナー・パスをリダイレクトすることができます。データベース相対パスが使用された場合には、これが特に重要です。

例

例 1: mydata1 表スペースのスキーマ schema1 および schema2 を、TARGETDB に正常に転送します。

```
db2 restore db originaldb tablespace (mydata1) schema(schema1,schema2)
  from <Media_Target_clause> taken at <date-time>
  transport into targetdb redirect
db2 list tablespaces
db2 set tablespace containers for <tablespace ID for mydata1>
 using (path '/db2DB/data1')

db2 restore db originaldb continue
```

この結果、TARGETDB には mydata1 表スペースと schema1 および schema2 が含まれるようになります。

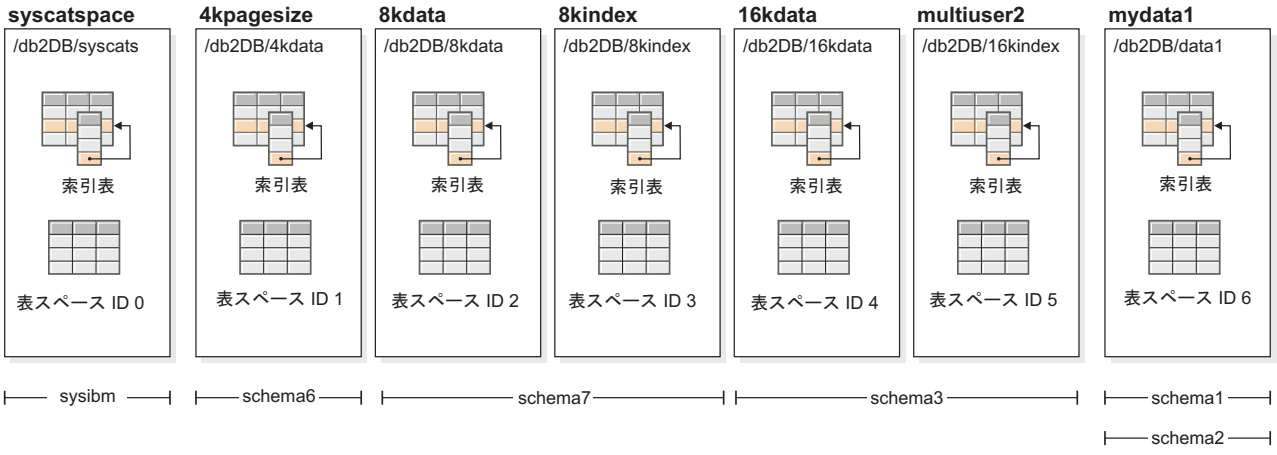


図 28. 転送後の TARGETDB データベース

例 2: mydata2 および myindex 表スペースのスキーマ schema3 を TARGETDB に転送します。ターゲット・データベースに既に存在するスキーマは、転送できません。

```
db2 restore db originaldb tablespace (mydata2,myindex) schema(schema3)
  transport into targetdb
```

スキーマ schema3 は既にターゲット・データベースに存在するため、この転送操作は失敗します。 TARGETDB の状態に変更はありません。 SQLCODE=SQL2590N rc=3 です。

例 3: multidata1、multiindex1、および multiuser2 表スペースのスキーマ schema4 および schema5 を TARGETDB に転送します。ターゲット・データベースに既に存在する表スペースは、転送できません。

```
db2 restore db originaldb tablespace (multidata1,multiindex1,multiuser2)
  schema(schema4,schema5) transport into targetdb
```

表スペース multiuser2 は既にターゲット・データベースに存在するため、この転送操作は失敗します。 TARGETDB の状態に変更はありません。
SQLCODE=SQL2590N rc=3 です。

例 4: myindex 表スペースを TARGETDB に転送します。スキーマを部分的に転送することはできません。

```
db2 restore db originaldb tablespace (myindex) schema(schema3)
  transport into targetdb
```

転送対象の表スペースおよびスキーマのリストが、有効な転送可能セットではありません。転送操作が失敗して、TARGETDB は不変のままです。
SQLCODE=SQL2590N rc=1 です。

例 5: syscatspace 表スペースを TARGETDB にリストアします。システム・カタログは転送できません。

```
db2 restore db originaldb tablespace (syscatspace) schema(sysibm)
  transport into targetdb
```

システム・カタログは転送できないため、この転送操作は失敗します。
SQLCODE=SQL2590N rc=4 です。 transport オプションを指定しない RESTORE DATABASE コマンドによって、ユーザー定義表スペースを転送したり、システム・カタログをリストアしたりできます。

例 6: システムに存在しないターゲット・データベースには、リストアできません。

```
db2 restore db originaldb tablespace (mydata1) schema(schema1,schema2)
  transport into notexists
```

この転送操作は失敗します。存在しないターゲット・データベースには、表スペースを転送できません。

トラブルシューティング: スキーマの転送

ステージング・データベースまたはターゲット・データベースでエラーが発生した場合、リストア操作の全体を再実行する必要があります。発生したすべての障害はターゲット・サーバーの db2diag log ファイルに記録されます。 **RESTORE** コマンドを再発行する前に、**db2diag** ログを確認してください。

エラーの処理

リストア中に発生するエラーは、コピー対象のオブジェクトの種類や転送の段階に応じてさまざまな方法で扱われます。電源障害などの状況によっては、すべてのデータがクリーンアップされるとは限りません。

転送操作は、次のような段階から成ります。

- ステージング・データベースの作成
- 物理的な表スペース・コンテナーのリストア
- ロールフォワード処理
- スキーマ妥当性検査
- 表スペース・コンテナーの所有権の転送
- ターゲット・データベースでのスキーマ再作成
- ステージング・データベースのドロップ (ただし **STAGE IN** パラメーターが指定されていない場合)

物理オブジェクトの転送に関して、スキーマの再作成段階の最後で何らかのエラーがログに記録されている場合には、リストア操作が失敗してエラーが戻されます。ターゲット・データベース上のオブジェクト作成はすべてロールバックされ、ステージング・データベースで内部的に作成されたすべての表がクリーンアップされます。発生する可能性のあるすべてのエラーを **db2diag** ログ・ファイルに記録できるようにするため、ロールバックは再作成段階の最後に発生します。コマンドを再発行する前に、戻されたすべてのエラーを調査することができます。

成功または失敗の後、ステージング・データベースは自動的にドロップされます。ただし **STAGE IN** パラメーターが指定されている場合には、失敗した場合でもドロップされません。そのステージング・データベースの名前を再使用する前に、ステージング・データベースをドロップする必要があります。

第 14 章 ロールフォワードの概要

次の例のように、最も単純な形式の **ROLLFORWARD DATABASE** コマンドで必要なのは、ロールフォワード・リカバリーするデータベースの別名を指定することだけです。

```
db2 ROLLFORWARD DB sample
```

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます: データベースのロールフォワード. タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、およびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

ロールフォワード・リカバリーを実行する際に使用できる方法の 1 つを以下に示します。

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。

   ログの最後までのリカバリーを指定する場合、回復したポイント・イン・タイムが期待した時点より早いと、 **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知らせることがあります。

   ポイント・イン・タイム指定リカバリーを指定する場合、 **QUERY STATUS** オプションはロールフォワード操作が確実に正しいポイント・イン・タイムに完了するようにするのに役立ちます。
3. **STOP** オプションを指定してロールフォワード・ユーティリティーを起動する。操作の停止後は、追加変更項目をロールフォワードすることはできません。

ロールフォワード・リカバリーを実行する際に使用できる代替方法を以下に示します。

1. **AND 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。
2. 追加のステップを行う必要があるかどうかは、ロールフォワード操作の結果に応じて異なる。
   - 正常に実行された場合は、ロールフォワードは完了し、データベースは接続可能で使用可能になります。この時点で、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。
   - エラーが戻された場合は、何であれ問題を修正するのに必要な処置を取ります。例えば、ログ・ファイルがない場合はログ・ファイルを検索し、検索エラーの場合はログ・アーカイブが作動していることを確認します。それから、 **AND 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再発行します。

データベースは (restore ユーティリティーを使用し) ロールフォワードする前に正常にリストアしておく必要があります。ただし、表スペースについてはその必要はありません。表スペースを一時的にロールフォワード・ペンディング状態にすることができます。ただし、その状態を取り消すためにリストア操作を行う必要はありません (例えば、電源割り込みの後など)。

ロールフォワード・ユーティリティーが起動されると、次のようになります。

- データベースがロールフォワード・ペンディング状態の場合、データベースはロールフォワードされます。データベース・バックアップ・イメージの後に取得されたバックアップ・イメージからリストアされた表スペースのうち、現在ロールフォワード・ペンディング状態のものもロールフォワードされます。データベース・レベル・バックアップの前に取得された表スペースのうち、データベース・レベル・バックアップがリストアされた後にリストアされたものはすべて、ロールフォワード・ペンディング状態のままになります。これらをリカバリーするには、表スペース・レベルのロールフォワードをさらに実行する必要があります。
- データベースはロールフォワード・ペンディング状態にない が、表スペースはロールフォワード・ペンディング状態にある 場合は以下のようになります。
  - 表スペースのリストを指定すると、それらの表スペースだけがロールフォワードされます。
  - 表スペースのリストを指定しないと、ロールフォワード・ペンディング状態の表スペースすべてがロールフォワードされます。

データベース・ロールフォワード操作は、オフラインで実行されます。ロールフォワード操作が正常に完了するまでは、データベースは使用不可です。また、ユーティリティーの起動時に **STOP** オプションを指定していないと操作は完了できません。

表スペース・ロールフォワード操作は、オフラインで実行できます。ロールフォワード操作が正常に完了するまでは、データベースは使用不可です。ログの最後まで行われた場合、またはユーティリティーの起動時に **STOP** オプションを指定していた場合、正常に完了します。

SYSCATSPACE が含まれていない限り、*オンライン* で表スペースのロールフォワード操作を実行することができます。表スペースに対してオンラインでロールフォワード操作を実行するときは、その表スペース自体は使用できませんが、データベース内の他の表スペースは使用できます。

データベースを初めて作成した時点では、循環ロギングだけが使用可能になります。つまり、ログは保管やアーカイブされるのではなく再使用されます。循環ロギングでは、ロールフォワード・リカバリーは不可能です。クラッシュ・リカバリーまたはバージョン・リカバリーだけが行えます。アーカイブ・ログは、バックアップを取った後に生じるデータベースへの変更を文書化します。ログ・アーカイブ (およびロールフォワード・リカバリー) を使用可能にするには、 **logarchmeth1** データベース構成パラメーターを、デフォルトの OFF 以外の値に設定します。**logarchmeth1** を OFF 以外の値に設定すると、データベースはバックアップ・ペンディング状態になり、データベースを再び使用するにはデータベースのオフライン・バックアップを作成する必要があります。

注: 項目は、ロールフォワード操作で使用されるログ・ファイルごとに、リカバリー履歴ファイルに作成されます。
この場合、コマンドの戻りは以下のようになります。

パーティション・データベース環境および DB2 pureScale環境では、この状況情報は、データベース・パーティションまたはメンバーごとに返されます。

```
db2 rollforward db mydb to end of logs

                               Rollforward Status

 Input database alias                   = mydb
 Number of members have returned status = 3

 Member ID Rollforward   Next log      Log files processed        Last committed transaction
           status        to be read
 --------- ------------  ------------  -------------------------  -------------------------
         0  DB  working  S0000001.LOG  S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC
         1  DB  working  S0000010.LOG  S0000000.LOG-S0000009.LOG  2009-05-06-15.28.20.000000 UTC
         2  DB  working  S0000005.LOG  S0000000.LOG-S0000004.LOG  2009-05-06-15.27.33.000000 UTC

DB20000I  ROLLFORWARD コマンドが正常に完了しました。
```

ロールフォワードの使用

ROLLFORWARD DATABASE コマンドを使用して、データベース・ログ・ファイルに記録されたトランザクションを、リストアされたデータベース・バックアップ・イメージまたは表スペース・バックアップ・イメージに適用します。

始める前に

ロールフォワード・リカバリーを実行するデータベースに接続していてはなりません。ロールフォワード・ユーティリティーは指定されたデータベースに自動的に接続を確立し、この接続はロールフォワード操作が完了すると終了します。

このタスクについて

進行中のロールフォワード操作をキャンセルせずに表スペースをリストアすることはしないでください。そうしてしまうと、ある表スペースはロールフォワード進行状態にあり、別の表スペースはロールフォワード・ペンディング状態にある、という状態になることがあります。進行中のロールフォワード操作は、ロールフォワード進行状態にある表スペース上でのみ作動します。

データベースは、ローカルとリモートのいずれかです。

ロールフォワード・ユーティリティーには、以下の制限が適用されます。

- ロールフォワード操作は一度に 1 つしか起動することができません。リカバリーする表スペースが多数ある場合は、それらすべてを同一の操作に指定することができます。
- 最後に実行したバックアップの後で表スペースの名前を変更した場合、その表スペースをロールフォワードする際には必ず新しい名前を使用してください。以前の表スペースの名前は認識されません。
- 実行中のロールフォワード操作をキャンセルすることはできません。完了したロールフォワード操作のみキャンセルすることができます。ただし、**STOP** パラメーター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 ROLLFORWARD 操作のタイム・スタンプより前のタイム・スタンプを指定して、表スペースのロールフォワード操作を特定のポイント・イン・タイムまで「継続する」ことはできません。特定のポイント・イン・タイムが指定されない場合は、直前の ROLLFORWARD 操作のタイム・スタンプのポイント・イン・タイムが使用されます。 STOP を指定するだけで、指定したポイント・イン・タイムで終了するロールフォワード操作を実行することができます。ただし、これは関係する表スペースがすべて同一のオフライン・バックアップ・イメージからリストアされた場合にのみ可能です。この場合、ログ処理は必要ありません。進行中のロールフォワード操作が完了する前またはキャンセルされる前に、別のロールフォワード操作を別の表スペース・リストで開始すると、エラー・メッセージ (SQL4908) が戻されます。 **LIST TABLESPACES** コマンドをすべてのデータベース・パーティション上で実行 (または MON_GET_TABLESPACE 表関数を使用) し、現在ロールフォワード中 (ロールフォワード進行状態) の表スペース、およびロールフォワード作動可能 (ロールフォワード・ペンディング状態) の表スペースを判別してください。次の 3 つのオプションがあります。
  - すべての表スペースに対する進行中のロールフォワード操作を終了する。
  - 表スペースのサブセットに対するロールフォワード操作を終了する。 (ロールフォワード操作が特定のポイント・イン・タイムまで継続するようになっている場合、すべてのデータベース・パーティションがかかわるため、このオプションは使用できないことがあります。)
  - 進行中のロールフォワード操作をキャンセルする。
- パーティション・データベース環境では、ロールフォワード・ユーティリティーはデータベースのカタログ・パーティションから起動する必要があります。
- 表スペースのポイント・イン・タイム・ロールフォワードは、DB2 バージョン 9.1 クライアントで導入されました。表スペースをポイント・イン・タイムにロールフォワードするには、すべてのクライアントを、バージョン 10.1 にアップグレードする必要があります。
- ログは以前のリリースのバージョンからロールフォワードすることはできません。

手順

ロールフォワード・ユーティリティーを起動するには、以下を使用します。

- **ROLLFORWARD DATABASE** コマンド、または
- db2Rollforward アプリケーション・プログラミング・インターフェース (API)。
- **ROLLFORWARD DATABASE** コマンドに関する IBM Data Studio のタスク・アシストを開きます。

例

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
```

関連情報:

表スペースにおける変更のロールフォワード

データベースが順方向リカバリーであれば、データベース全体ではなく、表スペースをバックアップしたり、リストアしたり、ロールフォワードしたりできます。個

別の表スペースについてリカバリー計画を決めておくこともでき、これにより時間が節約されます。つまり、データベース全体をリカバリーするよりは、データベースの一部をリカバリーした方が時間は短縮されるからです。例えば、ディスクが不良で、そのディスクに表スペースが 1 つしか含まれていない場合、その表スペースのみをリストアおよびロールフォワードすることができます。その際データベース全体をリカバリーする必要はなく、データベースの残りの部分に対するユーザー・アクセスに影響を与えることもありません。ただし、損傷した表スペースにシステム・カタログ表が含まれている場合は別です。その状態ではデータベースに接続することができません。 (システム・カタログ表を含む表スペース・レベルのバックアップ・イメージが使用可能である場合、システム・カタログ表スペースはそれだけでリストアできます。)表スペース・レベルのバックアップにより、データベースの重要な部分を他の部分より頻繁にバックアップすることも可能になり、これによりデータベース全体のバックアップより時間は短縮されます。

表スペースがリストアされた後は、常にロールフォワード・ペンディング状態になります。表スペースを使用可能にするためには、表スペースにロールフォワード・リカバリーを実行する必要があります。ログの最後までロールフォワードすることもでき、特定の時点までロールフォワードすることもできます。ただし、システム・カタログ表を含む表スペースを特定の時点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。

表スペースに影響するいずれのログ・レコードも含まれていないことが明確なログ・ファイルをスキップする場合は、**DB2_COLLECT_TS_REC_INFO** レジストリー変数が ON (デフォルト) に設定されていることを確認してください。このレジストリー変数はログ・ファイルを作成および使用する前に設定し、ログ・ファイルのスキップに必要な情報が収集されるようにする必要があります。
DB2_COLLECT_TS_REC_INFO が OFF に設定されている場合、表スペースのロールフォワード時に、その表スペースに影響を与えるログ・レコードがログ・ファイルに含まれていない場合であっても、DB2 はすべてのログ・ファイルを処理します。

注: ログのスキップは DB2 pureScale 環境ではサポートされていません。

データベース・ディレクトリーにある、表スペース変更ヒストリー・ファイル (DB2TSCHG.HIS) は、それぞれの表スペースごとにどのログを処理するべきかを追跡します。 **db2logsForRfwd** ユーティリティーを使用してこのファイルの内容を表示することができ、そこから項目を削除するには、**PRUNE HISTORY** コマンドを使用します。データベース・リストア操作中、DB2TSCHG.HIS はバックアップ・イメージからリストアされ、その後データベース・ロールフォワード操作中に最新になります。ログ・ファイルの情報が使用できない場合には、ログ・ファイルは、すべての表スペースのリカバリーにそれが必要であるかのように扱われます。

それぞれのログ・ファイルの情報は、ログが非アクティブになった後にディスクにフラッシュされるので、破損の結果としてこの情報が消失する可能性があります。これを補うため、リカバリー操作がログ・ファイルの途中から開始した場合には、ログ全体は、それがシステムのすべての表スペースの変更を含んでいるかのように扱われます。その後、アクティブ・ログが処理され、ログ・ファイルのための情報は再ビルドされます。より古いログ・ファイルまたはアーカイブ・ログ・ファイルの情報が破損状態で消失し、それらの情報がデータ・ファイル内にない場合には、

それらのログ・ファイルは、表スペース・リカバリー操作時に、それらがすべての表スペースの変更を含んでいるものとして扱われます。

表スペースをロールフォワードする前に、MON_GET_TABLESPACE 表関数を使用して、表スペースがロールフォワードできる最早ポイントである*最小リカバリー時間* を判別します。最小リカバリー時間は、データ定義言語 (DDL) ステートメントが表スペースまたは表スペース内の表に対し実行されると、更新されます。表スペースは、システム・カタログ表に含まれる情報と同期するように、少なくとも最小リカバリー時間までロールフォワードする必要があります。複数の表スペースをリカバリーする場合、表スペースは少なくとも、リカバリーされるすべての表スペースの最小リカバリー時間のうちで最大の時間までロールフォワードする必要があります。バックアップ・タイム・スタンプより前の時間に表スペースをロールフォワードすることはできません。 パーティション・データベース環境では、表スペースは少なくとも、すべてのデータベース・パーティションにあるすべての表スペースの最小リカバリー時間のうちで最大の時間までロールフォワードする必要があります。

特定の時点にロールフォワードする場合で、表が複数の表スペースに含まれている場合には、すべての表スペースを同時にロールフォワードする必要があります。例えば、表データがある表スペースに含まれていて、その表の索引が別の表スペースに含まれている場合は、両方の表スペースを同じ特定の時点まで同時にロールフォワードする必要があります。

表のデータおよび長形式オブジェクトが別々の表スペースに存在する場合に、その長形式オブジェクト・データが再編成された場合は、データと長形式のオブジェクトのための表スペースは同時にリストアし、ロールフォワードする必要があります。表再編成後に、該当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する場合で、その表スペースの中の表が次のいずれかである場合には、

- 別の表スペースにあるマテリアライズ照会表またはステージング表の基礎表
- 別の表スペースにあるマテリアライズ照会表またはステージング表

両方の表スペースを同じ時点までロールフォワードする必要があります。そのようにしない場合は、SET INTEGRITY ペンディング状態で、マテリアライズ照会表またはステージング表がロールフォワード操作の最後に置かれます。マテリアライズ照会表は完全に更新される必要があり、ステージング表は不完全とマークされます。

特定の時点まで表スペースをロールフォワードしようとするときに、その表スペースに含まれる表が別の表スペースに含まれる別の表との間で参照整合性のリレーションシップにある場合は、同じ特定の時点まで両方の表スペースを同時にロールフォワードする必要があります。そのようにしない場合は、参照整合性のリレーションシップにある子表は、ロールフォワード操作の最後に SET INTEGRITY ペンディング状態になります。子表が後で制約違反のチェックをされた時に、表全体のチェックが必要になります。以下のいずれかの表が存在する場合には、それらも子表とともに SET INTEGRITY ペンディング状態に置かれます。

- 子表の任意の従属マテリアライズ照会表

- 子表の任意の従属ステージング表
- 子表の任意の従属外部キー表

それらの表は、SET INTEGRITY ペンディング状態から戻るために完全な整合性処理が必要です。両方の表スペースを同時にロールフォワードすると、ポイント・イン・タイム指定ロールフォワード操作の終了時に制約がアクティブなままになります。

ポイント・イン・タイム指定の表スペース・ロールフォワード操作を実行する際、トランザクションが、ある表スペースではロールバックされ、別の表スペースではコミットされるという状態にならないよう注意してください。この状態が発生するのは次の場合です。

- ポイント・イン・タイム指定ロールフォワード操作が、トランザクションにより更新された表スペースのサブセットに実行され、その特定の時点がトランザクションのコミットされた時点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時点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 (共有、更新意図、または排他モードで) は、 (ロックを使用して) それらの表スペースに対する実行中のトランザクションがすべて完了するのを待ち、新規の要求をブロックします。静止要求が認可されると、表スペースは整合性のある状態になります。ロールフォワード操作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、それらが最小リカバリー時間よりも後に起こったかを確認することができます。

表スペースのポイント・イン・タイム指定ロールフォワード操作が完了した後、表スペースはバックアップ・ペンディング状態になります。ロールフォワードを実行した時点と現在の時間との間の表スペースに対するすべての更新は除去されてしまうため、その表スペースのバックアップを作成する必要があります。表スペースは、直前のデータベース・レベルや表スペース・レベルのバックアップ・イメージからロールフォワードできなくなります。以下の例では、表スペース・レベルのバックアップ・イメージが必要な理由とその使用方法を示しています。 (表スペースを使用可能にするためには、データベース全体、バックアップ・ペンディング状態の表スペース、またはバックアップ・ペンディング状態の表スペースを含む表スペースのセットのいずれかをバックアップできます。)

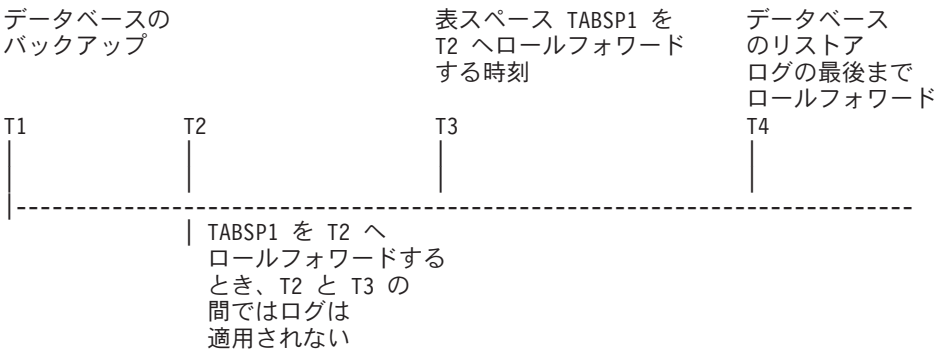


図 29. 表スペースのバックアップ要件

上記の例では、データベースは時刻 T1 にバックアップされます。次に、ポイント・イン・タイム T3 に、表スペース TABSP1 は特定のポイント・イン・タイム (T2) までロールフォワードされ、表スペースはポイント・イン・タイム T3 の後でバックアップされます。表スペースはバックアップ・ペンディング状態であるため、このバックアップ操作は必須です。表スペース・バックアップ・イメージのタイム・スタンプは時刻 T3 より後ですが、表スペースは時刻 T2 にあります。 T2 と T3 の間では、ログ・レコードは TABSP1 には適用されません。時刻 T4 では、データベースが T1 で作成されたバックアップ・イメージを使用してリストアされ、ログの最後までロールフォワードされます。表スペース TABSP1 は時刻 T3 でリストア・ペンディング状態になります。これは、TABSP1 に対し T3 と T4 の間では T2 と T3 の間のログ変更項目を表スペースに適用しないで操作が行われたものとデータベース・マネージャーが想定するからです。これらのログ変更が、実際にはデータベースに対するロールフォワードの一部として適用されていた場合、これは誤った想定となります。表スペース・レベルのバックアップは、表スペースが指定ポイント・イン・タイムまでロールフォワードされた後で作成しなければなりませんが、このバックアップにより、直前のポイント・イン・タイム指定ロールフォワード操作 (例では T3) の後でもその表スペースをロールフォワードできます。

表スペース TABSP1 を T4 までリカバリーする場合は、 T3 の後に作成したバックアップ・イメージ (必須バックアップまたはそれ以後のバックアップ) から表スペースをリストアし、ログの最後まで TABSP1 をロールフォワードします。

上記の例では、時刻 T4 までデータベースをリストアする最も効果的な方法は、必須ステップを以下の順序で実行する方法です。

1. データベースをリストアする。
2. 表スペースをリストアする。
3. データベースをロールフォワードする。

データベースをロールフォワードする前に表スペースをリストアするので、データベースをロールフォワードする際にログ・レコードを表スペースに適用するのにリソースを使用することはありません。

時刻 T3 より後の時間の TABSP1 バックアップ・イメージが見つからないか、TABSP1 を T3 またはそれより前にリストアする場合は、以下の操作を実行できます。

- 表スペースを T3 までロールフォワードする。表スペースはデータベース・バックアップ・イメージからリストアされているため、表スペースを再びリストアする必要はありません。
- 時刻 T1 で作成したデータベース・バックアップを使用して表スペースを再びリストアし、次に表スペースを時刻 T3 より前の時刻までロールフォワードする。
- 表スペースをドロップする。

パーティション・データベース環境では、以下のようになります。

- 表スペースのすべての部分を同時に同じ時点までロールフォワードする必要があります。これにより、表スペースはそれぞれのデータベース・パーティションで整合性が保証されます。
- 一部のデータベース・パーティションがロールフォワード・ペンディング状態で、さらに他のデータベース・パーティションで一部の表スペースがロールフォワード・ペンディング状態である (ただしデータベース・パーティション自体はその状態でない) 場合は、まずデータベース・パーティションをロールフォワードし、次に表スペースをロールフォワードします。
- 表スペースをログの最後までロールフォワードする場合は、データベース・パーティションごとにリストアする必要はありません。リカバリーが必要なデータベース・パーティションでのみリストアが必要です。ただし、特定のポイント・イン・タイムまで表スペースをロールフォワードする場合は、各データベース・パーティションごとにリストアする必要があります。

パーティション表のあるデータベースでは、以下のようになります。

- パーティション表の一部を含む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は、その表を含む他のすべての表スペースも同じ時点までロールフォワードしなければなりません。ただし、ログの最後までロールフォワードする場合は、パーティション表の一部を含む単一の表スペースのみのロールフォワードができます。パーティション表がデータ・パーティションをアタッチしたり、デタッチしたり、ドロップしたりした場合、ポイント・イン・タイム指定ロールフォワードにそれらのデータ・パーティションに関するすべての表スペースも含めなければなりません。パーティション表がデータ・パーティションをアタッチしたり、デタッチしたり、ドロップしたりしたかどうかを判別するには、SYSCAT.DATAPARTITIONS カタログ・ビューを照会してください。

ロールフォワード操作のモニター

db2pd または **LIST UTILITIES** コマンドを使用して、データベースでのロールフォワード操作の進行をモニターできます。

手順

- **LIST UTILITIES** コマンドを発行して、**SHOW DETAIL** パラメーターを指定します。

```
LIST UTILITIES SHOW DETAIL
```

- 次のように **db2pd** コマンドを発行して、**-recovery** パラメーターを指定します。

```
db2pd -recovery
```

タスクの結果

ロールフォワード・リカバリーの場合、進行状況モニターには、FORWARD と BACKWARD の 2 つのフェーズがあります。 FORWARD フェーズでは、ログ・ファイルが読み取られて、そのログ・レコードがデータベースに適用されます。ロールフォワード・リカバリーの場合、このフェーズを開始すると、作業合計の見積もりには UNKNOWN が指定されます。バイト単位での処理済み作業量は、処理の進捗と共に更新されます。

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

例

以下は、**db2pd** コマンドを使用して、ロールフォワード操作のパフォーマンスをモニターするときの出力例です。

```
Recovery:
Recovery Status     0x00000401
Current Log         S0000005.LOG
Current LSN         0000001F07BC
Current LSO         000002551BEA
Job Type            ROLLFORWARD RECOVERY
Job ID              7
Job Start Time      (1107380474) Wed Feb  2 16:41:14 2005
Job Description     Database Rollforward Recovery
Invoker Type        User
Total Phases        2
Current Phase       1

Progress:
Address            PhaseNum Description StartTime               CompletedWork  TotalWork
0x0000000200667160 1        Forward     Wed Feb  2 16:41:14 2005 2268098 bytes  Unknown
0x0000000200667258 2        Backward    NotStarted              0 bytes        Unknown
```

以下は、SHOW DETAIL オプションを指定した **LIST UTILITIES** コマンドを使用して、データベース・ロールフォワード操作のパフォーマンスをモニターするときの出力例です。

```
ID                                = 7
Type                              = ROLLFORWARD RECOVERY
Database Name                     = TESTDB
Member Number                     = 0
Description                       = Database Rollforward Recovery
Start Time                        = 01/11/2012 16:56:53.770404
State                             = Executing
Invocation Type                   = User
Progress Monitoring:
   Estimated Percentage Complete = 50
   Phase Number                   = 1
      Description                 = Forward
      Total Work                  = 928236 bytes
      Completed Work              = 928236 bytes
      Start Time                  = 01/11/2012 16:56:53.770492

   Phase Number [Current]         = 2
      Description                 = Backward
      Total Work                  = 928236 bytes
      Completed Work              = 0 bytes
      Start Time                  = 01/11/2012 16:56:56.886036
```

以下は、SHOW DETAIL オプションを指定した **LIST UTILITIES** コマンドを使用して、表スペースのロールフォワード操作のパフォーマンスをモニターするときの出力例です。

```
ID                                 = 17
Type                               = ROLLFORWARD RECOVERY
Database Name                      = TESTDB
Member Number                      = 0
Description                        = Offline Tablespace Rollforward Recovery: 3
Start Time                         = 01/11/2012 17:04:27.269171
State                              = Executing
Invocation Type                    = User
Progress Monitoring:
   Estimated Percentage Complete = 63
   Phase Number                    = 1
      Description                  = Forward
      Total Work                   = 142
      Completed Work               = 90
      Start Time                   = 01/11/2012 17:04:27.269283

   Phase Number [Current]          = 2
      Description                  = Backward
      Total Work                   = 0
      Completed Work               = 0
      Start Time                   = Not Started
```

ロールフォワードに必要な許可

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・マネージャーの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・セッション - CLP の例

例 1

ROLLFORWARD DATABASE コマンドでは、それぞれをキーワード AND で区切ることによって、一度に複数の操作を指定することができます。例えば、ログの終わりまでロールフォワードし、完了する場合、コマンドを別々に指定すると、次のようになります。

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete
```

これらは次のように結合することができます。

```
db2 rollforward db sample to end of logs and complete
```

上記の 2 つは同じですが、このような操作は 2 つのステップで実行することをお勧めします。ロールフォワード操作が期待どおりに進行したことを確認してから、ロールフォワード操作を停止することが重要です。ログを失わないためにもこれを行うのは大切なことです。

ロールフォワード・コマンドでエラーが発生すると、ロールフォワード操作は完了しません。エラーが戻され、そのエラーを修正した後、コマンドを再発行することができます。しかし、エラーを修正できない場合は、次のコマンドを発行することによりロールフォワードを強制的に完了することができます。

```
db2 rollforward db sample complete
```

このコマンドは、失敗前のログの時点でデータベースをオンラインにします。

例 2

ログの終わりまでデータベースをロールフォワードします (2 つの表スペースがリストアされています)。

```
db2 rollforward db sample to end of logs
db2 rollforward db sample to end of logs and stop
```

これら 2 つのステートメントは同等です。ログの終わりまで表スペースのロールフォワード操作を実行する場合、 AND STOP または AND COMPLETE を使用する必要はありません。表スペース名は必須ではありません。指定しない場合には、ロールフォワード・リカバリーを必要としているすべての表スペースが組み込まれます。これらの表スペースの一部のみをロールフォワードする場合は、それらの名前を指定しなければなりません。

例 3

3 つの表スペースがリストアされた後、1 つをログの終わりまでロールフォワードし、他の 2 つをある時点までロールフォワードします (両方ともオンラインで行われます)。

```
db2 rollforward db sample to end of logs tablespace(TBS1) online
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
   tablespace(TBS2, TBS3) online
```

2 つのロールフォワード操作が並行して実行されるわけではないことに注意してください。 2 番目のコマンドは、最初のロールフォワード操作が正常に完了した場合にのみ起動することができます。

例 4

データベースをリストアした後、 OVERFLOW LOG PATH でユーザー出口がアーカイブ・ログを保管するディレクトリーを指定して、ある時点までロールフォワードします。

```
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
   overflow log path (/logs)
```

例 5

以下の例では、sample という名前のデータベースがあります。データベースがバックアップされ、バックアップ・イメージにリカバリー・ログが含められます。次に

データベースがリストアされます。最後にデータベースがバックアップ・タイム・スタンプの最後までロールフォワードされます。

リカバリー・ログをバックアップ・イメージに入れ、データベースをバックアップします。

```
db2 backup db sample online include logs
```

そのバックアップ・イメージを使用してデータベースをリストアします。

```
db2 restore db sample
```

バックアップ・タイム・スタンプの最後ま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 backup
```

例 6 (パーティション・データベース環境)

データベース・パーティション 0、1、および 2 があります。表スペース TBS1 はすべてのデータベース・パーティションで定義されており、表スペース TBS2 はデータベース・パーティション 0 および 2 で定義されています。データベース・パーティション 1 でデータベースをリストアし、データベース・パーティション 0 および 2 で TBS1 をリストアした後、データベース・パーティション 1 でデータベースをロールフォワードします。

```
db2 rollforward db sample to end of logs and stop
```

これにより、警告 SQL1271 (「データベースは回復されましたが、1 つ以上の表スペースがデータベース・パーティション 0 および 2 でオフラインです。」) が戻されます。

```
db2 rollforward db sample to end of logs
```

これにより、データベース・パーティション 0 および 2 で TBS1 がロールフォワードされます。この場合、TABLESPACE(TBS1) 節はオプションです。

例 7 (パーティション・データベース環境)

以下の例では、sample という名前のパーティション・データベースがあります。すべてのデータベース・パーティションがシングル・システム・ビュー・バックアップでバックアップされます。次にデータベースがすべてのデータベース・パーティションでリストアされます。最後にデータベースがバックアップ・タイム・スタンプの最後までロールフォワードされます。

シングル・システム・ビュー (SSV) バックアップを実行します。

```
db2 backup db sample on all nodes online include logs
```

すべてのデータベース・パーティションでデータベースをリストアします。

```
db2_all "db2 restore db sample taken at 1998-04-03-14.21.56"
```

バックアップ・タイム・スタンプの最後ま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 backup on all nodes
```

例 8 (パーティション・データベース環境)

以下の例では、sample という名前のパーティション・データベースがあります。db2_all を使用する 1 つのコマンドですべてのデータベース・パーティションがバックアップされます。次にデータベースがすべてのデータベース・パーティションでリストアされます。最後にデータベースがバックアップ・タイム・スタンプの最後までロールフォワードされます。

db2_all を使用する 1 つのコマンドですべてのデータベース・パーティションをバックアップします。

```
db2_all "db2 backup db sample include logs to //dir/"
```

すべてのデータベース・パーティションでデータベースをリストアします。

```
db2_all "db2 restore db sample from //dir/"
```

バックアップ・タイム・スタンプの最後ま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 backup on all nodes
```

例 9 (パーティション・データベース環境)

データベース・パーティション 0 および 2 でのみ表スペース TBS1 をリストアした後、データベース・パーティション 0 および 2 で TBS1 をロールフォワードします。

```
db2 rollforward db sample to end of logs
```

データベース・パーティション 1 は無視されます。

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

データベース・パーティション 1 で TBS1 がロールフォワード・リカバリー可能な状態になっていないため、これは失敗します。SQL4906N が報告されます。

```
db2 rollforward db sample to end of logs on
   dbpartitionnums (0, 2) tablespace(TBS1)
```

これは正常に完了します。

```
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
   tablespace(TBS1)
```

データベース・パーティション 1 で TBS1 がロールフォワード・リカバリー可能な状態になっていないため、これは失敗します。すべての部分は一緒にロールフォワードされなければなりません。

注: 表スペースを特定のポイント・イン・タイムまでロールフォワードする場合、dbpartitionnum 節は受け入れられません。ロールフォワード操作は、表スペースが存在するすべてのデータベース・パーティションで行う必要があります。

データベース・パーティション 1 で TBS1 をリストアした後

```
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
   tablespace(TBS1)
```

これは正常に完了します。

例 10 (パーティション・データベース環境)

すべてのデータベース・パーティションで表スペースをリストアした後、 PIT2 までロールフォワードしますが、AND STOP は指定しません。ロールフォワード操作はまだ進行中です。それを取り消し、PIT1 までロールフォワードします。

```
db2 rollforward db sample to pit2 tablespace(TBS1)
db2 rollforward db sample cancel tablespace(TBS1)

** restore TBS1 on all dbpartitionnums **

db2 rollforward db sample to pit1 tablespace(TBS1)
db2 rollforward db sample stop tablespace(TBS1)
```

例 11 (パーティション・データベース環境)

db2nodes.cfg ファイルにリスト表示されている 8 個のデータベース・パーティション (3 から 10 まで) に存在する表スペースをロールフォワード・リカバリーします。

```
db2 rollforward database dwtest to end of logs tablespace (tssprodt)
```

ログの終わり (ある時点ではなく) までのこの操作は正常に完了します。表スペースが存在するデータベース・パーティションは指定する必要がありません。ユーティリティーは、デフォルトとして db2nodes.cfg ファイルを使用します。

例 12 (パーティション・データベース環境)

(データベース・パーティション 6 上の) 単一データベース・パーティションのデータベース・パーティション・グループに存在する 6 個の小さな表スペースをロールフォワード・リカバリーします。

```
db2 rollforward database dwtest to end of logs on dbpartitionnum (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

ログの終わり (ある時点ではなく) までのこの操作は正常に完了します。

例 13 (パーティション表 - すべてのデータ・パーティションについて、ログの最後にロールフォワードする)

表スペース tbsp1、tbsp2、tbsp3 を使用し、索引を tbsp0 内に持つパーティション表が作成されています。その後、ユーザーは tbsp4 の表にデータ・パーティションを追加し、tbsp5 の表からのデータ・パーティションをアタッチします。この環境ではすべての表スペースを END OF LOGS にロールフォワードすることができます。

```
db2 rollforward db PBARDB to END OF LOGS and stop
   tablespace(tbsp0, tbsp1, tbsp2, tbsp3, tbsp4, tbsp5)
```

これは正常に完了します。

例 14 (パーティション表 - 1 つの表スペースについて、ログの最後にロールフォワードする)

表スペース tbsp1、tbsp2、tbsp3 を使用し、索引を tbsp0 内に持つパーティション表が最初に作成されています。その後、ユーザーは tbsp4 の表にデータ・パーティシ

ョンを追加し、tbsp5 の表からのデータ・パーティションをアタッチします。表スペース tbsp4 は破損し、リストアおよびログの最後へのロールフォワードを必要とします。

```
db2 rollforward db PBARDB to END OF LOGS and stop tablespace(tbsp4)
```

これは正常に完了します。

例 15 (パーティション表 - 追加、アタッチ、デタッチされたデータ・パーティション、または索引を持つデータ・パーティションを含む、すべてのデータ・パーティションの PIT にロールフォワードする)

表スペース tbsp1、tbsp2、tbsp3 を使用し、索引を tbsp0 内に持つパーティション表が作成されています。その後、ユーザーは tbsp4 の表にデータ・パーティションを追加し、tbsp5 の表からのデータ・パーティションをアタッチし、tbsp1 からデータ・パーティションをデタッチします。ユーザーは、INDEX IN 節で指定される表スペースを含め、パーティション表によって使用されるすべての表スペースとともに、PIT へのロールフォワードを実行します。

```
db2 rollforward db PBARDB to 2005-08-05-05.58.53 and stop
   tablespace(tbsp0, tbsp1, tbsp2, tbsp3, tbsp4, tbsp5)
```

これは正常に完了します。

例 16 (パーティション表 - 表スペースのサブセット上の PIT にロールフォワードする)

3 つの表スペース (tbsp1、tbsp2、tbsp3) を使ってパーティション表が作成されています。その後、ユーザーは tbsp3 からすべてのデータ・パーティションをデタッチします。 PIT へのロールフォワードは tbsp1 および tbsp2 でのみ許可されます。

```
db2 rollforward db PBARDB to 2005-08-05-06.02.42 and stop
   tablespace( tbsp1, tbsp2)
```

これは正常に完了します。

第 15 章 IBM Tivoli Storage Manager (TSM) でのデータ・リカバリー

BACKUP DATABASE コマンドまたは **RESTORE DATABASE** コマンドを呼び出す時に、IBM Tivoli Storage Manager (TSM) 製品を使用してデータベースまたは表スペースのバックアップの管理またはリストア操作の管理を行うことを指定できます。

以下のシステムを除き、TSM クライアント API の最低限必要なレベルは、バージョン 4.2.0 です。

- 64 ビット Solaris システムでは、TSM クライアント API バージョン 4.2.1 が必要です。
- 64 ビット Windows オペレーティング・システムでは、TSM クライアント API バージョン 5.1 が必要です。
- すべての Windows x64 システムで、TSM クライアント API バージョン 5.3.2 が必要です。
- 32 ビット Linux for IBM Power Systems™ and pSeries では、 TSM クライアント API バージョン 5.1.5 以上が必要です。
- 64 ビット Linux for IBM Power Systems and pSeries では、 TSM クライアント API バージョン 5.2.2 以上が必要です。
- 64 ビット Linux on AMD Opteron システムでは、TSM クライアント API バージョン 5.2.0 以上が必要です。
- Linux for zSeries では、TSM クライアント API バージョン 5.2.2 以上が必要です。

Tivoli Storage Manager クライアントの構成

DB2 データベース・マネージャーが、 IBM Tivoli Storage Manager (TSM) クライアントを使用して、データベースまたは表スペースのバックアップやリストア操作を管理できるようにするには、 TSM 環境を構成する必要があります。

始める前に

機能している TSM クライアントおよびサーバーをインストールし、構成する必要があります。加えて、TSM クライアント API を各 DB2 データベース・サーバーにインストールする必要があります。 TSM クライアント・プロキシー・ノードがサポートされるのは、TSM サーバーがこれらをサポートするよう構成されている場合です。サーバー構成とプロキシー・ノードのサポートに関する情報は、462 ページの『Tivoli Storage Manager を使用する際の考慮事項』を参照するか、Tivoli 資料をご覧ください。

手順

DB2 データベース・システムが使用できるように TSM 環境を構成するには、次のようにします。

1. TSM クライアント API で使う環境変数を設定します。

DSMI_DIR
API トラステッド・エージェント・ファイル (dsmtca) が存在しているユーザー定義のディレクトリー・パスを識別します。

DSMI_CONFIG
dsm.opt ファイルへのユーザー定義ディレクトリー・パスを識別します。これには、TSM ユーザー・オプションが含まれています。他の 2 つの変数とは異なり、この変数には完全修飾パスおよびファイル名を含めなければなりません。

DSMI_LOG
エラー・ログ (dsierror.log) が作成されるユーザー定義のディレクトリー・パスを識別します。

注: 複数パーティション・データベース環境では、これらの設定を sqllib/userprofile ファイルの中で指定する必要があります。

2. それらの環境変数に変更が加えられ、データベース・マネージャーが稼働している場合は、データベース・マネージャーを停止して再始動してください。 例えば、以下のようにします。
   - **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。
3. サーバーの構成によっては、Tivoli クライアントは TSM サーバーとインターフェースを持つためにパスワードが必要になる場合があります。

   TSM 環境が PASSWORDACCESS=generate を使用するように構成されている場合には、 Tivoli クライアントは、パスワードを設定する必要があります。

   実行可能ファイル dsmapipw は、インスタンス所有者の sqllib/adsm ディレクトリーにインストールされています。この実行可能ファイルにより、TSM パスワードの確立と再設定が可能になります。

   dsmapipw コマンドを実行するには、ローカル管理者または「root」ユーザー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実行時には、このパスワードは、ご使用のノードが TSM サーバーに登録された時に TSM 管理者から提供されたものになります。
   - *新規パスワード。* TSM サーバーに保管させる、TSM ノードの新しいパスワード。 (新規パスワードは、入力エラーがないかどうかを調べるため、2 回入力するよう求められます。)

   注: BACKUP DATABASE または **RESTORE DATABASE** コマンドを呼び出すユーザーは、このパスワードを知っている必要はありません。 **dsmapipw** コマンドを実行する必要があるのは、初期接続時のパスワードを確立する場合、および TSM サーバーのパスワードが初期化された場合のみです。

次のタスク

バックアップとログ・アーカイブの方針によっては、プロキシー・ノードを使用するために、TSM クライアントを構成する追加の手順を実行する必要が生じることがあります。プロキシー・ノードを使用すると、複数のクライアント・ノード上 (または複数ユーザーの下) で存在するデータベースのバックアップおよびログ・アーカイブを、TSM サーバー上の共通のターゲット・ノード名に統合することができます。この構成は、例えばクラスターを使用する場合など、バックアップを実行する管理者またはコンピューターが時間の経過とともに変わる可能性のある場合に役立ちます。さらに asnodename オプションを使用すると、バックアップを実行したユーザーとは異なるユーザーが、または異なるコンピューターから、データをリストアできます。

DB2 pureScale環境で TSM を使用する場合は、プロキシー・ノード構成をお勧めします。この構成では、各メンバーを TSM クライアントまたはノードとして表し、共通プロキシー・ノードにマップすることが可能だからです。

デフォルトでプロキシー・ノードを使用しない予定の場合は、追加的なクライアント・セットアップは必要ありません。プロキシー・ノードを使ってバックアップ操作またはリストア操作を実行する場合には、**BACKUP DATABASE** または **RESTORE DATABASE** コマンドの呼び出し時に **OPTIONS** パラメーターで asnodename 値を指定します。

デフォルトで TSM プロキシー・ノードを使用するには、次の方法に従ってください。

- 異なるデータベースに対して異なるプロキシー・ノードを使用するよう、データベース構成パラメーターを更新します。
- 1 つのマシン上のすべてのユーザーとデータベースに対して同じプロキシー・ノードを使用するよう、dsm.sys ファイルを更新します。

  注: 同一の TSM プロキシー名を使用するユーザーとホストの組み合わせは、すべて同一の DB2 インスタンスとして TSM から認識されます。これは、TSM クライアント・ノード・プロキシー構成内で、複数の DB2 インスタンスが同一のデータベース名を使用する場合、それらのインスタンスが互いのログ・アーカイブおよびバックアップ・イメージを上書きする可能性があることを意味します。これを回避するには、以下のようにします。
  - それぞれの DB2 インスタンスに異なるプロキシー・ホスト名を作成する。
  - 複数の DB2 インスタンスが同じ TSM プロキシー名を使用するデータベースを作成する可能性がある場合は、TSM のクライアント・ノード・プロキシー機能を使用しない。

vendoropt、logarchopt1、および logarchopt2 を使用する TSM クライアント・セットアップ

以下に示す 1 つ以上のデータベース構成パラメーターを設定すると、各データベースに対して複数のプロキシー・ノード設定が可能になります。

- (バックアップ、リストアなどの) TSM を使用するコマンドでプロキシー・ノードを使用可能にするには、次のようにして、**vendoropt** データベース構成パラメーターで asnodename オプションを指定します。

  ```
  db2 update db cfg for dbname using vendoropt "'-asnodename=proxynode'"
  ```

ここで、*proxynode* は共有される TSM プロキシー・ノードの名前です。

- TSM サーバーへのログ・アーカイブを構成するには、次のように、**logarchmeth1** データベース構成パラメーターを TSM に設定して、**logarchopt1** データベース構成パラメーターでプロキシー・ノードの名前を asnodename 値として指定します。

  ```
  db2 update db cfg for dbname using logarchmeth1 tsm
  logarchopt1 "'-asnodename=proxynode'"
  ```

  ここで、*proxynode* は共有される TSM プロキシー・ノードの名前です。

  同様にして、**logarchmeth2** および **logarchopt2** データベース構成パラメーターを更新することができます。

DB2 pureScale環境では、これらのデータベース構成パラメーターはグローバル・パラメーターであり、任意のメンバーから設定できます。

dsm.sys ファイルを使用する TSM クライアント・セットアップ方法

1. 次のように dsm.sys ファイルを編集してプロキシー・ノード情報を追加します。

   ```
   asnodename proxynode
   ```

   ここで、*proxynode* は共有される TSM プロキシー・ノードの名前です。

2. 次のようにして、**DSMI_CONFIG** パスで指定される dsm.opt ファイルに TSM サーバーの名前が含まれるようにします。

   ```
   servername servername
   ```

   ここで *servername* は TSM サーバーの名前です。

Tivoli Storage Manager を使用する際の考慮事項

- Tivoli Storage Manager (TSM) 内部の特定のフィーチャーを使用するためには、そのフィーチャー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生じる場合があります。 (Windows オペレーティング・システムでは、/ の代わりに ¥ が使用されることに注意してください。) 完全修飾パス名は、次のようになります。
  - フル・データベース・リカバリー・オブジェクト */database*/DBPART*nnn*/FULL_BACKUP.timestamp.seq_no
  - 増分データベース・リカバリー・オブジェクト */database*/DBPART*nnn*/DB_INCR_BACKUP.timestamp.seq_no
  - 増分差分データベース・リカバリー・オブジェクト */database*/DBPART*nnn*/DB_DELTA_BACKUP.timestamp.seq_no
  - フル表スペース・リカバリー・オブジェクト */database*/DBPART*nnn*/TSP_BACKUP.timestamp.seq_no
  - 増分表スペース・リカバリー・オブジェクト */database*/DBPART*nnn*/TSP_INCR_BACKUP.timestamp.seq_no
  - 増分差分表スペース・リカバリー・オブジェクト */database*/DBPART*nnn*/TSP_DELTA_BACKUP.timestamp.seq_no

ここで、*database* はデータベースの別名で、 DBPART*nnn* はデータベース・パーティション番号です。 大文字で表記されている名前は大文字で入力しなければなりません。

- 複数のバックアップ・イメージで同じデータベース別名が使用されている場合は、完全修飾名は、タイム・スタンプとシーケンス番号で区別されます。使用するバックアップ・バージョンを判別するには、TSM を照会する必要があります。
- USE TSM オプションと INCLUDE LOGS オプションを指定してオンライン・バックアップ操作を行う場合、2 つのプロセスが同時に同じ磁気テープ・ドライブに書き込もうとすると、デッドロックが発生することがあります。磁気テープ・ドライブをログとバックアップ・イメージのストレージ・デバイスとして使用する場合は、TSM に 2 つの別々のテープ・プールを定義する必要があります (1 つはバックアップ・イメージ用、もう 1 つはアーカイブ・ログ用)。
- クライアント・プロキシー・ノードを使用するためには、TSM 管理者は TSM サーバーで以下の手順を完了する必要があります。
  1. DB2 クライアントがまだ TSM サーバーに登録されていない場合、TSM コマンド **register node** を使って各クライアントを登録します。
  2. TSM コマンド **register node** を使用して、クライアント・グループによって使われる (仮想) 共通 TSM ノード名を TSM サーバーに登録します。
  3. TSM コマンド **grant proxynode** を使用して、グループ内のすべてのコンピューターにプロキシー権限を付与します。

  プロキシー・ノード・クライアントのセットアップ方法に関する情報は、459 ページの『Tivoli Storage Manager クライアントの構成』を参照するか、Tivoli 資料をご覧ください。
- わずかなページ数のみが変更された増分バックアップを使用する場合、最大の表スペースのバックアップを完了するのに必要な時間よりも、TSM パラメーター **IDLETIMEOUT** を大きくしなければならないことがあります。こうすることで、増分バックアップの完了より前に TSM がセッションを閉じることがなくなります。

第 16 章 DB2 拡張コピー・サービス (ACS)

DB2 拡張コピー・サービス (ACS) によって、ストレージ・デバイスの高速コピー・テクノロジーを使用して、バックアップ操作およびリストア操作のデータ・コピー部分を実行できます。

従来のバックアップ操作またはリストア操作では、データベース・マネージャーはオペレーティング・システム呼び出しを使用して、ディスクまたはストレージ・デバイスとの間でデータをコピーします。ストレージ・デバイスを使用したデータ・コピーが可能になると、バックアップ操作およびリストア操作がかなり高速になります。 DB2 ACS を使用するバックアップ操作を、スナップショット・バックアップと呼びます。

スナップショット・バックアップ操作およびリストア操作を実行するには、ご使用のストレージ・デバイス用の DB2 ACS API ドライバーが必要です。 IBM Data Server には、以下のストレージ・ハードウェアのための DB2 ACS API ドライバーが組み込まれています。

- IBM TotalStorage SAN ボリューム・コントローラー
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N シリーズ
- NetApp V-series
- NetApp FAS series

DB2 拡張コピー・サービス (ACS) のセットアップと使用について詳しくは、拡張コピー・サービスに関する Tivoli の資料 (http://www.ibm.com/developerworks/wikis/display/tivolidoccentral/Tivoli+Storage+Manager+for+Advanced+Copy+Services) を参照してください。

DB2 拡張コピー・サービス (ACS) のベスト・プラクティス

DB2 拡張コピー・サービス (ACS) をインストールおよび構成するときは、以下のベスト・プラクティスを考慮してください。

ログ・パスには専用のボリューム・グループを指定する
ログ・パスは、データベース・ディレクトリーおよびデータベース・コンテナーから独立した独自のスナップショット・ボリューム内に含めることをお勧めします。

データベース・パーティションごとに 1 つのボリューム・グループを指定する
パーティション・データベース環境では、各データベース・パーティションは、他のデータベース・パーティションから独立したスナップショット・ボリューム・セットに存在しなければなりません。

DB2 拡張コピー・サービス (ACS) の制約事項

DB2 拡張コピー・サービス (ACS) をインストールおよび構成するときは、考慮すべきいくつかの制約事項があります。

ボリューム共有はサポートされていません。データベース・パーティションが他のいずれかのデータベース・パーティションと同じストレージ・ボリュームにある場合、スナップショット操作は許可されません。表 16 に、 IBM Tivoli Storage Manager (TSM) のフル・ライセンスを取得することで解除されるいくつかの制約事項をリストします。

表 16. DB2 付属の DB2 組み込み ACS でサポートされる機能と、IBM Tivoli Storage Manager (TSM) 製品のフル・バージョンでサポートされる機能の比較

機能項目	組み込み DB2 サポート	TSM for ACS サポート
ローカルのスナップショット・バックアップのバージョン	最大 2 つのバージョンのスナップショットがサポートされます。	製品の制限はありません。ACS は、ストレージ・デバイスおよび使用可能なリソースのある限り、多数のバージョンをサポートします。
ミラーリングと統合されたスナップショットのサポート	サポートされていません。	AIX 論理ボリューム・マネージャー (LVM) のミラーリングによるソースまたはターゲットのいずれかのミラー・セットから取得したスナップショットをサポートします。
スナップショット・イメージのテープへのバックアップの統合化	統合化はサポートされていません。従来のバックアップおよびスナップショット・バックアップにより補完できますが、統合化されていません。	スナップショット・イメージの TSM へのバックアップは、完全に統合化されてサポートされています。バックアップ・コマンド 1 つで、スナップショット・バックアップの取得と TSM へのバックアップを実行します。
実動サーバーからオフロードしたテープへのバックアップ	テープへのバックアップの統合化はサポートされていません。	2 次ホストから TSM へのバックアップの実行は、完全に統合化されてサポートされています。
ACS リポジトリーのバックアップの統合化	サポートされていません。リポジトリーが非アクティブまたはシャットダウンされている時に、外部バックアップを実行できます。	TSM へ自動バックアップされます。

DB2 拡張コピー・サービス (ACS) の使用可能化

DB2 拡張コピー・サービス (ACS) を使用するか、スナップショットのバックアップ操作を実行するには、DB2 ACS をインストール、活動化、および構成する必要があります。

始める前に

DB2 ACS は、IBM DB2 高可用性 (HA)フィーチャーの一部です。DB2 ACS を使用するには、DB2 HA フィーチャーのライセンスが必要です。

スナップショット・バックアップ操作およびリストア操作を実行するには、ご使用のストレージ・デバイス用の DB2 ACS API ドライバーが必要です。 IBM Data Server には、以下のストレージ・ハードウェアのための DB2 ACS API ドライバーが組み込まれています。

- IBM TotalStorage SAN ボリューム・コントローラー
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N シリーズ
- NetApp V-series
- NetApp FAS series

手順

1. DB2 ACS をインストールします。『DB2 拡張コピー・サービス (ACS) のインストール』を参照してください。
2. DB2 ACS を使用するためのデータベース・マネージャー・インスタンス (複数も可) を作成します。

   新規データベース・マネージャー・インスタンスを作成するとき、`acs` と呼ばれるディレクトリーが新規インスタンスの `sqllib` ディレクトリーに作成されます。各データベース・マネージャー・インスタンスに `acs` ディレクトリーがあるので、各データベース・マネージャー・インスタンスをそれぞれに構成することができます。
3. DB2 ACS を使用するためのデータベース・マネージャー・インスタンスごとに、以下のステップを実行します。
   a. DB2 ACS を活動化します。 469 ページの『DB2 拡張コピー・サービス (ACS) の活動化』を参照してください。
   b. DB2 ACS を構成します。 470 ページの『DB2 拡張コピー・サービス (ACS) の構成』を参照してください。

タスクの結果

DB2 ACS を使用可能にした後、スナップショットのバックアップ操作を実行することができます。

DB2 拡張コピー・サービス (ACS) のセットアップと使用について詳しくは、拡張コピー・サービスに関する Tivoli の資料 (http://www.ibm.com/developerworks/wikis/display/tivolidoccentral/Tivoli+Storage+Manager+for+Advanced+Copy+Services) を参照してください。

DB2 拡張コピー・サービス (ACS) のインストール

DB2 拡張コピー・サービス (ACS) に必要なファイルおよびライブラリーは、標準インストールおよびカスタム・インストールでのみインストールされます。

始める前に

ACS をインストールする前に、以下のライブラリーをインストールしておく必要があります。

AIX の場合:

- `ln -s /opt/freeware/lib/powerpc-ibm-aix5.3.0//libgcc_s.a /usr/lib/libgcc_s.a`

Red Hat Enterprise Linux の場合:

- `ln -s libssl.so.0.9.7xxx libssl.so.0.9.7`
- `ln -s libcrypto.so.0.9.7xxx libcrypto.so.0.9.7`
- `ln -s libssl.so.0.9.7xxx libssl.so`
- `ln -s libssl.so.0.9.7xxx libssl.so.0`

制約事項

DB2 ACS は、IBM Data Server がサポートするハードウェアおよびオペレーティング・システムのサブセットをサポートします。DB2 ACS がサポートするハードウェアおよびオペレーティング・システムのリストについては、521 ページの『DB2 拡張コピー・サービス (ACS) をサポートするオペレーティング・システムおよびハードウェア』を参照してください。

手順

1. DB2 ACS インストールおよび構成のベスト・プラクティスおよび DB2 ACS の制約事項を参照します。465 ページの『DB2 拡張コピー・サービス (ACS) のベスト・プラクティス』および 466 ページの『DB2 拡張コピー・サービス (ACS) の制約事項』を参照してください。
2. **db2_install** コマンド、標準インストール、カスタム・インストール、または **ACS** 応答ファイル・キーワードを使用したカスタム応答ファイル・インストールを使用して、DB2 Database for Linux, UNIX, and Windows をインストールします。

   重要: コマンド**db2_install** は推奨されておらず、今後のリリースで除去される可能性があります。代わりに、**db2setup** コマンドと応答ファイルを使用してください。

   注: 過去のリリースにあった、コンパクト・インストール時の DB2 ACS の自動インストールは廃止されました。コンパクト・インストールを完了した後に、DB2 ACS をインストールする必要が生じた場合は、ACS キーワードを使用したカスタム応答ファイル・インストールを使用してください。
3. TCP/IP サービス・ファイルに DB2 ACS エージェント用のポートを追加します。 以下に例を示します。

   ```
   db2acs  5400/tcp # DB2 ACS service port
   ```

次のタスク

DB2 ACS をインストールした後、スナップショットのバックアップ操作を実行する前に、 DB2 ACS を活動化し、DB2 ACS を構成する必要があります。469 ページの

ジの『DB2 拡張コピー・サービス (ACS) の活動化』および 470 ページの『DB2 拡張コピー・サービス (ACS) の構成』を参照してください。

DB2 拡張コピー・サービス (ACS) のセットアップと使用について詳しくは、拡張コピー・サービスに関する Tivoli の資料 (http://www.ibm.com/developerworks/wikis/display/tivolidoccentral/Tivoli+Storage+Manager+for+Advanced+Copy+Services) を参照してください。

DB2 拡張コピー・サービス (ACS) の活動化

DB2 拡張コピー・サービス (ACS) を使用して、指定されたデータベース・マネージャー・インスタンスのスナップショットのバックアップを実行する前に、そのインスタンスで DB2 ACS 機能を活動化する必要があります。スクリプトを実行することによって、DB2 ACS を活動化します。

始める前に

DB2 ACS を活動化する前に、以下のタスクを実行する必要があります。

1. DB2 ACS をインストールします。詳しくは、 467 ページの『DB2 拡張コピー・サービス (ACS) のインストール』を参照してください。
2. DB2 ACS を使用するためのデータベース・マネージャー・インスタンス (複数も可) を作成します。

このタスクについて

データベース・マネージャー・インスタンスの作成中の、IBM Data Server のアップグレード時に、データベース・マネージャーは自動的に **setup.sh** を呼び出して、DB2 ACS 機能を活動化します。

手動で DB2 ACS を活動化することもできます。

手順

DB2 ACS を手動で活動化するには、DB2 ACS を活動化するための適切なパラメーターを使用した **setup.sh** スクリプトを、root 権限を持つユーザーとして実行します。
setup.sh について詳しくは、 471 ページの『DB2 拡張コピー・サービス (ACS) セットアップ・スクリプト setup.sh』を参照してください。

タスクの結果

setup.sh スクリプトの実行の重要な結果の 1 つは、`sqllib/acs` ディレクトリーの DB2 ACS 実行可能ファイルの所有権および許可が検証されることです。

次のタスク

DB2 ACS を活動化した後、スナップショットのバックアップ操作を実行する前に、DB2 ACS を構成する必要があります。

DB2 拡張コピー・サービス (ACS) のセットアップと使用について詳しくは、拡張コピー・サービスに関する Tivoli の資料 (http://www.ibm.com/developerworks/wikis/

display/tivolidoccentral/Tivoli+Storage+Manager+for+Advanced+Copy+Services) を参照してください。

DB2 拡張コピー・サービス (ACS) の構成

DB2 拡張コピー・サービス (ACS) を使用してスナップショットのバックアップを実行する前に、DB2 ACS を構成する必要があります。構成ファイルを使用して DB2 ACS を構成します。

始める前に

DB2 ACS を構成する前に、以下のタスクを実行する必要があります。

1. DB2 ACS をインストールします。詳しくは、467 ページの『DB2 拡張コピー・サービス (ACS) のインストール』を参照してください。
2. DB2 ACS を使用するためのデータベース・マネージャー・インスタンス (複数も可) を作成します。
3. DB2 ACS を活動化します。詳しくは、469 ページの『DB2 拡張コピー・サービス (ACS) の活動化』を参照してください。

手順

パラメーターを指定せずに、**sqllib/acs** ディレクトリーから setup.sh スクリプトを実行します。 これは DB2 ACS を構成する対話式のテキスト・ベースのウィザードによって行われます。このウィザードは構成プロファイル・ファイルを作成し、マシンの /etc/initab を変更して、DB2 ACS デーモンを起動します。**setup.sh** ウィザードの出力例を以下に示します。

```
./setup.sh
Do you have a full TSM license to enable all features of TSM for ACS ?[y/n]

n

****** Profile parameters for section GLOBAL: ******
ACS_DIR [/home/krodger/sqllib/acs ]
ACSD [localhost 57328 ]
TRACE [NO ]

****** Profile parameters for section ACSD: ******
ACS_REPOSITORY *mandatory parameter* /home/krodger/acsrepository

****** Profile parameters for section CLIENT: ******
MAX_VERSIONS [ADAPTIVE ] 2
LVM_FREEZE_THAW [YES ]
DEVICE_CLASS [STANDARD ]

****** Profile parameters for section STANDARD: ******
COPYSERVICES_HARDWARE_TYPE *mandatory parameter*
NAS_NSERIES COPYSERVICES_PRIMARY_SERVERNAME *mandatory parameter* fas960a
COPYSERVICES_USERNAME [superuser ] root

=========================================================================

The profile has beeen successfully created.
Do you want to continue by specifying passwords for the defined devices? [y/n]

y

Please specify the passwords for the following profile sections:
STANDARD
```

```
master

Creating password file at /home/krodger/sqllib/acs/shared/pwd.acsd.
A copy of this file needs to be available to all components that connect to acsd.

BKI1555I: Profile successfully created. Performing additional checks.
Make sure to restart all ACS components to reload the profile.
```

タスクの結果

DB2 ACS を構成した後、スナップショットのバックアップ操作を実行できます。

DB2 拡張コピー・サービス (ACS) のセットアップと使用について詳しくは、拡張コピー・サービスに関する Tivoli の資料 (http://www.ibm.com/developerworks/wikis/display/tivolidoccentral/Tivoli+Storage+Manager+for+Advanced+Copy+Services) を参照してください。

DB2 拡張コピー・サービス (ACS) のディレクトリーの構成

新規データベース・マネージャー・インスタンスを作成するとき、acs と呼ばれるディレクトリーが新規インスタンスの sqllib ディレクトリーに作成されます。DB2 拡張コピー・サービス (ACS) はこの acs ディレクトリーを使用して、ターゲット・ボリューム制御ファイルおよびリカバリー・オブジェクトの共有リポジトリーのような構成ファイルを保管します。この acs ディレクトリーを変更または構成する方法に関しては、制限事項があります。

このタスクについて

1. acs ディレクトリーは、DB2 ACS またはスナップショットのバックアップ操作に組み込むことはできません。
2. acs ディレクトリーは、IBM Tivoli Storage Manager (TSM) を使用して、すべてのデータベース・パーティションおよびスナップショット・バックアップ用のバックアップ・システムで、NFS エクスポートおよび NFS 共有が可能です。

DB2 拡張コピー・サービス (ACS) セットアップ・スクリプト setup.sh

setup.sh スクリプトは、DB2 拡張コピー・サービス (ASC) を活動化および構成します。

ロケーション

setup.sh スクリプトは、sqllib/acs ディレクトリーにあります。

構文

以下は、**setup.sh** の構文です。

```
使用法: setup.sh -a action
                 -d DB2_Instance_Directory
                 -u Instance_user_ID_name
                 -g Instance_primary_group_name
```

ここで、*action* は以下のいずれかになります。

- start
- stop

- query
- enable
- disable

使用法

データベース・マネージャー・インスタンスの作成中の、IBM Data Server のアップグレード時に、データベース・マネージャーは自動的に **setup.sh** を呼び出して、DB2 ACS 機能を活動化します。

以下のように、手動で **setup.sh** スクリプトを呼び出すこともできます。

DB2 ACS の活動化

前述のパラメーターを使用して、root 権限を持つユーザーとして **setup.sh** を実行することにより、DB2 ACS を活動化できます。

DB2 ACS の構成

setup.sh をパラメーターを使用せずに実行することにより、DB2 ACS を構成できます。パラメーターを使用せずに **setup.sh** を実行すると、ウィザードが DB2 ACS の構成を先導していきます。

setup.sh スクリプトの実行の重要な結果の 1 つは、sqllib/acs ディレクトリーの DB2 ACS 実行可能ファイルの所有権および許可が検証されることです。

DB2 拡張コピー・サービス (ACS) のアンインストール

DB2 ACS は、DB2 製品をアンインストールする際に、自動的にアンインストールされます。バージョン 9.7 フィックスパック 2 以降、**db2_deinstall** コマンド、DB2 セットアップ・ウィザード、または応答ファイルを使用して、DB2 ACS のみをアンインストールできるようになりました。

手順

DB2 ACS をアンインストールするには、以下のいずれかの方法を使用してください。

- 次のように、**db2_deinstall** コマンドに **-F ACS** パラメーターを指定して実行します。

  ```
  db2_deinstall -F ACS
  ```

- DB2 セットアップ・ウィザードで「**既存の製品を操作**」をクリックし、既にインストール済みの DB2 コピーから、DB2 ACS コンポーネントの選択を解除します。
- 次のように、**ACS** キーワードを応答ファイルに追加します。

  ```
  REMOVE_COMP = ACS
  ```

次のタスク

ログ・ファイルにあるメッセージをチェックします。ログ・ファイルは、次のディレクトリーにあります。

- ルート・インストールの場合: /tmp/db2_deinstall.log.*processID*。ここで *processID* は、DB2 インストーラーのプロセス ID を表しています。
- 非ルート・インストールの場合: /tmp/db2_deinstall_*userID*.log。ここで *userID* は、非ルート・インストールを所有するユーザー ID を表しています。

次のように、インストール済みコンポーネントをすべてリストする **db2ls** コマンドを使用して、DB2 ACS が削除されたことを確認します。

```
db2ls -q -a -b base-install-path
```

ここで *base-install-path* は、照会するディレクトリーを表しています。

DB2 拡張コピー・サービス (ACS) API

DB2 拡張コピー・サービス (ACS) アプリケーション・プログラミング・インターフェース (API) は、データベース・マネージャーがスナップショット・バックアップ操作を実行するために、ストレージ・ハードウェアとの通信に使用する関数のセットを定義します。

スナップショット・バックアップ操作およびリストア操作を実行するには、ご使用のストレージ・デバイス用の DB2 ACS API ドライバーが必要です。 IBM Data Server には、以下のストレージ・ハードウェアのための DB2 ACS API ドライバーが組み込まれています。

- IBM TotalStorage SAN ボリューム・コントローラー
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N シリーズ
- NetApp V-series
- NetApp FAS series

DB2 拡張コピー・サービス (ACS) API 関数

データベース・マネージャーは、DB2 ACS API 関数を通して DB2 ACS 要求をストレージ・ハードウェアに伝えます。

db2ACSQueryApiVersion - DB2 拡張コピー・サービス (ACS) API の現行バージョンを戻す

DB2 拡張コピー・サービス (ACS) API の現行バージョンを戻します。

API 組み込みファイル

```
db2ACSApi.h
```

API およびデータ構造構文

```
db2ACS_Version db2ACSQueryApiVersion();
```

パラメーター

なし。

使用上の注意

可能な戻り値:

- DB2ACS_API_VERSION1
- DB2ACS_API_VERSION_UNKNOWN

db2ACSInitialize - DB2 拡張コピー・サービス (ACS) のセッションの初期化

新規の DB2 拡張コピー・サービス (ACS) のセッションを初期化します。この呼び出しによって、データベース・マネージャーの DB2 ACS ライブラリーとストレージ・ハードウェア用の DB2 ACS API ドライバーとの間の通信が確立されます。

組み込みファイル

db2ACSApi.h

構文およびデータ構造

```
/* ============================================================================
 * Session Initialization
 * ============================================================================ */
db2ACS_RC db2ACSInitialize(
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

db2ACSInitialize を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->session
pControlBlock->options

DB2 ACS API ドライバーは、戻す前に以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 17. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INIT_FAILED	データベース・マネージャーが DB2 ACS セッションの初期化を試みましたが、初期化が失敗しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_COMM_ERROR	磁気テープ・ドライブなどのストレージ・デバイスに通信エラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_NO_DEV_AVAIL	現在、使用可能な磁気テープ・ドライブなどのストレージ・デバイスがありません。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。

- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

db2ACSQueryAPIVersion() 呼び出しを除くすべての DB2 ACS API 呼び出しを行う前に、データベース・マネージャーは db2ACSInitialize() を呼び出す必要があります。 db2ACSInitialize() を呼び出して DB2 ACS セッションを確立した後、データベース・マネージャーは、DB2 ACS の照会、読み取り、書き込み、または削除操作の任意の組み合わせを実行できます。 db2ACSTerminate() を呼び出すことによって、データベース・マネージャーは DB2 ACS セッションを終了できます。

db2ACSTerminate - DB2 拡張コピー・サービス (ACS) セッションの終了

DB2 拡張コピー・サービス (ACS) セッションを終了します。

組み込みファイル

db2ACSApi.h

構文およびデータ構造

```
/* ============================================================================
 * Session Termination
 * ============================================================================ */
db2ACS_RC db2ACSTerminate(
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

データベース・マネージャーは、db2ACSInitialize() を呼び出す前に、このパラメーター用のメモリーを割り振っています。このメモリーの解放は、db2ACSTerminate() の後に、データベース・マネージャーが行います。

db2ACSTerminate() を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->options

DB2 ACS API ドライバーは、pControlBlock->vendorInfo.vendorCB 内のメモリーを無効にして、解放する場合があります。

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2

ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 18. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	このセッションに割り振られたすべてのメモリーを解放し、終了します。
DB2ACS_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

DB2 ACS API ドライバーは、db2ACSTerminate() の DB2 ACS セッションに割り振ったすべてのメモリーを解放する必要があります。

db2ACSTerminate() がエラーなしで完了したかどうかに関係なく、データベース・マネージャーは、この DB2 ACS セッションで、db2ACSInitialize() を再び呼び出してからでないと、どの DB2 ACS 関数も再び呼び出せません。

db2ACSPrepare - スナップショット・バックアップ操作の実行の準備

スナップショット・バックアップが実行されると、データベース・マネージャーはデータベースを中断状態にします。 db2ACSPrepare() は、データベース・マネージャーがデータベースを中断状態にする前に、スナップショット・バックアップ操作の実行を準備するための各ステップを実行します。

組み込みファイル

db2ACSApi.h

構文およびデータ構造

```
/* ============================================================================
 * Prepare
 * ============================================================================ */
db2ACS_RC db2ACSPrepare(
               db2ACS_GroupList        * pGroupList,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

pGroupList

データ・タイプ: db2ACS_GroupList *

db2ACS_GroupList には、スナップショットのバックアップ操作に組み込まれているグループのリストが含まれています。

pGroupList が NULL の場合、すべてのグループ (パス) がスナップショット・バックアップ操作に組み込まれます。

pGroupList が NULL でない場合は、以下のようになります。

- **pGroupList** に、スナップショット・バックアップ操作に組み込まれるグループ (パス) のリストが入ります。
- **pGroupList** のメモリーの割り振りおよび解放は、データベース・マネージャーが行います。
- データベース・マネージャーは、**pGroupList** を db2ACSPrepare() に渡す前に、以下のフィールドにデータを設定します。

  pGroupList->numGroupID
  pGroupList->id

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSPrepare を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

  pControlBlock->handle
  pControlBlock->vendorInfo
  pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 19. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、 518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

db2ACSPrepare() が成功した場合、データベース・マネージャーは、db2ACSSnapshot() を呼び出す前に、データベースを中断状態にします。

db2ACSBeginOperation - DB2 拡張コピー・サービス (ACS) 操作の開始

DB2 拡張コピー・サービス (ACS) 操作を開始します。

組み込みファイル

db2ACSApi.h

構文およびデータ構造

```
/* ============================================================================
 * Operation Begin
 *
 * A valid ACS operation is specified by passing an ObjectType OR'd with one of
 * the following Operations, such as:
 *
 *    (DB2ACS_OP_CREATE | DB2ACS_OBJTYPE_SNAPSHOT)
 * ============================================================================ */
db2ACS_RC db2ACSBeginOperation(
               db2ACS_Operation           operation,
               db2ACS_CB                * pControlBlock,
               db2ACS_ReturnCode        * pRC );
```

パラメーター

operation

データ・タイプ: db2ACS_Operation。

operation は、DB2 ACS 操作の開始を示すビット・マスクであり、含まれているオブジェクトのタイプです。

操作タイプ:

DB2ACS_OP_CREATE
DB2ACS_OP_READ
DB2ACS_OP_DELETE

オブジェクト・タイプ:

DB2ACS_OBJTYPE_BACKUP
DB2ACS_OBJTYPE_LOG
DB2ACS_OBJTYPE_LOADCOPY
DB2ACS_OBJTYPE_SNAPSHOT

例: (DB2ACS_OP_CREATE | DB2ACS_OBJTYPE_SNAPSHOT) または (DB2ACS_OP_DELETE | DB2ACS_OBJTYPE_LOADCOPY)。

データベース・マネージャーは、**operation** を db2ACSBeginOperation() 関数呼び出しに渡します。

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSBeginOperation() を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを追加します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

operation が DB2ACS_OP_CREATE または DB2ACS_OP_READ の場合、データベース・マネージャーは以下のフィールドにもデータを追加します。

pControlBlock->operation

pControlBlock->operation 内に含まれている情報は、特定の DB2 ACS 操作のコンテキスト内でのみ有効です。pControlBlock->operation は、db2ACSBeginOperation() 中に設定され、db2ACSEndOperation() が戻るまで未変更のままです。データベース・マネージャーも DB2 ACS API ドライバーも、DB2 ACS 操作の有効範囲外では pControlBlock->operation を参照すべきではありません。

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 20. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_OPTIONS	データベース・マネージャーが無効なオプションを指定しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、 518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

なし。

db2ACSEndOperation - DB2 拡張コピー・サービス (ACS) 操作の終了

DB2 拡張コピー・サービス (ACS) 操作を終了します。

組み込みファイル

```
db2ACSApi.h
```

構文およびデータ構造

```
/* ============================================================================
 * Operation End
 * ============================================================================ */
db2ACS_RC db2ACSEndOperation(
               db2ACS_EndAction          endAction,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

endAction

データ・タイプ: db2ACS_EndAction。

endAction は、DB2 ACS API ドライバーが DB2 ACS 操作を終了する方法を示すビット・マスクです。

値は以下のとおりです。

DB2ACS_END_COMMIT
DB2ACS_END_ABORT

データベース・マネージャーは、**endAction** を db2ACSEndOperation() 関数呼び出しに渡します。

pControlBlock

データ・タイプ: db2ACS_CB

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSEndOperation を呼び出す前に、データベース・マネージャーは以下のフィールドを追加します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2

ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 21. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_COMMIT_FAILED	DB2 ACS API ドライバーがトランザクションをコミットできませんでした。	
DB2ACS_RC_ABORT_FAILED	データベース・マネージャーが DB2 ACS 操作のアボートを試みましたが、アボートの試みが失敗しました。	

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

データベース・マネージャーが DB2ACS_END_ABORT を **endAction** パラメーターとして渡す場合、スナップショットのバックアップ・オブジェクトが削除される結果になります。

db2ACSBeginQuery - スナップショット・バックアップ・オブジェクトに関する照会の開始

リストア操作に使用可能なスナップショット・バックアップ・オブジェクトに関する DB2 拡張コピー・サービス (ACS) 照会操作を開始します。

組み込みファイル

db2ACSApi.h

構文およびデータ構造

```
db2ACS_RC db2ACSBeginQuery(
              db2ACS_QueryInput       * pQueryInput,
              db2ACS_CB               * pControlBlock,
              db2ACS_ReturnCode       * pRC );
```

パラメーター

pQueryInput

データ・タイプ: db2ACS_QueryInput *

db2ACS_QueryInput には db2ACS_ObjectInfo と同じフィールドがあります。 db2ACS_ObjectInfo には、DB2 拡張コピー・サービス (ACS) API を使用して作成したオブジェクトに関する情報が入ります。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

db2ACSBeginQuery を呼び出す前に、データベース・マネージャーは **pQueryInput** の各フィールドにデータを設定します。

DB2 ACS API ドライバーは、照会で以下のワイルドカードの使用をサポートする必要があります。

- ストリング・フィールド内の DB2ACS_WILDCARD
- データベース・パーティション・フィールドの DB2ACS_ANY_PARTITIONNUM
- 32 ビット符号なし整数 (Uint32) フィールドの DB2ACS_ANY_UINT32

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSBeginQuery を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 22. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、 518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

db2ACSBeginQuery() は照会データを戻しません。

db2ACSGetNextObject - リストアに使用できるスナップショット・バックアップ・オブジェクトのリストの次項目を戻す

リストア操作に使用できるスナップショット・バックアップ・オブジェクトのリスト内の次の項目を戻します。

組み込みファイル

db2ACSApi.h

構文およびデータ構造

```
db2ACS_RC db2ACSGetNextObject(
              db2ACS_QueryOutput      * pQueryOutput,
              db2ACS_CB               * pControlBlock,
              db2ACS_ReturnCode       * pRC );
```

パラメーター

pQueryOutput

データ・タイプ: db2ACS_QueryOutput *

db2ACS_QueryOutput には、スナップショットのバックアップ・オブジェクトについての照会結果情報が含まれています。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pQueryOutput** の各フィールドにデータを設定します。

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSGetNextObject を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 23. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

表 23. 戻りコード (続き)

戻りコード	説明	注
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_OBJ_NOT_FOUND	DB2 ACS API ドライバーが、データベース・マネージャーによって指定されたスナップショット・バックアップ・オブジェクトを検出できませんでした。	関数呼び出しは失敗ではありませんでしたが、db2ACSBeginQuery() に渡す基準に一致するスナップショット・バックアップ・オブジェクトがありません。
DB2ACS_RC_END_OF_DATA	DB2 ACS API ドライバーが、これ以上のスナップショット・バックアップ・オブジェクトを検出できません。	関数呼び出しは失敗ではありませんでしたが、db2ACSBeginQuery() に渡す基準に一致するスナップショット・バックアップ・オブジェクトがもうありません。
DB2ACS_RC_MORE_DATA	ストレージ・ロケーションからデータベース・マネージャーへ転送するデータがまだあります。	db2ACSBeginQuery() に渡す基準に一致するスナップショット・バックアップ・オブジェクトに関する情報が戻され、db2ACSBeginQuery() に渡す基準に一致するスナップショット・バックアップ・オブジェクトがまだあります。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

データベース・マネージャーは、db2ACSGetNextObject() を呼び出す前に、db2ACSBeginQuery() を呼び出す必要があります。データベース・マネージャーは、db2ACSBeginQuery() に渡された db2ACS_QueryInput パラメーターに検索基準を指定します。

db2ACSGetNextObject() は、db2ACSBeginQuery()に渡された検索基準に一致する 1 つのスナップショット・バックアップ・オブジェクトに関する情報を戻します。db2ACSGetNextObject() が DB2ACS_RC_MORE_DATA を戻した場合、データベース・マネージャーは db2ACSGetNextObject() を再び呼び出して、検索基準に一致する別のスナップショット・バックアップ・オブジェクトに関する情報を受け取れます。db2ACSGetNextObject() が DB2ACS_RC_END_OF_DATA を戻した場合、検索基準に一致するスナップショット・バックアップ・オブジェクトはもうありません。

db2ACSEndQuery - スナップショット・バックアップ・オブジェクトに関する照会の終了

データベース・マネージャーは、DB2 拡張コピー・サービス (ACS) API 関数の db2ACSBeginQuery() および db2ACSGetNextObject() を使用して、リストア操作に使用できるスナップショット・バックアップ・オブジェクトについて照会します。db2ACSEndQuery() は、この DB2 ACS 照会セッションを終了します。

組み込みファイル

```
db2ACSApi.h
```

構文およびデータ構造

```
db2ACS_RC db2ACSEndQuery(
              db2ACS_CB               * pControlBlock,
              db2ACS_ReturnCode       * pRC );
```

パラメーター

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSEndQuery() を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 24. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

データベース・マネージャーは、この DB2 ACS セッションでは、db2ACSBeginQuery() を再び呼び出してからでないと、db2ACSGetNextObject() を再び呼び出せません。

db2ACSSnapshot - DB2 拡張コピー・サービス (ACS) 操作の実行

DB2 拡張コピー・サービス (ACS) 操作を実行します。

組み込みファイル

```
db2ACSApi.h
```

構文およびデータ構造

```
typedef union db2ACS_ReadList
{
   db2ACS_GroupList           group;
} db2ACS_ReadList;


db2ACS_RC db2ACSSnapshot(
               db2ACS_Action             action,
               db2ACS_ObjectID           objectID,
               db2ACS_ReadList         * pReadList,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

action データ・タイプ: db2ACS_Action

実行する DB2 ACS アクションのタイプ。値:

DB2ACS_ACTION_WRITE
DB2ACS_ACTION_READ_BY_OBJECT
DB2ACS_ACTION_READ_BY_GROUP

データベース・マネージャーは、**action** を db2ACSSnapshot() に渡します。

objectID

データ・タイプ: db2ACS_ObjectID

db2ACS_ObjectID は、各保管オブジェクトごとのユニーク ID であり、照会によってストレージ・リポジトリーに戻されます。 db2ACS_ObjectID は、単一の DB2 ACS セッションの時間フレーム内でのみ、ユニークで永続的であることが保証されています。

データベース・マネージャーは、db2ACSBeginOperation() 呼び出しの **operation** として、DB2ACS_OP_READ または DB2ACS_OP_DELETE を指定した場合、 **objectID** の値を db2ACSSnapshot() に渡します。

pReadList

データ・タイプ: db2ACS_ReadList *

db2ACS_ReadList には、グループのリストが含まれています。

pReadList は、**action** が DB2ACS_ACTION_READ_BY_GROUP の場合のみ使用されます。

action が DB2ACS_ACTION_READ_BY_GROUP の場合、データベース・マネージャーは、db2ACSSnapshot() を呼び出す前に、**pReadLIst** の各フィールド用のメモリーを割り振り、データを設定し、後で **pReadLIst** のメモリーを解放する必要があります。

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSSnapshot を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 25. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

データベース・マネージャーは、db2ACSPartition()、db2ACSPrepare()、および db2ACSSnapshot() を呼び出す前に、db2ACSBeginOperation() を呼び出します。データベース・マネージャーは、DB2 ACS API ドライバーが実行する必要がある DB2 ACS 操作のタイプを、 db2ACSBeginOperation() 呼び出しの **operation** パラメーターに指定します。

db2ACSPartition - データベース・パーティションのターゲット・データのグループ化

1 つのデータベース・パーティションに属しているものとして、データベース・マネージャーによってリストされた各パスにグループ ID を関連付けます。

組み込みファイル

```
db2ACSApi.h
```

構文およびデータ構造

```
/* ============================================================================
 * Partition
 * ============================================================================ */
db2ACS_RC db2ACSPartition(
               db2ACS_PathList         * pPathList,
               db2ACS_CreateObjectInfo * pCreateObjInfo,
               db2ACS_CB               * PControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

pPathList

データ・タイプ: db2ACS_PathList

db2ACS_PathList には、DB2 ACS 操作に固有の各データベース・パスの追加情報を含む、データベース・パスのリストが含まれています。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

db2ACS_PathList 構造の **entry** フィールドは、タイプ db2ACS_PathEntry のエレメントの配列です。db2ACS_PathEntry には、データベース・パスについての情報が含まれています。

db2ACSPartition を呼び出す前に、データベース・マネージャーは、**pPathList** 内の各 db2ACS_PathEntry 項目の以下のフィールドにデータを設定します。

- **path**
- **type**
- **toBeExcluded**

このデータベース・パーティションに属しているとデータベース・マネージャーによって識別されるすべてのパスは、DB2 ACS API ドライバーによってグループ ID を与えられます。 DB2 ACS API ドライバーは、戻す前に、**pPathList** 内の各 db2ACS_PathEntry の **groupID** フィールドにデータを設定します。

pCreateObjInfo

データ・タイプ: db2ACS_CreateObjectInfo

db2ACS_CreateObjectInfo には、DB2 ACS バックアップ・オブジェクト作成に関する情報が入ります。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

データベース・マネージャーは、db2ACSPartition を呼び出す前に、**pCreateObjInfo** の各フィールドにデータを設定します。

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSPartition を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 26. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INIT_FAILED	データベース・マネージャーが DB2 ACS セッションの初期化を試みましたが、初期化が失敗しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_OBJ_OUT_OF_SCOPE	データベース・マネージャーが、DB2 ACS API ドライバーによって管理されていないリカバリー・オブジェクトに、DB2 ACS 操作の実行を試みました。	

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

DB2 拡張コピー・サービスは、単一のデータベース・パーティション上のデータをアトミックに処理します。すなわち、1 つのデータベース・パーティションのデータは、アクションが複数のデータベース・パーティションに関連する操作の一部で

ある場合でも、他のデータベース・パーティションからは独立して、一緒にバックアップまたはリストアされます。 db2ACSPartition は、単一のデータベース・パーティションに関するパス情報をグループ化します。

データベース・マネージャーは、db2ACSSnapshot を呼び出す前に、db2ACSPartition を呼び出します。データベース・マネージャーは、**pPathList** パラメーター内の当該データベース・パーティションに関連するすべてのパスをリストします。データベース・マネージャーは、db2ACSSnapshot に渡される **pReadList** パラメーター内のパスのサブセットを指定して、**pPathList** にリストされているパスのサブセットに DB2 ACS 操作を実行できます。

db2ACSVerify - DB2 拡張コピー・サービス (ACS) 操作の正常な完了の検証

DB2 拡張コピー・サービス (ACS) 操作の成功を検証します。

組み込みファイル

```
db2ACSApi.h
```

構文およびデータ構造

```
/* ============================================================================
 * Verify
 * ============================================================================ */
db2ACS_RC db2ACSVerify(
               db2ACS_PostObjectInfo   * pPostObjInfo,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

pPostObjInfo

データ・タイプ: db2ACS_PostObjectInfo

db2ACS_DB2ID は、スナップショットのバックアップ・オブジェクトの作成時には認識されないものの、オブジェクト・リポジトリーで保守されなければならないデータのセットです。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

データベース・マネージャーは、db2ACSVerify を呼び出す前に、**pPostObjInfo** の各フィールドにデータを設定します。 **pPostObjInfo** には、DB2 ACS 操作後に関連する情報が入ります。例えば、スナップショット・バックアップが成功した後、**pPostObjInfo** には最初のアクティブ・ログ・ファイルが入る可能性があります。 DB2 ACS 操作後に関連するデータがない場合、データベース・マネージャーは **pPostObjInfo** を NULL に設定します。

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSVerify を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 27. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。

- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

db2ACSVerify が、スナップショット・バックアップ操作が成功したことを戻した場合、スナップショット・バックアップによって生成されたリカバリー・オブジェクトがリストア操作で使用できることを意味します。

db2ACSDelete - DB2 拡張コピー・サービス (ACS) を使用して作成したリカバリー・オブジェクトの削除

DB2 拡張コピー・サービス (ACS) を使用して作成したリカバリー・オブジェクトを削除します。

組み込みファイル

```
db2ACSApi.h
```

構文およびデータ構造

```
/* ============================================================================
 * Delete
 * ============================================================================ */
db2ACS_RC db2ACSDelete(
               db2ACS_ObjectID           objectID,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

objectID

データ・タイプ: db2ACS_ObjectID

db2ACS_ObjectID は、各保管オブジェクトごとのユニーク ID であり、照会によってストレージ・リポジトリーに戻されます。 db2ACS_ObjectID は、単一の DB2 ACS セッションの時間フレーム内でのみ、ユニークで永続的であることが保証されています。

データベース・マネージャーは、db2ACSQuery() を使用して、db2ACSDelete に渡す有効な **objectID** を入手できます。

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSDelete を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 28. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	指定したオブジェクトが削除されます。このオブジェクトには、DB2 ACS 操作が行えなくなります。
DB2ACS_RC_DELETE_FAILED	DB2 ACS API ドライバーが、データベース・マネージャーによって指定されたスナップショット・バックアップ・オブジェクトの削除に失敗しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_OBJ_NOT_FOUND	DB2 ACS API ドライバーが、データベース・マネージャーによって指定されたスナップショット・バックアップ・オブジェクトを検出できませんでした。	

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。

- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

データベース・マネージャーが db2ACSDelete を呼び出すと、DB2 ACS API ドライバーが **objectID** によって示されたリカバリー・オブジェクトを削除します。

データベース・マネージャーが db2ACSDelete を呼び出すのは、ユーザーが DELETE パラメーターを指定して **db2acsutil** を呼び出した場合です。

db2ACSStoreMetaData - DB2 拡張コピー・サービス (ACS) を使用して生成したリカバリー・オブジェクトに関するメタデータの保管

DB2 拡張コピー・サービス (ACS) を使用して作成したリカバリー・オブジェクトに関するメタデータを保管します。

組み込みファイル

```
db2ACSApi.h
```

構文およびデータ構造

```
db2ACS_RC db2ACSStoreMetaData(
               db2ACS_MetaData         * pMetaData,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

pMetaData

データ・タイプ: db2ACS_MetaData

db2ACS_MetaData は、スナップショットのバックアップ・メタデータを保管します。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

pMetaData の **data** フィールドに保管されるメタデータは、データベース・マネージャー内部のものであり、変更される可能性があるので、DB2 ACS API ドライバーは、このデータをバイナリー・ストリームとして処理します。

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSStoreMetaData を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 29. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

スナップショット・バックアップ操作は、db2ACSInitialize、db2ACSBeginOperation、db2ACSPrepare、および db2ACSSnapshot などの複数の DB2 ACS API 関数呼び出しから構成されます。 db2ACSStoreMetaData もこの操作の一部です。db2ACSStoreMetaData を含む上記のすべての API 呼び出しが成功しなければ、スナップショット・バックアップ操作も成功しません。db2ACSStoreMetaData が失敗した場合、DB2 ACS バックアップ操作によって生成されたリカバリー・オブジェクトは、使用できません。

db2ACSRetrieveMetaData - DB2 拡張コピー・サービス (ACS) を使用して生成したリカバリー・オブジェクトに関するメタデータの検索

DB2 拡張コピー・サービス (ACS) を使用して作成したリカバリー・オブジェクトに関するメタデータを検索します。

組み込みファイル

```
db2ACSApi.h
```

構文およびデータ構造

```
db2ACS_RC db2ACSRetrieveMetaData(
               db2ACS_MetaData         * pMetaData,
               db2ACS_ObjectID           objectID,
               db2ACS_CB               * pControlBlock,
               db2ACS_ReturnCode       * pRC );
```

パラメーター

pMetaData

データ・タイプ: db2ACS_MetaData

db2ACS_MetaData は、スナップショットのバックアップ・メタデータを保管します。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

pMetaData の **data** フィールドに保管されるメタデータは、データベース・マネージャー内部のものであり、変更される可能性があるので、DB2 ACS API ドライバーは、このデータをバイナリー・ストリームとして処理します。

objectID

データ・タイプ: db2ACS_ObjectID

db2ACS_ObjectID は、各保管オブジェクトごとのユニーク ID であり、照会によってストレージ・リポジトリーに戻されます。 db2ACS_ObjectID は、単一の DB2 ACS セッションの時間フレーム内でのみ、ユニークで永続的であることが保証されています。

データベース・マネージャーは、db2ACSQuery() を使用して、db2ACSRetrieveMetaData() に渡す有効な **objectID** を入手できます。

pControlBlock

データ・タイプ: db2ACS_CB *

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

db2ACSRetrieveMetaData を呼び出す前に、データベース・マネージャーは以下のフィールドにデータを設定します。

pControlBlock->handle
pControlBlock->vendorInfo
pControlBlock->options

pRC データ・タイプ: db2ACS_ReturnCode *

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

データベース・マネージャーは、このパラメーターのメモリーを割り振り、インスタンス化されたオブジェクトを示すポインターを関数に渡します。このメモリーの解放は、データベース・マネージャーが行います。

DB2 ACS API ドライバーは、戻す前に **pRC** の各フィールドにデータを設定します。

戻りコード

表 30. 戻りコード

戻りコード	説明	注
DB2ACS_RC_OK	操作は正常に終了しました。	
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

表 30. 戻りコード (続き)

戻りコード	説明	注
DB2ACS_RC_OBJ_NOT_FOUND	DB2 ACS API ドライバーが、データベース・マネージャーによって指定されたスナップショット・バックアップ・オブジェクトを検出できませんでした。	DB2 ACS API ドライバーがエラーを検出しました。データベース・マネージャーは DB2 ACS API セッションを使用できません。

DB2 ACS API ドライバーは、エラーを検出した場合、DB2 ACS 操作をアボートする可能性があります。 DB2 ACS セッションは、以下を除くアクションでは使用できません。

- db2ACSBeginQuery() の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndQuery() を呼び出せます。
- db2ACSBeginOperation の呼び出しが成功した後に、データベース・マネージャーは db2ACSEndOperation を呼び出せます。
- db2ACSInitialize の呼び出しが成功した後に、データベース・マネージャーは db2ACSTerminate を呼び出せます。

DB2 ACS API の戻りコードについての詳細は、518 ページの『DB2 拡張コピー・サービス (ACS) API の戻りコード』 を参照してください。

使用上の注意

なし。

DB2 拡張コピー・サービス (ACS) API のデータ構造

DB2 拡張コピー・サービス (ACS) API 関数を呼び出すには、DB2 ACS API のデータ構造を使用する必要があります。

db2ACS_BackupDetails DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_BackupDetails には、スナップショット・バックアップ操作に関する情報が入ります。

```
/* ------------------------------------------------------------------------ */
typedef struct db2ACS_BackupDetails
{
   /* A traditional DB2 backup can consist of multiple objects (logical tapes),
    * where each object is uniquely numbered with a non-zero natural number.
    * ------------------------------------------------------------------------ */
   db2Uint32                  sequenceNum;

   char                       imageTimestamp[SQLU_TIME_STAMP_LEN + 1];
} db2ACS_BackupDetails;
```

sequenceNum

データ・タイプ: db2Uint32

固有の番号によってバックアップ・オブジェクトを識別します。

imageTimestamp

データ・タイプ: char[]

長さが SQLU_TIME_STAMP_LEN + 1 の文字ストリング。

db2ACS_CB DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_CB には、DB2 ACS セッションの初期化および終了に必要な基本情報が入ります。

```
/* ****************************************************************************
 * DB2 Backup Adapter Control Block
 * ************************************************************************** */
typedef struct db2ACS_CB
{
   /* Output:  Handle value for this session.
    * -------------------------------------------------------------------------- */
   db2Uint32                  handle;
   db2ACS_VendorInfo          vendorInfo;

   /* Input fields and parameters.
    * -------------------------------------------------------------------------- */
   db2ACS_SessionInfo         session;
   db2ACS_Options             options;

   /* Operation info is optional, possibly NULL, and is only ever valid
    * within the context of an operation (from call to BeginOperation() until
    * the EndOperation() call returns).
    *
    * The operation info will be present during creation or read operations
    * of snapshot and backup objects.
    * -------------------------------------------------------------------------- */
   db2ACS_OperationInfo     * operation;
} db2ACS_CB;
```

handle データ・タイプ: db2Uint32

DB2 ACS セッションを参照するためのハンドル。

vendorInfo

データ・タイプ: db2ACS_VendorInfo

db2ACS_VendorInfo には、DB2 ACS API ドライバーに関する情報が入ります。

session データ・タイプ: db2ACS_SessionInfo

db2ACS_SessionInfo には、DB2 ACS セッションに関するすべての情報が入ります。

options

データ・タイプ: db2ACS_Options

db2ACS_Options は、DB2 ACS 操作に使用するオプションを指定します。このストリングの内容は、DB2 ACS API ドライバーに固有です。

operation

データ・タイプ: db2ACS_OperationInfo *

db2ACS_OperationInfo には、スナップショット・バックアップ操作に関する情報が入ります。

db2ACS_CreateObjectInfo DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_CreateObjectInfo には、DB2 ACS バックアップ・オブジェクト作成に関する情報が入ります。

```
/* ============================================================================
 * Object Creation Parameters.
 * ============================================================================ */
typedef struct db2ACS_CreateObjectInfo
{
   db2ACS_ObjectInfo          object;
   db2ACS_DB2ID               db2ID;

   /* ------------------------------------------------------------------------
    * The following fields are optional information for the database manager
    * to use as it sees fit.
    * ------------------------------------------------------------------------ */

   /* Historically both the size estimate and management
    * class parameters have been used by the TSM client API for traditional
    * backup objects, log archives, and load copies, but not for snapshot
    * backups.
    * ------------------------------------------------------------------------ */
   db2Uint64                  sizeEstimate;
   char                       mgmtClass[DB2ACS_MAX_MGMTCLASS_SZ + 1];

   /* The appOptions is a copy of the iOptions field of flags passed to DB2's
    * db2Backup() API when this execution was initiated.  This field will
    * only contain valid data when creating a backup or snapshot object.
    * ------------------------------------------------------------------------ */
   db2Uint32                  appOptions;
} db2ACS_CreateObjectInfo;
```

object データ・タイプ: db2ACS_ObjectInfo

db2ACS_ObjectInfo には、DB2 拡張コピー・サービス (ACS) API を使用して作成したオブジェクトに関する情報が入ります。

db2ID データ・タイプ: db2ACS_DB2ID

db2ACS_DB2ID は IBM Data Server を識別します。

sizeEstimate

データ・タイプ: db2Uint64

作成中のバックアップ・オブジェクトのサイズの見積もり。この見積もりは、ログ・アーカイブ、ロード・コピー、またはスナップショット・バックアップ・オブジェクトには適用されません。

mgmtClass

データ・タイプ: db2ACS_MgmtClass

長さ db2ACS_MAX_MGMTCLASS_SZ + 1 の文字ストリング。

これは、スナップショット・バックアップ・オブジェクトには適用されません。

appOptions

データ・タイプ: db2Uint32

スナップショット・バックアップを開始したバックアップ・コマンドに渡されたバックアップ・オプションのコピー。

db2ACS_DB2ID DB2 拡張コピー・サービス (ACS) API データ構造

db2ACS_DB2ID は IBM Data Server を識別します。

```
/* ============================================================================
 * DB2 Data Server Identifier
 * ============================================================================ */
```

```
typedef struct db2ACS_DB2ID
{
   db2Uint32                 version;
   db2Uint32                 release;
   db2Uint32                 level;
   char                      signature[DB2ACS_SIGNATURE_SZ + 1];
} db2ACS_DB2ID;
```

version

データ・タイプ: db2Uint32

IBM Data Server のバージョン。例えば、9。

release データ・タイプ: db2Uint32

IBM Data Server のリリース・レベル。例えば、5。

level データ・タイプ: db2Uint32

IBM Data Server のレベル ID。例えば、0。

signature

データ・タイプ: char[]

長さが DB2ACS_SIGNATURE_SZ + 1 の文字ストリング。例えば、「SQL09050」。

db2ACS_GroupList DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_GroupList には、スナップショットのバックアップ操作に組み込まれているグループのリストが含まれています。

```
/* ============================================================================
 * Snapshot Group List
 *
 * This is an array of size 'numGroupIDs', indicating the set of groups that
 * are to be included in the snapshot operation.
 * ============================================================================ */
typedef struct db2ACS_GroupList
{
   db2Uint32                 numGroupIDs;
   db2Uint32               * id;
} db2ACS_GroupList;
```

numGroupIDs

データ・タイプ: db2Uint32。

id 配列内のグループの数。

id データ・タイプ: db2Uint32 *。

グループ ID の配列。識別されるグループは、スナップショットのバックアップ操作に組み込まれているグループ (またはパスのリスト) です。

db2ACS_LoadcopyDetails DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_LoadcopyDetails には、ロード・コピー操作に関する情報が入ります。

```
/* ---------------------------------------------------------------------------- */
typedef struct db2ACS_LoadcopyDetails
{
   /* Just like the BackupDetails, a DB2 load copy can consist of multiple
    * objects (logical tapes), where each object is uniquely numbered with a
```

```
    * non-zero natural number.
    * ------------------------------------------------------------------------ */
   db2Uint32                  sequenceNum;

   char                       imageTimestamp[SQLU_TIME_STAMP_LEN + 1];
} db2ACS_LoadcopyDetails;
```

sequenceNum

データ・タイプ: db2Uint32

固有の番号によってバックアップ・オブジェクトを識別します。

imageTimestamp

データ・タイプ: char[]

長さが SQLU_TIME_STAMP_LEN + 1 の文字ストリング。

db2ACS_LogDetails DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_LogDetails には、特定のデータベース・ログ・ファイルを識別する情報が入ります。

```
/* ------------------------------------------------------------------------ */
typedef struct db2ACS_LogDetails
{
   db2Uint32                  fileID;
   db2Uint32                  chainID;
} db2ACS_LogDetails;
```

fileID データ・タイプ: db2Uint32

データベース・ログ・ファイルのファイル名である番号。

chainID

データ・タイプ: db2Uint32

データベース・ログ・ファイル **fileID** の所属先のデータベース・ログ・ファイル・チェーンを識別する番号。

db2ACS_ObjectInfo DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_ObjectInfo には、DB2 拡張コピー・サービス (ACS) API を使用して作成したオブジェクトに関する情報が入ります。

```
/* ========================================================================
 * Object Description and Associated Information.
 *
 * This structure is used for both input and output, and its contents define
 * the minimum information that must be recorded about any object created
 * through this interface.
 * ======================================================================== */
typedef struct db2ACS_ObjectInfo
{
   db2ACS_ObjectType          type;
   SQL_PDB_NODE_TYPE          dbPartitionNum;

   char                       db[SQL_DBNAME_SZ + 1];
   char                       instance[DB2ACS_MAX_OWNER_SZ + 1];
   char                       host[SQL_HOSTNAME_SZ + 1];
   char                       owner[DB2ACS_MAX_OWNER_SZ + 1];

   union
   {
```

```
      db2ACS_BackupDetails    backup;
      db2ACS_LogDetails       log;
      db2ACS_LoadcopyDetails  loadcopy;
      db2ACS_SnapshotDetails  snapshot;
   } details;
} db2ACS_ObjectInfo;
```

type データ・タイプ: db2ACS_ObjectType

スナップショット・バックアップ・オブジェクトのタイプを指定します。値:

DB2ACS_OBJTYPE_ALL
DB2ACS_OBJTYPE_BACKUP
DB2ACS_OBJTYPE_LOG
DB2ACS_OBJTYPE_LOADCOPY
DB2ACS_OBJTYPE_SNAPSHOT

DB2ACS_OBJTYPE_ALL は照会のフィルターとしてのみ使用できます。タイプ 0 のオブジェクトはありません。

dbPartitionNum

データ・タイプ: SQL_PDB_NODE_TYPE

このデータベース・パーティションの ID。

db データ・タイプ: char[]

長さが SQL_DBNAME_SZ + 1 の文字ストリング。

instance

データ・タイプ: char[]

長さが DB2ACS_MAX_OWNER_SZ + 1 の文字ストリング。

host データ・タイプ: char[]

長さが SQL_HOSTNAME_SZ + 1 の文字ストリング。

owner データ・タイプ: char[]

長さが DB2ACS_MAX_OWNER_SZ + 1 の文字ストリング。

details

backup

データ・タイプ: db2ACS_BackupDetails

db2ACS_BackupDetails には、スナップショット・バックアップ操作に関する情報が入ります。

log データ・タイプ: db2ACS_LogDetails

db2ACS_LogDetails には、特定のデータベース・ログ・ファイルを識別する情報が入ります。

loadcopy

データ・タイプ: db2ACS_LoadcopyDetails

db2ACS_LoadcopyDetails には、ロード・コピー操作に関する情報が入ります。

snapshot

データ・タイプ: db2ACS_SnapshotDetails

db2ACS_SnapshotDetails には、スナップショット・バックアップ操作に関する情報が入ります。

db2ACS_ObjectStatus DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_ObjectStatus には、スナップショットのバックアップ操作の状況または進行についての情報、またはスナップショットのバックアップ・オブジェクトの状況またはユーザビリティーについての情報が含まれています。

```
typedef struct db2ACS_ObjectStatus
{
   /* The total and completed bytes refer only to the ACS snapshot backup
    * itself, not to the progress of any offloaded tape backup.
    *
    * A bytesTotal of 0 indicates that the progress could not be determined.
    * ------------------------------------------------------------------------ */
   db2Uint64                 bytesCompleted;
   db2Uint64                 bytesTotal;
   db2ACS_ProgressState      progressState;
   db2ACS_UsabilityState     usabilityState;
} db2ACS_ObjectStatus;
```

bytesCompleted

データ・タイプ: db2Uint64。

完了したスナップショットのバックアップの量 (バイト単位)。

bytesTotal

データ・タイプ: db2Uint64。

完了したスナップショットのバックアップのサイズ (バイト単位)。

progressState

データ・タイプ: db2ACS_ProgressState。

スナップショットのバックアップ操作の状態。値は以下のとおりです。

DB2ACS_PSTATE_UNKNOWN
DB2ACS_PSTATE_IN_PROGRESS
DB2ACS_PSTATE_SUCCESSFUL
DB2ACS_PSTATE_FAILED

usabilityState

データ・タイプ: db2ACS_UsabilityState。

スナップショットのバックアップ・オブジェクトの状態、スナップショットのバックアップ・オブジェクトの使用方法。 値は以下のとおりです。

DB2ACS_USTATE_UNKNOWN
DB2ACS_USTATE_LOCALLY_MOUNTABLE
DB2ACS_USTATE_REMOTELY_MOUNTABLE
DB2ACS_USTATE_REPETITIVELY_RESTORABLE
DB2ACS_USTATE_DESTRUCTIVELY_RESTORABLE
DB2ACS_USTATE_SWAP_RESTORABLE
DB2ACS_USTATE_PHYSICAL_PROTECTION

DB2ACS_USTATE_FULL_COPY
DB2ACS_USTATE_DELETED
DB2ACS_USTATE_FORCED_MOUNT
DB2ACS_USTATE_BACKGROUND_MONITOR_PENDING
DB2ACS_USTATE_TAPE_BACKUP_PENDING
DB2ACS_USTATE_TAPE_BACKUP_IN_PROGRESS
DB2ACS_USTATE_TAPE_BACKUP_COMPLETE

db2ACS_OperationInfo DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_OperationInfo には、スナップショット・バックアップ操作に関する情報が入ります。

```
/* ============================================================================
 * Operation Info
 *
 * The information contained within this structure is only valid within the
 * context of a particular operation.  It will be valid at the time
 * BeginOperation() is called, and will remain unchanged until EndOperation()
 * returns, but must not be referenced outside the scope of an operation.
 * ============================================================================ */
typedef struct db2ACS_OperationInfo
{
   db2ACS_SyncMode            syncMode;

   /* List of database and backup operation partitions.
    *
    * For details, refer to the db2ACS_PartitionList definition.
    * ------------------------------------------------------------------------- */
   db2ACS_PartitionList     * dbPartitionList;
} db2ACS_OperationInfo;
```

syncMode

データ・タイプ: db2ACS_SyncMode

個別のデータベース・パーティション上のバックアップ操作間の同期レベル。

値:

DB2ACS_SYNC_NONE
複数のデータベース・パーティション上の関連する操作間に同期がありません。複数のデータベース・パーティション間の同期を使用しない操作中に使用します。

DB2ACS_SYNC_SERIAL
複数のデータベース・パーティションに並行スナップショット・バックアップ操作を実行する場合に使用します。スナップショット・バックアップ操作が発行され、データベース・パーティションでの入出力 (IO) が並行にではなく、順番に再開された場合、各データベース・パーティションはその IO を中断させます。

SYNC_PARALLEL
複数のパーティションでスナップショット操作を並行して実行します。スナップショット・バックアップ操作に関連するすべてのデータベース・パーティションがスナップショット・バックアップ操作の準備を完了した後、入出力 (IO) はすべてのデータベース・パー

ティションで中断されます。残りのスナップショット・バックアップ・ステップは、関連するすべてのデータベース・パーティションで並行して行われます。

dbPartitionList

データ・タイプ: db2ACS_PartitionList *

db2ACS_PartitionList には、データベース内にあり、DB2 ACS 操作に関連しているデータベース・パーティションに関する情報が入ります。

db2ACS_Options DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_Options は、DB2 ACS 操作に使用するオプションを指定します。このストリングの内容は、DB2 ACS API ドライバーに固有です。

```
/* ============================================================================
 * DB2 Backup Adapter User Options
 * ============================================================================ */
typedef struct db2ACS_Options
{
   db2Uint32                  size;
   void                     * data;
} db2ACS_Options;
```

size データ・タイプ: db2Uint32

バイト単位での **data** のサイズ。

data データ・タイプ: void *

オプションが入っているメモリーのブロックを指すポインター。

db2ACS_PartitionEntry DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_PartitionEntry は db2ACS_PartitionList のエレメントです。

```
typedef struct db2ACS_PartitionEntry
{
   SQL_PDB_NODE_TYPE          num;
   char                       host[SQL_HOSTNAME_SZ + 1];
} db2ACS_PartitionEntry;
```

num データ・タイプ: SQL_PDB_NODE_TYPE

このデータベース・パーティション項目の ID。

host データ・タイプ: char[]

長さが SQL_HOSTNAME_SZ + 1 の文字ストリング。

db2ACS_PartitionList DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_PartitionList には、データベース内にあり、DB2 ACS 操作に関連しているデータベース・パーティションに関する情報が入ります。

```
typedef struct db2ACS_PartitionList
{
   db2Uint64                  numPartsInDB;
   db2Uint64                  numPartsInOperation;
   db2ACS_PartitionEntry    * partition;
} db2ACS_PartitionList;
```

numPartsInDB

データ・タイプ: db2Uint64

データベース内のデータベース・パーティションの数。

numPartsInOperation

データ・タイプ: db2Uint64

DB2 ACS 操作に関連しているデータベース・パーティションの数。

partition

データ・タイプ: db2ACS_PartitionEntry *

db2ACS_PartitionEntry は db2ACS_PartitionList のエレメントです。

db2ACS_PathEntry DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_PathEntry には、データベース・パスについての情報が含まれています。

```
typedef struct db2ACS_PathEntry
{
   /* INPUT: The path and type will be provided by the database server, as well
    *        as a flag indicating if the path is to be excluded from the backup.
    * ------------------------------------------------------------------------ */
   char                       path[DB2ACS_MAX_PATH_SZ + 1];
   db2ACS_PathType            type;
   db2Uint32                  toBeExcluded;

   /* OUTPUT: The group ID is to be provided by the backup adapter for use by
    *         the DB2 server.  The group ID will be used during with snapshot
    *         operations as an indication of which paths are dependent and must
    *         be included together in any snapshot operation.  Unique group IDs
    *         indicate that the paths in those groups are independent for the
    *         purposes of snapshot operations.
    * ------------------------------------------------------------------------ */
   db2Uint32                  groupID;
} db2ACS_PathEntry;
```

path データ・タイプ: char[]

長さが DB2ACS_MAX_PATH_SZ + 1 の文字ストリング。

type データ・タイプ: db2ACS_PathType。

パスのタイプ。値は以下のとおりです。

DB2ACS_PATH_TYPE_UNKNOWN
DB2ACS_PATH_TYPE_LOCAL_DB_DIRECTORY
DB2ACS_PATH_TYPE_DBPATH
DB2ACS_PATH_TYPE_DB_STORAGE_PATH
DB2ACS_PATH_TYPE_TBSP_CONTAINER
DB2ACS_PATH_TYPE_TBSP_DIRECTORY
DB2ACS_PATH_TYPE_TBSP_DEVICE
DB2ACS_PATH_TYPE_LOGPATH
DB2ACS_PATH_TYPE_MIRRORLOGPATH

toBeExcluded

データ・タイプ: db2Uint32。

スナップショットのバックアップに指定されたパスが含まれているかどうかを示すフラグ。値は以下のとおりです。

- 0 - スナップショットのバックアップにパスが含まれている
- 1 - スナップショットのバックアップにパスが含まれていない

groupID

データ・タイプ: db2Uint32。

グループ ID。

db2ACS_PathList DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_PathList には、DB2 ACS 操作に固有の各データベース・パスの追加情報を含む、データベース・パスのリストが含まれています。

```
/* ============================================================================
 * Snapshot File List
 *
 * This is an array of 'numEntries' db2ACS_PathEntry's, where each path entry is
 * a path to some storage on the DB2 server which is in use by the current
 * database.
 * ============================================================================ */

typedef struct db2ACS_PathList
{
   db2Uint32                  numEntries;
   db2ACS_PathEntry         * entry;
} db2ACS_PathList;
```

numEntries

データ・タイプ: db2Uint32。

entry 配列内のパス項目の数。

entry データ・タイプ: db2ACS_PathEntry。

db2ACS_PathEntry には、データベース・パスについての情報が含まれています。

db2ACS_PostObjectInfo DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_DB2ID は、スナップショットのバックアップ・オブジェクトの作成時には認識されないものの、オブジェクト・リポジトリーで保守されなければならないデータのセットです。

```
/* ============================================================================
 * The PostObjectInfo is a set of data that can not be known at object
 * creation time, but which must be maintained in the object repository.  This
 * is an optional field on the Verify() call, which may be NULL if there are
 * no post-operation updates to be made.
 * ============================================================================ */
typedef struct db2ACS_PostObjectInfo
{
   /* The first active log will only be valid when creating a backup or
    * snapshot object.  It will indicate the file number and chain id of the
    * first log required for recovery using this object.
    * ------------------------------------------------------------------------ */
   db2ACS_LogDetails          firstActiveLog;
} db2ACS_PostObjectInfo;
```

firstActiveLog

データ・タイプ: db2ACS_LogDetails。

db2ACS_LogDetails には、特定のデータベース・ログ・ファイルを識別する情報が入ります。

db2ACS_QueryInput および db2ACS_QueryOutput DB2 拡張コピー・サービス (ACS) API データ構造

db2ACS_QueryInput には、照会中のオブジェクトについての識別情報が含まれています。db2ACS_QueryOutput には、スナップショットのバックアップ・オブジェクトについての照会結果情報が含まれています。

```
/* ============================================================================
 * Unique Querying.
 *
 * When using this structure as query input, to indicate the
 * intention to supply a 'wildcard' search criteria, DB2 will supply:
 *
 *    -- character strings as "*".
 *    -- numeric values as (-1), cast as the appropriate signed or unsigned
 *       type.
 * ============================================================================ */

typedef struct db2ACS_ObjectInfo db2ACS_QueryInput;

typedef struct db2ACS_QueryOutput
{
   db2ACS_ObjectID            objectID;
   db2ACS_ObjectInfo          object;
   db2ACS_PostObjectInfo      postInfo;
   db2ACS_DB2ID               db2ID;
   db2ACS_ObjectStatus        status;

   /* Size of the object in bytes.
    * ------------------------------------------------------------------------  */
   db2Uint64                  objectSize;

   /* Size of the metadata associated with the object, if any, in bytes.
    * ------------------------------------------------------------------------  */
   db2Uint64                  metaDataSize;

   /* The creation time of the object is a 64bit value with a definition
    * equivalent to an ANSI C time_t value (seconds since the epoch, GMT).
    *
    * This field is equivalent to the file creation or modification time in
    * a traditional filesystem.  This should be created and stored
    * automatically by the BA subsystem, and a valid time value should be
    * returned with object query results, for all object types.
    * ------------------------------------------------------------------------  */
   db2Uint64                  createTime;
} db2ACS_QueryOutput;
```

objectID

データ・タイプ: db2ACS_ObjectID。

db2ACS_ObjectID は、各保管オブジェクトごとのユニーク ID であり、照会によってストレージ・リポジトリーに戻されます。 db2ACS_ObjectID は、単一の DB2 ACS セッションの時間フレーム内でのみ、ユニークで永続的であることが保証されています。

object データ・タイプ: db2ACS_ObjectInfo

db2ACS_ObjectInfo には、DB2 拡張コピー・サービス (ACS) API を使用して作成したオブジェクトに関する情報が入ります。

postInfo

データ・タイプ: db2ACS_PostObjectInfo。

db2ACS_DB2ID は、スナップショットのバックアップ・オブジェクトの作成時には認識されないものの、オブジェクト・リポジトリーで保守されなければならないデータのセットです。

db2ID データ・タイプ: db2ACS_DB2ID。

db2ACS_DB2ID は IBM Data Server を識別します。

status データ・タイプ: db2ACS_ObjectStatus。

db2ACS_ObjectStatus には、スナップショットのバックアップ操作の状況または進行についての情報、またはスナップショットのバックアップ・オブジェクトの状況またはユーザビリティーについての情報が含まれています。

objectSize

データ・タイプ: db2Uint64。

バイト単位でのオブジェクトのサイズ。

metaDataSize

データ・タイプ: db2Uint64。

バイト単位でのオブジェクトに関連したメタデータのサイズ (存在する場合)。

createTime

データ・タイプ: db2Uint64。

オブジェクトの作成時間。**createTime** の値は、ANSI C の time_t 値と等しくなります。

db2ACS_ReadList DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_ReadList には、グループのリストが含まれています。

```
/* The ReadList will only be used for snapshots where the action is READ, and
 * where one of the granularity modifiers other than BY_OBJ has been specified.
 * In the typical usage scenario of ( READ | BY_OBJ ) the ReadList parameter
 * should be ignored.
 *
 * When the action is DB2ACS_ACTION_BY_GROUP the union is to be interpreted
 * as a group list.
 * ------------------------------------------------------------------------ */
typedef union db2ACS_ReadList
{
   db2ACS_GroupList           group;
} db2ACS_ReadList;
```

group データ・タイプ: db2ACS_GroupList。

db2ACS_GroupList には、スナップショットのバックアップ操作に組み込まれているグループのリストが含まれています。

db2ACS_ReturnCode DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_ReturnCode には、ストレージ・ハードウェアに固有のメッセージ・テキストおよびエラー・コードを含んでいる診断情報が入ります。 DB2 ACS API 関数呼び出しの db2ACS_ReturnCode パラメーターの内容は、データベース・マネージャーの診断ログに記録されます。

```
/* ************************************************************************
 * Storage Adapter Return Code and Diagnostic Data.
 *
 * These will be recorded in the DB2 diagnostic logs, but are intended to be
 * internal return and reason codes from the storage layers which can be used
 * in conjunction with the DB2ACS_RC to provide more detailed diagnostic info.
 * ************************************************************************ */
typedef struct db2ACS_ReturnCode
{
   int                          returnCode;
   int                          reasonCode;
   char                         description[DB2ACS_MAX_COMMENT_SZ + 1];
} db2ACS_ReturnCode;
```

returnCode

データ・タイプ: int

ストレージ・ハードウェアに固有の戻りコード。

reasonCode

データ・タイプ: int

ストレージ・ハードウェアに固有の理由コード。

description

データ・タイプ: char[]

長さが DB2ACS_MAX_COMMENT_SZ + 1 の文字ストリング。

db2ACS_SessionInfo DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_SessionInfo には、DB2 ACS セッションに関するすべての情報が入ります。

```
/* ************************************************************************
 * Session Info
 * ************************************************************************ */
typedef struct db2ACS_SessionInfo
{
   db2ACS_DB2ID                 db2ID;

   /* Fields identifying the backup session originator.
    * ------------------------------------------------------------------- */
   SQL_PDB_NODE_TYPE            dbPartitionNum;
   char                         db[SQL_DBNAME_SZ + 1];
   char                         instance[DB2ACS_MAX_OWNER_SZ + 1];
   char                         host[SQL_HOSTNAME_SZ + 1];
   char                         user[DB2ACS_MAX_OWNER_SZ + 1];
   char                         password[DB2ACS_MAX_PASSWORD_SZ + 1];

   /* The fully qualified ACS vendor library name to be used.
    * ------------------------------------------------------------------- */
   char                         libraryName[DB2ACS_MAX_PATH_SZ + 1];
} db2ACS_SessionInfo;
```

db2ID データ・タイプ: db2ACS_DB2ID

db2ACS_DB2ID は IBM Data Server を識別します。

dbPartitionNum
データ・タイプ: SQL_PDB_NODE_TYPE

データベース・パーティションに固有の数値 ID。

db データ・タイプ: char[]

長さが SQL_DBNAME_SZ + 1 の文字ストリング。

instance
データ・タイプ: char[]

長さが DB2ACS_MAX_OWNER_SZ + 1 の文字ストリング。

host データ・タイプ: char[]

長さが SQL_HOSTNAME_SZ + 1 の文字ストリング。

user データ・タイプ: char[]

長さが DB2ACS_MAX_OWNER_SZ + 1 の文字ストリング。

password
データ・タイプ: char[]

長さが DB2ACS_MAX_PASSWORD_SZ + 1 の文字ストリング。

libraryName
データ・タイプ: char[]

長さが DB2ACS_MAX_PATH_SZ + 1 の文字ストリング。

db2ACS_SnapshotDetails DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_SnapshotDetails には、スナップショット・バックアップ操作に関する情報が入ります。

```
typedef struct db2ACS_SnapshotDetails
{
   char                      imageTimestamp[SQLU_TIME_STAMP_LEN + 1];
} db2ACS_SnapshotDetails;
```

imageTimestamp
データ・タイプ: char[]

長さが SQLU_TIME_STAMP_LEN + 1 の文字ストリング。

db2ACS_MetaData DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_MetaData は、スナップショットのバックアップ・メタデータを保管します。

```
/* ****************************************************************************
 * The metadata structure itself is internal to DB2 and is to be treated by
 * the storage interface as an unstructured block of data of the given size.
 * **************************************************************************** */
typedef struct db2ACS_MetaData
{
   db2Uint64                 size;
   void                    * data;
} db2ACS_MetaData;
```

size データ・タイプ: db2Uint32。

バイト単位での **data** のサイズ。

data データ・タイプ: void *。

データベース・マネージャーがスナップショットのバックアップ・メタデータを保管するのに使用するメモリーのブロックへのポインター。

db2ACS_VendorInfo DB2 拡張コピー・サービス (ACS) API のデータ構造

db2ACS_VendorInfo には、DB2 ACS API ドライバーに関する情報が入ります。

```
/* ============================================================================
 * Storage Vendor Identifier
 * ============================================================================ */
typedef struct db2ACS_VendorInfo
{
   void                    * vendorCB;             /* Vendor control block  */
   db2Uint32                 version;              /* Current version       */
   db2Uint32                 release;              /* Current release       */
   db2Uint32                 level;                /* Current level         */
   char                      signature[DB2ACS_MAX_VENDORID_SZ + 1];
} db2ACS_VendorInfo;
```

vendorCB

データ・タイプ: void *

DB2 ACS API ドライバーに固有の制御ブロックを示すポインター。

version

データ・タイプ: db2Uint32

DB2 ACS API ドライバーのバージョン。

release データ・タイプ: db2Uint32

DB2 ACS API ドライバーのリリース・レベル。

level データ・タイプ: db2Uint32

DB2 ACS API ドライバーのレベル ID。

signature

データ・タイプ: db2ACS_VendorSignature

長さが DB2ACS_MAX_VENDORID_SZ + 1 の文字ストリング。

DB2 拡張コピー・サービス (ACS) API の戻りコード

DB2 拡張コピー・サービス (ACS) API 関数は、可能性がある戻りコードの定義済みセットを戻します。

表 31. DB2 拡張コピー・サービス (ACS) API の戻りコード

戻りコード	説明
DB2ACS_RC_OK	操作は正常に終了しました。
DB2ACS_RC_LINK_EXIST	セッションはすでにアクティブにされています。
DB2ACS_RC_COMM_ERROR	磁気テープ・ドライブなどのストレージ・デバイスに通信エラーがあります。

表 31. DB2 拡張コピー・サービス (ACS) API の戻りコード (続き)

戻りコード	説明
DB2ACS_RC_INV_VERSION	データベース・マネージャーの DB2 ACS ライブラリーのバージョンと、 DB2 ACS API ドライバーのバージョンに互換性がありません。
DB2ACS_RC_INV_ACTION	データベース・マネージャーが、DB2 ACS API ドライバーからの無効なアクションを要求しました。
DB2ACS_RC_NO_DEV_AVAIL	現在、使用可能な磁気テープ・ドライブなどのストレージ・デバイスがありません。
DB2ACS_RC_OBJ_NOT_FOUND	DB2 ACS API ドライバーが、データベース・マネージャーによって指定されたスナップショット・バックアップ・オブジェクトを検出できませんでした。
DB2ACS_RC_OBJS_FOUND	DB2 ACS API ドライバーが、データベース・マネージャーによる指定と一致する複数のスナップショット・バックアップ・オブジェクトを検出しました。
DB2ACS_RC_INV_USERID	データベース・マネージャーが、DB2 ACS API ドライバーに無効なユーザー ID を渡しました。
DB2ACS_RC_INV_PASSWORD	データベース・マネージャーが、DB2 ACS API ドライバーに無効なパスワードを渡しました。
DB2ACS_RC_INV_OPTIONS	データベース・マネージャーが無効なオプションを指定しました。
DB2ACS_RC_INIT_FAILED	データベース・マネージャーが DB2 ACS セッションの初期化を試みましたが、初期化が失敗しました。
DB2ACS_RC_INV_DEV_HANDLE	データベース・マネージャーがストレージ・デバイスの無効なハンドルを渡しました。
DB2ACS_RC_BUFF_SIZE	データベース・マネージャーが無効なバッファー・サイズを指定しました。
DB2ACS_RC_END_OF_DATA	DB2 ACS API ドライバーが、これ以上のスナップショット・バックアップ・オブジェクトを検出できません。
DB2ACS_RC_END_OF_TAPE	ストレージ・デバイスが、テープ・バックアップ・メディアの終わりに予期せずに到達しました。
DB2ACS_RC_DATA_RESEND	磁気テープ・ドライブなどのストレージ・デバイスが、データベース・マネージャーによるデータの最新のバッファーの再送を要求しました。
DB2ACS_RC_COMMIT_FAILED	DB2 ACS API ドライバーがトランザクションをコミットできませんでした。
DB2ACS_RC_DEV_ERROR	磁気テープ・ドライブなどのストレージ・デバイスにエラーがあります。
DB2ACS_RC_WARNING	ストレージ・ハードウェアが警告を戻しました。詳しくは、データベース・マネージャーの診断ログを参照してください。
DB2ACS_RC_LINK_NOT_EXIST	セッションが以前にアクティブにされていません。
DB2ACS_RC_MORE_DATA	ストレージ・ロケーションからデータベース・マネージャーへ転送するデータがまだあります。
DB2ACS_RC_ENDOFMEDIA_NO_DATA	ストレージ・デバイスが、データを検出できないまま、ストレージ・メディアの終わりに到達しました。

表 31. DB2 拡張コピー・サービス (ACS) API の戻りコード (続き)

戻りコード	説明
DB2ACS_RC_ENDOFMEDIA	ストレージ・デバイスが、ストレージ・メディアの終わりに到達しました。
DB2ACS_RC_MAX_LINK_GRANT	最大数のリンクが設定されました。データベース・マネージャーは、これ以上のリンクを設定できません。
DB2ACS_RC_IO_ERROR	DB2 ACS API ドライバーが、入力操作または出力操作の結果、エラーを検出しました。
DB2ACS_RC_DELETE_FAILED	DB2 ACS API ドライバーが、データベース・マネージャーによって指定されたスナップショット・バックアップ・オブジェクトの削除に失敗しました。
DB2ACS_RC_INV_BKUP_FNAME	データベース・マネージャーが、スナップショット・バックアップ・オブジェクトの無効なファイル名を指定しました。
DB2ACS_RC_NOT_ENOUGH_SPACE	DB2 ACS API ドライバーが、データベース・マネージャーによって指定されたデータベースのスナップショット・バックアップを実行するために十分なストレージ・スペースがないと見積もりました。
DB2ACS_RC_ABORT_FAILED	データベース・マネージャーが DB2 ACS 操作のアボートを試みましたが、アボートの試みが失敗しました。
DB2ACS_RC_UNEXPECTED_ERROR	DB2 ACS API ドライバーが重大な不明エラーを検出しました。
DB2ACS_RC_NO_DATA	DB2 ACS API ドライバーが、データベース・マネージャーにデータを戻しませんでした。
DB2ACS_RC_OBJ_OUT_OF_SCOPE	データベース・マネージャーが、DB2 ACS API ドライバーによって管理されていないリカバリー・オブジェクトに、DB2 ACS 操作の実行を試みました。
DB2ACS_RC_INV_CALL_SEQUENCE	データベース・マネージャーが、DB2 ACS API 関数を無効な順序で呼び出しました。例えば、データベース・マネージャーは db2ACSInitialize を呼び出してからでないと、db2ACSQueryAPIVersion を除く他の DB2 ACS API 関数を呼び出せません。
DB2ACS_RC_SHARED_STORAGE_GROUP	データベース・マネージャーが、他のデータベースまたはアプリケーションによって使用中であるストレージ・オブジェクトに対してスナップショット操作の実行を試みました。

DB2 拡張コピー・サービス (ACS) をサポートするオペレーティング・システムおよびハードウェア

IBM Data Server には、IBM Data Server がサポートするオペレーティング・システムのサブセットとハードウェアをサポートする DB2 ACS API ドライバーが統合されています。

表 32. DB2 拡張コピー・サービス (ACS) API をサポートするオペレーティング・システムおよびハードウェア

ハードウェア	**Storage Area Network (SAN) のストレージを持つ AIX オペレーティング・システム**	**ネットワーク・ファイル・システム (NFS) のストレージを持つ AIX オペレーティング・システム**	**ネットワーク・ファイル・システム (NFS) のストレージ[1]を持つ Linux オペレーティング・システム**
IBM TotalStorage SAN ボリューム・コントローラー	完全サポート。	サポートなし。	サポートなし。
IBM System Storage DS6000	以下を除いて、完全サポート。 • 増分コピーはサポートされていません。	サポートなし。	サポートなし。
IBM System Storage DS8000	以下を除いて、完全サポート。 • 増分コピーはサポートされていません。	サポートなし。	サポートなし。
IBM System Storage N シリーズ	完全サポート。	完全サポート。	完全サポート。
NetApp V-series	完全サポート。	完全サポート。	完全サポート。
NetApp FAS series	完全サポート。	完全サポート。	完全サポート。

[1]以下のシステムのみが DB2 ACS および Linux でサポートされています。

- x86 (Intel Pentium、 Intel Xeon、および AMD) プロセッサー上の 64 ビットのみ
- POWER® (Linux をサポートする System z、System i または pSeries システム)

DB2 ACS のハードウェア要件およびソフトウェア要件の完全なリストについては、次の技術情報を参照してください。https://www.ibm.com/support/docview.wss?uid=swg21321830

第 17 章 DB2 pureScale環境でのデータ・リカバリー

DB2 pureScale環境でのバックアップおよびリストア操作

DB2 pureScale環境では、いずれかのメンバー上で **BACKUP DATABASE** または **RESTORE DATABASE** コマンドを 1 回実行するだけで、すべてのメンバーのバックアップまたはリストア操作が開始されます。

DB2 pureScale環境では、データベース・パーティションを 1 つだけ持つことができるため、バックアップ操作が処理するデータ集合は 1 つのみであり、生成されるバックアップ・イメージはグループ全体で 1 つのみとなります。他のメンバーについては、処理が必要なものは、データベースのメタデータおよびトランザクション・ログのみであり、これらも単一のバックアップ・イメージに含められます。

バックアップ・イメージには、指定された表スペースのデータ、必要なメタデータ、および現在定義されている全メンバーの構成情報が含まれます。DB2 pureScale インスタンスの他のメンバー上で、追加のバックアップ操作を実行する必要はありません。また、データベースおよびメンバー固有のメタデータを全メンバーに対してリストアする場合も、 **RESTORE DATABASE** コマンドを 1 回実行するだけで済みます。クラスターをリストアするために、他のメンバー上で追加のリストア操作を実行する必要はありません。連続バックアップ・イメージのタイム・スタンプは、どのメンバーが生成したかにかかわらず、固有であり、増加していく値です。

オフライン・バックアップ操作を試行するには、全メンバーが整合状態でなければなりません。一度に実行できるオフライン・バックアップ操作は 1 つだけです。このバックアップ・ユーティリティーは、データベースに対する超排他アクセスを、すべてのメンバーにおいて取得するためです。並行オンライン・バックアップ操作はサポートされていますが、別々のバックアップ操作で同じ表スペースを同時にコピーすることはできません。順番を待つ必要があります。

データベースからデータおよびメタデータを読み取る操作のすべてと、バックアップ・イメージに書き込む操作のすべては、単一のメンバー上で行われます。バックアップまたはリストア操作がその他のメンバーと交わす相互作用は、データベース・メタデータ (表スペース定義、ログ・ファイル・ヘッダー、データベース構成など) のコピーまたは更新に限られます。

オンライン・バックアップ操作の実行中に、別のメンバーがオフラインであったり、オフラインになったり、またはオンラインに戻ったりしても、操作は正常に実行されます (524 ページの表 33)。データベースのリストア操作は、他のメンバーの状態による影響を受けませんが、バックアップ操作は、場合によっては、オフラインで不整合状態にあるメンバー上でメンバー・クラッシュ・リカバリーが完了するまで、短時間待機しなければならないことがあります。

表 33. データベースのバックアップおよびリストア操作に対して DB2 pureScaleインスタンスの他のメンバーの状態が与える影響

操作	他のメンバーの状態	
	オフラインおよび整合	**オフラインおよび不整合**
オンライン・バックアップ	バックアップ操作は正常終了します。バックアップ・ユーティリティーが、バックアップ操作の開始直後にログ・ファイル・ヘッダー (LFH) にアクセスしている間、または、バックアップ操作の終了間近にログ・ストリームにアクセスしている間、他のメンバーはアクティブになれません。	バックアップ操作は正常終了しますが、メンバー・クラッシュ・リカバリーが完了して、他のメンバーがアクティブまたは整合状態のいずれかになるまで待機する必要があります。バックアップ・ユーティリティーが、バックアップ操作の開始直後に LFH にアクセスしている間、または、バックアップ操作の終了間近にログ・ストリームにアクセスしている間、他のメンバーはアクティブになれません。
リストア	リストア操作は正常に完了します。	リストア操作は正常に完了します。

イメージおよびアーカイブの名前

ディスク上に作成するバックアップ・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*.DBPART*nnn*.*Timestamp*.*Seq_num*

DB_alias
バックアップ・ユーティリティーの呼び出し時に指定したデータベース別名。

タイプ バックアップ操作のタイプ。 0 はデータベースのフル・バックアップ、 3 は表スペースのバックアップ、 4 は **COPY** NO オプションを指定した **LOAD** コマンドによって生成されたバックアップ・イメージを表します。

Inst_name
DB2INSTANCE 環境変数の値である、現在のインスタンスの名前。

nnn データベース・パーティション番号。 DB2 pureScale環境では、この番号は常に 000 です。

Timestamp
バックアップ操作を実行した日付と時刻を 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
- *mm* は月 (01 から 12)
- *dd* は日 (01 から 31)
- *hh* は時 (00 から 23)
- *nn* は分 (00 から 59)

- *ss* は秒 (00 から 59)

Seq_num
ファイル拡張子として使用する 3 桁の番号。

例えば、以下のようにします。

```
SAMPLE.0.krodger.DBPART000.200802241234.001
```

INCLUDE LOGS を指定したオンライン・バックアップ

INCLUDE LOGS オプションを指定した (デフォルト) オンライン・バックアップ操作では、データベースをリストアして最小リカバリー時間までロールフォワードするのに必要な範囲のログ・ファイルを含む、バックアップ・イメージが生成されます。このバックアップ・イメージを使用して、 (災害復旧時などに) 新規データベースにリストアした場合に、後続のロールフォワード操作で使用できるログが、このバックアップ・イメージに含まれているログしかないと、多くの場合、**ROLLFORWARD...TO END OF LOGS** コマンドは、ログ・ファイルの欠落を示すエラー・メッセージ (SQL1273N) を返します。このような状況が予期されるシチュエーションにはいくつかありますが、原因は、バックアップ操作後にさらにログが書き込まれていることを、データベース・マネージャーが検出したにもかかわらず、それらのログを現行のロールフォワード操作で使用できないためです。また、整合性が取れた時点までデータベースをロールフォワードするのに必要なログが 1 つ以上欠落している場合も、このような状況が予期されます。いずれの場合も、ロールフォワード操作のエンドポイントが許容可能であることを確認してから、**ROLLFORWARD...AND STOP** コマンドを実行します。ロールフォワード操作が、ログ・ファイルが欠落しているにもかかわらず、最小リカバリー時間に到達すれば、**ROLLFORWARD...AND STOP** コマンドは正常に終了します。そうでない場合は、SQL1276N (ロールフォワード操作は、このバックアップ・イメージを使用して最小リカバリー時間に到達できなかった) が返されます。

DB2 pureScale環境でのログ・シッピングによる災害復旧および高可用性

*ログ・シッピング*とは、すべてのログ・ファイルをスタンバイ・マシンにコピーする処理のことです。これは、1 次データベースにおいて、アーカイブ・ログを保持しているストレージ装置から直接コピー、または、アーカイブ・ログをコピーするユーザー出口プログラムを利用して行います。ログのアーカイブ時にそれらをスタンバイ・データベースに適用することによって、スタンバイ・データベースを最新の状態に維持することができます。または、スタンバイ・サイトにデータベースまたは表スペースのバックアップ・イメージとログのアーカイブを保管しておき、災害が発生した場合にのみ、リストアおよびロールフォワード操作を実行することもできます。いずれの場合も、スタンバイ・サイト上のロールフォワード操作で、1 つ以上のログ・ファイルの欠落が検出されて SQL1273N が返される可能性があります。ロールフォワード操作が許容可能なタイム・スタンプに到達していることを確認してください。または、問題を解決するために適切な処置を行ってください。

ログ・ストリーム・マージ操作中に、DB2 データベース・マネージャーが、いずれかのログ・ストリームに損失ログ・ファイルがあることを検出すると、エラーが返されます。ロールフォワード・ユーティリティーは SQL1273N を返し、db2ReadLog API は SQL2657N を返します。ログのアーカイブ時にそれらをスタンバイ・データ

ベースに適用して、スタンバイ・データベースを最新の状態に維持することを選択した場合は、ロールフォワード操作で、頻繁にログの欠落を検出する可能性があります。

図 30 は、2 つのメンバーがアクティブ・ログ・ストリーム内のログ・ファイルにログ・レコードを書き込む方法の例を示します。それぞれのログ・ファイルはボックスで表現されています。高可用性を目的として、プライマリー・サイトとスタンバイ・サイトの両方がセットアップされているシナリオについて考察します。**END OF LOGS** オプションを指定した **ROLLFORWARD** コマンドを、スタンバイ・サイトに対して、時刻 A、B、および C に試行します。任意の時点において、その時点より前にクローズされているログ・ファイルはすべてアーカイブ済みで、スタンバイ上でアクセス可能です。そうでない場合、そのログ・ファイルはまだプライマリー上でアクティブなため、スタンバイからは使用できません (ログ・ストリーム 1 上の、時刻 B における、ログ・ファイル 4 を参照)。

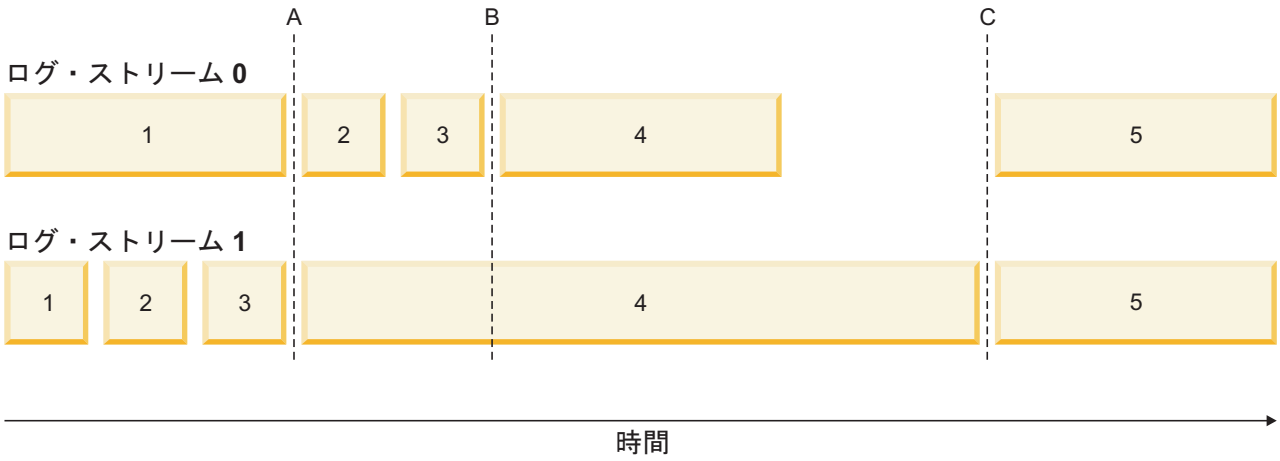


図 30. DB2 pureScale環境でのログ・ファイル

時刻 A では、ログ・ストリーム 0 のログ・ファイル 1 が、ログ・ストリーム 1 のログ・ファイル 3 と同時にクローズおよびアーカイブされているため、**ROLLFORWARD** コマンドが正常に完了します。しかし、時刻 B では、**ROLLFORWARD** コマンドが SQL1273N を戻します。これは、コマンドがスタンバイ・サイトに対して発行されるときに、スタンバイ・サイトから、ログ・ストリーム 0 のログ・ファイル 2 および 3 にはアクセスできますが、ログ・ストリーム 1 のログ・ファイル 4 には (このログ・ファイルがプライマリー・サイトでまだオープンされていて、アクティブであるため) アクセスできないためです。さらに、ログ・ストリーム 0 のファイル 2 および 3 のログ・レコードが、ログ・ストリーム 1 のログ・ファイル 4 の始めの部分と同じ時間枠の中で書き込まれているため、ロールフォワード操作は、ログ・ストリーム 1 のログ・ファイル 4 が使用可能になるまで、ログ・ファイル 2 および 3 を処理できません。時刻 C では、ログ・ストリーム 1 でログ・ファイル 4 がクローズおよびアーカイブされているので、**ROLLFORWARD** コマンドが正常に完了します。ARCHIVE LOG コマンドを使用したり、すべてのメンバーにわたってデータベースを非アクティブ化することにより、すべてのログ・ストリームにわたり、ファイルの切り捨てとアーカイブを強制することができます。**ARCHIVE LOG** コマンドの場合、各ログ・ストリーム上の現行ログ・ファイルが独立して切り捨てられるため、すべてのメンバーにおいて、完全に同時に切り捨てが行われる保

証はありません。従って、**ARCHIVE LOG** コマンドを発行したとしても、**ROLLFORWARD** コマンドの実行時に SQL1273N エラーが出る可能性があります。

DB2 pureScale環境でログ・シッピングを使用している場合、ログの欠落はよく起こり、予期される状態ですが、たいていは、スタンバイに対してロールフォワード操作を実行するごとに、(SQL1273 が戻されても) 前回の **ROLLFORWARD** コマンド実行時よりも改善されるため、多くの場合、このエラーは予期しておくべきでしょう。ただし、プライマリー・サイトで、他のログ・ストリームのログのアーカイブには成功したが、あるログ・ストリームのファイルのアーカイブに問題が発生したという場合も考えられます。これは、あるログ・ストリーム用のアーカイブ・ストレージへのアクセスに、一時的な問題が発生したことが原因である可能性があります。このような問題は、スタンバイ側でのログのマージおよび再生を遅延させ、災害時に損失するおそれのあるトランザクション数を増加させる可能性があります。スタンバイ・システムが最新状態であることを確認するには、ロールフォワード操作が SQL1273N を返すたびに、 **ROLLFORWARD...QUERY STATUS** コマンドを実行して、時間の経過とともに処理が進行していることを確認します。スタンバイ上のロールフォワード操作が長期間進行していない場合は、欠落しているとして報告されたログ・ファイルがスタンバイ・システム上で使用できない理由を調べて、問題を解決します。 **ARCHIVE LOG** コマンドを使用すると、各メンバー上で現在更新されているログ・ファイルを切り捨てることができます。これによって、それらのログ・ファイルはアーカイブされ、スタンバイ・システム上で再生できるようになります。

災害 (火事、地震、破壊行為、または他の壊滅的状況など) 発生時の復旧計画では、通常、残っているログをすべて使用してロールフォワード操作を実行するか、または、リストアしてから、使用可能なログをすべて使用してロールフォワード操作を実行します。前述のように、プライマリー上でログ・ファイルが書き込まれたのに、災害時点でアーカイブされていなければ、ロールフォワード操作で、 1 つ以上のログ・ファイルの欠落が検出される可能性があります (SQL1273N)。また、何らかの予期しない理由によって、アーカイブされたログをロールフォワード・ユーティリティーが検出できないという場合も考えられます。この場合も、ロールフォワード・ユーティリティーは SQL1273N を返す可能性があります。
ROLLFORWARD...QUERY STATUS コマンドを使用して、ロールフォワード操作のエンドポイントを確認し、ログの欠落状況が予期されるかどうかを調べることが重要です。ログの欠落状況が予期される場合、またはエンドポイントが許容可能である場合は、 **ROLLFORWARD...STOP** コマンドを実行して、ロールフォワード・リカバリー・プロセスを完了させることができます。

制約事項

DB2 pureScale Feature インストール済み環境と非 DB2 pureScale Feature インストール済み環境との間でのバックアップおよびリストア操作はサポートされません。

メンバーの追加またはドロップを含むトポロジー変更の後に、そのトポロジー変更の発生時点をまたぐロールフォワード・リカバリー操作は実行できません。メンバーを追加またはドロップすると、データベースはバックアップ・ペンディング状態になるため、データベースのフル・バックアップ操作を実行しないと、このデータベースには接続できません。リカバリーする場合は、このバックアップ・イメージをリストアし、ログの最後までロールフォワードします。トポロジー変更前に取得したバックアップ・イメージをリストアする必要がある場合は、トポロジー変更の

発生時点までしかロールフォワードできません。これを実行するには、**ROLLFORWARD...TO END OF LOGS** コマンドを実行し (SQL1546N が返されます)、その次に **ROLLFORWARD DATABASE...STOP** コマンドを実行します。この操作では、トポロジー変更後にデータベースを変更したトランザクションについてはリカバリーされません。

DB2 pureScale環境では、 **BACKUP DATABASE** コマンドの **ON ALL DBPARTITIONNUMS** パラメーターおよび **ON DBPARTITION (0)** パラメーターが有効です。ただし、0 以外のデータベース・パーティション番号を指定すると、データベース・パーティションは他に存在しないため、エラー (SQL0270N) が返されます。

このリリースには、次の制約事項があります。

- DB2 pureScale環境の外部にあるデータベースを、DB2 pureScale環境にマイグレーションすることができます。そうしたデータベースを DB2 pureScale環境にマイグレーションするために、データベース・リストア操作を使用することはできません。
- DB2 拡張コピー・サービス (ACS) を使用したスナップショットのバックアップ操作はサポートされていません。

例

- 4 メンバーの SAMPLE という名前のデータベースを、いずれかのメンバーからバックアップする

  ```
  BACKUP DB SAMPLE
  ```

- 1 メンバーの SAMPLE という名前のデータベースをリストアする

  ```
  RESTORE DB SAMPLE
  ```

- **RECOVER DATABASE** コマンドを使用して、 SAMPLE という名前のデータベースを、いずれかのメンバーからリストアしてロールフォワードする

  ```
  RECOVER DB SAMPLE TO END OF LOGS
  ```

  データベースが存在しない場合は、**RECOVER DATABASE** コマンドではなく、**RESTORE DATABASE** と **ROLLFORWARD DATABASE** コマンドを使用します。 **RECOVER DATABASE** コマンドは、完全なデータベース履歴を持つ既存のデータベースでないと、正常に実行されないからです。

ロールフォワード

DB2 pureScale環境におけるログ・ストリーム・マージおよびログ・ファイル管理

DB2 pureScale環境では、各メンバーが、独自のトランザクション・ログ・ファイルのセット (つまりログ・ストリーム) を共有ディスク上に持っていて、セットごとに異なるログ・パスに入っています。メンバーのログ・ファイルには、そのメンバー上で行われたすべてのデータ変更の履歴が入っています。

同時に複数のアプリケーションが、それぞれ異なるメンバーにアクセスすると、実行中に依存トランザクションが生成される可能性があります。例えば、2 つのトランザクションが同じ行を変更した場合、その両トランザクションの間に依存関係が生じる可能性があります。ログ・レコードの解釈を正しく行うために、DB2 デー

タ・サーバーは、全ログ・ストリームのレコードを調べ、実行時に発生した更新の順序を反映するように、レコードを順序付けする必要があります。この順序付けを、*ログ・ストリーム・マージ* 操作と呼びます。 DB2 pureScale 環境では、ログ・ストリーム・マージが必要となる操作タイプがいくつかあります。代表的なものには、グループ・クラッシュ・リカバリー、データベースのロールフォワード操作、表スペースのロールフォワード操作などがあります。

DB2 pureScale環境のロギング構成パラメーター

表 34 に、ロギング関連のデータベース構成パラメーターについて、どのパラメーターの有効範囲がグローバルであるか、また、どのパラメーターが動的に更新可能であるかを示します。

表 34. ロギング関連のデータベース構成パラメーター

パラメーター	グローバルである	動的に更新可能である
archretrydelay	はい	はい
blk_log_dsk_ful	いいえ	はい
failarchpath	はい	はい
logarchcompr1	はい	はい
logarchcompr2	はい	はい
logarchmeth1	はい	はい
logarchmeth2	はい	はい
logarchopt1	はい	はい
logarchopt2	はい	はい
logbufsz	いいえ	はい
logfilsiz	はい	いいえ
logprimary	はい	いいえ
logsecond	はい	はい
max_log	いいえ	はい
mirrorlogpath [1]	はい	いいえ
newlogpath [1]	はい	いいえ
num_log_span	いいえ	はい
numarchretry	はい	はい
overflowlogpath	はい	はい
softmax	はい	いいえ
vendoropt	はい	はい

[1] 最初にデータベースに接続した、またはデータベースをアクティブ化したメンバーが、このログ・パス・パラメーターに対する変更を処理します。DB2 データベース・マネージャーは、パスが存在することと、そのパスに対する読み取りおよび書き込み権限の両方があることを確認します。また、メンバー固有のログ・ファイル用サブディレクトリーを作成します。これらの操作のいずれかに失敗すると、DB2 データベース・マネージャーは、指定されたパスを拒否し、古いパスを使用してデータベースをオンラインにします。指定されたパスをデータベース・マネージャーが受け入れた場合は、その新しい値が各メンバーに伝搬されます。新しいパスへの切り替えの試行中に、いずれかのメンバーに障害が起こった場合、それ以降、そのデータベースに対してアクティブ化または接続を試行しても、失敗して SQL5099N が返されます。すべてのメンバーが同じログ・パスを使用する必要があります。

DB2 pureScale環境でのログ・ストリーム・マージ操作のためのログのリトリーブ

リトリーブしたログ・ファイル用のパス内に、サブディレクトリーが作成されます。サブディレクトリーは *log_path*/LOGSTREAM*xxxx* という形式になります。ここで *log_path* は、ログ・パス、オーバーフロー・ログ・パス、またはミラー・ログ・パスを表し、*xxxx* は、4 桁のログ・ストリーム ID を表しています (ログ・ストリーム ID は、必ずしも、関連するメンバーの ID と同じではありません)。メンバーがログのリトリーブを必要とする場合、 DB2 データベース・マネージャーは、このサブディレクトリー内に、各メンバーからリトリーブしたログを入れるためのサブディレクトリーを、もう 1 階層作成します。例えば、3 メンバーのシステムで `/home/dbuser/overflow/` をオーバーフロー・ログ・パスに指定している場合に、メンバー 0 上のアプリケーションが他のメンバー所有のログをリトリーブする必要があるときには、メンバー 0 用のパスは `/home/dbuser/overflow/NODE0000/LOGSTREAM0000` であり、以下の例に示すように、このパスの下のサブディレクトリーに他のメンバー所有のログをリトリーブしたものが入ります。

```
メンバー 0 が、独自のログをリトリーブする場所
   /home/dbuser/overflow/NODE0000/LOGSTREAM0000/LOGSTREAM0000
メンバー 0 が、メンバー 1 に属するログをリトリーブする場所
   /home/dbuser/overflow/NODE0000/LOGSTREAM0000/LOGSTREAM0001
メンバー 0 が、メンバー 2 に属するログをリトリーブする場所
   /home/dbuser/overflow/NODE0000/LOGSTREAM0000/LOGSTREAM0002
```

注: これらのリトリーブ・サブディレクトリーには、手動でログ・ファイルを入れないでください。手動でログ・ファイルをリトリーブする必要がある場合は、代わりにオーバーフロー・ログ・パスを使用してください。

他のメンバー所有のアーカイブ・ログ・ファイルを読み取る場合、メンバーは、通常、独自のログ・パスまたはオーバーフロー・ログ・パスに、ログ・ファイルをリトリーブする必要があります。この場合、ログ・ストリーム・マージ操作では、必要に応じて、ログ・ストリームごとに 1 つの **db2logmgr** エンジン・ディスパッチ可能単位 (EDU) を作成します。

前述のように、他のメンバー所有のログ・ファイルの保管に使用できるパスには、次のリストに示す 3 つのパスがあります。

1. **overflowlogpath** データベース構成パラメーターを設定している場合は、そのオーバーフロー・ログ・パスが使用されます。

   ヒント: **ROLLFORWARD DATABASE** および **RECOVER DATABASE** コマンドのオプションを使用して、代替オーバーフロー・ログ・パスを指定できます。これらのオプションの値は、そのリカバリー操作のためだけに、データベース構成をオーバーライドします。
2. 1 次ログ・パス
3. **mirrorlogpath** データベース構成パラメーターを設定している場合は、そのミラー・ログ・パスが使用されます。

DB2 データベース・マネージャーは、このリスト内の最初のパスにログ・ファイルを保管できない場合、その次のパスを使用しようとします。これらのパスに使用できるものがない場合、ログ・ストリーム・マージ操作を呼び出したユーティリティーは、そのユーティリティーに固有のエラーを返します。

DB2 pureScale環境の **GET DATABASE CONFIGURATION** コマンドの出力では、各ログ・パスは、メンバーの名前が後に付けられて示されます。例えば、ミラー・ログ・パスが /home/dbuser/mirrorpath/ に設定されている場合、メンバー 2 での出力には /home/dbuser/mirrorpath/NODE0000/LOGSTREAM0002 と表示されます。

他のメンバー所有のログ・ファイルを手動でリトリーブする必要がある場合は、必ず、データベース・マネージャーが、自動的に作成される場合と同じディレクトリー構造を使用して、それらのログ・ファイルにアクセスできるようにしてください。例えば、メンバー 1 のオーバーフロー・ログ・パス内で、メンバー 2 のログを使用できるようにするには、それらのログを /home/dbuser/overflow/NODE0000/LOGSTREAM0001/LOGSTREAM0002 ディレクトリーに配置します。

リトリーブされたログ・ファイルは、必要がなくなった時に自動的に削除されます。ログ・ストリーム・マージ操作時に作成されたサブディレクトリーは、今後の使用に備えて保持されます。

ログ・ストリーム・マージ操作中の損失ログの検出

リカバリー操作に必要なログ・ファイルを誤って削除、移動、またはアーカイブして失った場合、その失われたログ・ファイルより前の最後の整合ポイントまでは、データベースをロールフォワード・リカバリーできます。

ログ・ストリーム・マージ操作中に、DB2 データベース・マネージャーが、いずれかのログ・ストリームに損失ログ・ファイルがあることを検出すると、エラーが返されます。ロールフォワード・ユーティリティーは SQL1273N を返し、db2ReadLog API は SQL2657N を返します。

図 31 は、2 つのメンバーがアクティブ・ログ・ストリーム内のログ・ファイルにログ・レコードを書き込む方法の例を示します。それぞれのログ・ファイルはボックスで表現されています。

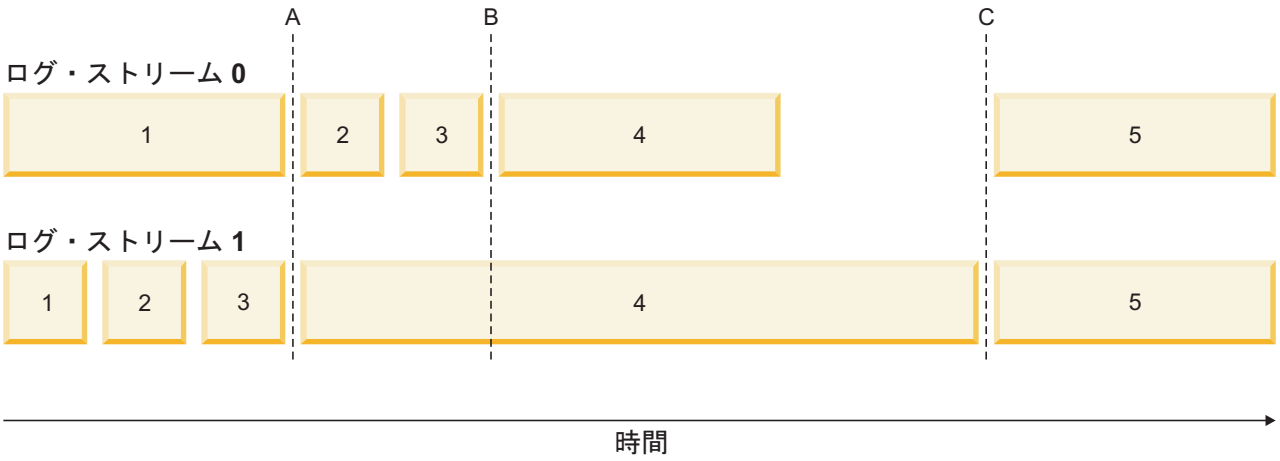


図 31. DB2 pureScale環境でのログ・ファイル

ログ・ストリーム 1 のログ・ファイル 4 のみが欠落しているシナリオを考えてください。時刻 A へのロールフォワード操作は成功しますが、時刻 B、時刻 C、または END OF LOGS へのロールフォワード操作は失敗します。ログ・ファイル 4 が使用不可のため、ROLLFORWARD コマンドは SQL1273N を戻します。さらに、

ログ・ストリーム 0 のファイル 2 および 3 のログ・レコードが、ログ・ストリーム 1 のログ・ファイル 4 の始めの部分と同じ時間枠の中で書き込まれているため、ロールフォワード操作は、ログ・ストリーム 1 のログ・ファイル 4 が使用可能になるまで、ログ・ファイル 2 および 3 を処理できません。その結果、ロールフォワード操作は時刻 A で停止し、それ以降のロールフォワード操作は、ストリーム 1 のログ 4 が使用可能になるまで、時刻 A より先に進むことができません。

ロールフォワード操作時に、ログ・ストリーム 0 のログ・ファイル 4 のみが欠落している、別のシナリオを考えてください。 **ROLLFORWARD** コマンドに **END OF LOGS** オプション (または、時刻 B 以後の任意の時刻) を指定して発行すると、ストリーム 0 のログ・ファイル 4 が欠落しているため、操作は時刻 B で停止し、SQL1273N を戻します。ロールフォワード操作は、ログ・ストリーム 0 のファイル 2 および 3 からのログ・レコードと、ストリーム 1 のファイル 4 からのログの一部 (時刻 B まで) を適用できます。このロールフォワード操作は、ストリーム 1 からの追加のログが使用可能であっても、時刻 B で停止しなければなりません。なぜなら、ログのマージ操作では、すべてのストリームのすべてのログが使用可能でなければならないためです。

損失ログ・ファイルが見つかった場合は、そのログ・ファイルを使用できるようにし、**ROLLFORWARD DATABASE** コマンドを再実行します。損失ログ・ファイルが見つからない場合は、 **ROLLFORWARD DATABASE...STOP** コマンドを実行して、損失ログ・ファイルの直前の最後の整合ポイントで、ロールフォワード操作を完了させます。

損失ログを検出することによって、損失ログ・ファイルに起因するデータベース破壊は発生しなくなりますが、損失ログ・ファイルがある場合、一部のトランザクションを再生できないため、それらの損失ログ・ファイルが見つかられなければ、結果的にデータ損失が発生する可能性があります。

必要なリソース

ログ・ストリーム・マージ操作には、追加の EDU が必要です。データベースを活動化するときに、各メンバー上に **db2lfr** EDU が 1 つずつ作成されます。ログ・ストリーム・マージを必要とするログ読み取り操作が開始されると、ログ・ストリームごとに 1 つの **db2shred** EDU と 1 つの **db2lfr** EDU が作成されます。**db2lfr**-**db2shred** の各グループが、独自に、ログ・ページおよびログ・レコード・バッファーのセットを割り当てますが、追加のメモリーまたはシステム・リソース量は多くはありません。ログ・ストリーム・マージの関与する各メンバー用に、約 400 KB が割り当てられます。

ログ・ストリーム・マージ操作では、メンバーが、他のメンバー所有のログ・ファイルを、自身のオーバーフロー・ログ・パス、1 次ログ・パス、またはミラー・ログ・パスにリトリーブします。DB2 pureScale環境の場合、ロールフォワード操作を開始する前に、リトリーブ・パスに十分な空きディスク・スペースがあることを確認してください。これによって、 DB2 pureScale環境で必要な多数のファイルを、パフォーマンスに影響を与えずに、アーカイブからリトリーブする操作が可能になります。すべてのメンバーのアクティブ・ログ・ファイルをリトリーブするのに必要な容量の概算は、次の計算式を使用して求めることができます: (**logprimary** + **logsecond**) * メンバー数。

例

- **newlogpath** グローバル・データベース構成パラメーターを更新します。

```
db2 update db cfg for db mydb using newlogpath /home/dbuser/logdir
```

- 1 つのメンバーに対して個別設定されているデータベース構成パラメーター **max_log** を更新します。

```
db2 update db cfg for db mydb member 1 using max_log 5
```

- 1 次ログ・パスを次のように更新します。

```
db2 connect to mydb
db2 update db cfg for mydb using newlogpath /home/dbuser/newlogpath
db2 get db cfg for mydb
...
Changed path to log files   (NEWLOGPATH) = /home/dbuser/newlogpath/NODE0000/LOGSTREAM0000/
Path to log files                        = /home/dbuser/dbuser/NODE0000/LOGSTREAM0000/
...
```

この変更は、まだメンバーがアクティブであるために、反映されていません。

```
db2 terminate
db2 deactivate db mydb
db2 connect to mydb
db2 get db cfg for mydb
...
Changed path to log files   (NEWLOGPATH) =
Path to log files                        = /home/dbuser/newlogpath/NODE0000/LOGSTREAM0000/
...
```

各メンバーが /home/dbuser/newlogpath/NODE0000/LOGSTREAM*xxxx* というログ・パスを使用するようになります。ここで *xxxx* は、このパスを使用するログ・ストリームのログ・ストリーム ID です。

- バックアップ・イメージのリストア中に新しい 1 次ログ・パスを設定します。

```
db2 restore db mydb newlogpath '/home/dbuser/newlogpath' without prompting
```

DB2 pureScale環境のログ・シーケンス番号

DB2 データベースは、ログ・シーケンス番号 (LSN) という 64 ビットの識別子を使用して、ログ・レコードを生成した操作の順序を判別します。

LSN は、増加し続ける値です。各メンバーは、独自のログ・ファイル・セット (ログ・ストリーム) に書き込みます。単一のログ・ストリーム内の LSN は、固有の番号です。

LSN は各メンバーで独立して生成され、複数のログ・ストリームが存在するため、異なるログ・ストリーム間で重複する LSN 値を持つ可能性があります。ログ・レコード識別子 (LRI) は、ログ・ストリーム間でログ・レコードを識別するために使用されます。データベース内のすべてのログ・ストリームに含まれる各ログ・レコードに、固有の LRI が割り当てられます。リカバリー操作で処理中の LRI を調べるには、 **db2pd** コマンドを使用します。

第 3 部 付録

付録 A. DB2 技術情報の概説

DB2 技術情報は、さまざまな方法でアクセスすることが可能な、各種形式で入手できます。

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2インフォメーション・センター
  - トピック (タスク、概念、およびリファレンス・トピック)
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

注: DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks® 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (http://www.ibm.com/software/data/sw-library/) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、IBM Publications Center (www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss) から利用できる DB2 ライブラリーについて説明しています。英語および翻訳された DB2 バージョン 10.1 のマニュアル (PDF 形式) は、www.ibm.com/support/docview.wss?rs=71&uid=swg2700947 からダウンロードできます。

この表には印刷資料が入手可能かどうかが示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: *DB2 インフォメーション・センター*は、PDF やハードコピー資料よりも頻繁に更新されます。

表 35. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
管理 API リファレンス	SA88-4671-00	入手可能	2012 年 4 月
管理ルーチンおよびビュー	SA88-4672-00	入手不可	2012 年 4 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻	SA88-4676-00	入手可能	2012 年 4 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻	SA88-4677-00	入手可能	2012 年 4 月
コマンド・リファレンス	SA88-4673-00	入手可能	2012 年 4 月
データベース: 管理の概念および構成リファレンス	SA88-4662-00	入手可能	2012 年 4 月
データ移動ユーティリティー ガイドおよびリファレンス	SA88-4693-00	入手可能	2012 年 4 月
データベースのモニタリング ガイドおよびリファレンス	SA88-4663-00	入手可能	2012 年 4 月
データ・リカバリーと高可用性 ガイドおよびリファレンス	SA88-4694-00	入手可能	2012 年 4 月
データベース・セキュリティー・ガイド	SA88-4695-00	入手可能	2012 年 4 月

表 35. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 ワークロード管理ガイドおよびリファレンス	SA88-4685-00	入手可能	2012 年 4 月
ADO.NET および OLE DB アプリケーションの開発	SA88-4665-00	入手可能	2012 年 4 月
組み込み SQL アプリケーションの開発	SA88-4666-00	入手可能	2012 年 4 月
Java アプリケーションの開発	SA88-4669-00	入手可能	2012 年 4 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SA88-4670-00	入手不可	2012 年 4 月
SQL および外部ルーチンの開発	SA88-4667-00	入手可能	2012 年 4 月
データベース・アプリケーション開発の基礎	GI88-4279-00	入手可能	2012 年 4 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4280-00	入手可能	2012 年 4 月
グローバリゼーション・ガイド	SA88-4696-00	入手可能	2012 年 4 月
DB2 サーバー機能 インストール	GA88-4679-00	入手可能	2012 年 4 月
IBM データ・サーバー・クライアント機能 インストール	GA88-4680-00	入手不可	2012 年 4 月
メッセージ・リファレンス 第 1 巻	SA88-4688-00	入手不可	2012 年 4 月
メッセージ・リファレンス 第 2 巻	SA88-4689-00	入手不可	2012 年 4 月
Net Search Extender 管理およびユーザーズ・ガイド	SA88-4691-00	入手不可	2012 年 4 月
パーティションおよびクラスタリングのガイド	SA88-4697-00	入手可能	2012 年 4 月
pureXML ガイド	SA88-4686-00	入手可能	2012 年 4 月
Spatial Extender ユーザーズ・ガイドおよびリファレンス	SA88-4690-00	入手不可	2012 年 4 月

表 35. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
SQL プロシージャー言語: アプリケーションのイネーブルメントおよびサポート	SA88-4668-00	入手可能	2012 年 4 月
SQL リファレンス 第 1 巻	SA88-4674-00	入手可能	2012 年 4 月
SQL リファレンス 第 2 巻	SA88-4675-00	入手可能	2012 年 4 月
Text Search ガイド	SA88-4692-00	入手可能	2012 年 4 月
問題判別およびデータベース・パフォーマンスのチューニング	SA88-4664-00	入手可能	2012 年 4 月
DB2 バージョン 10.1 へのアップグレード	SA88-4678-00	入手可能	2012 年 4 月
DB2 バージョン 10.1 の新機能	SA88-4684-00	入手可能	2012 年 4 月
XQuery リファレンス	SA88-4687-00	入手不可	2012 年 4 月

表 36. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Connect DB2 Connect Personal Edition インストールおよび構成	SA88-4681-00	入手可能	2012 年 4 月
DB2 Connect DB2 Connect サーバー機能インストールおよび構成	SA88-4682-00	入手可能	2012 年 4 月
DB2 Connect ユーザーズ・ガイド	SA88-4683-00	入手可能	2012 年 4 月

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果の原因になったと考えられる条件の SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順

SQL 状態ヘルプを開始するには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate または ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。
例えば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

異なるバージョンの DB2 インフォメーション・センターへのアクセス

他のバージョンの DB2 製品の資料は、ibm.com® のそれぞれのインフォメーション・センターにあります。

このタスクについて

DB2 バージョン 10.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v10r1 です。

DB2 バージョン 9.8 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/ です。

DB2 バージョン 9.7 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/ です。

DB2 バージョン 9.5 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r5 です。

DB2 バージョン 9.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9/ です。

DB2 バージョン 8 のトピックについては、*DB2 インフォメーション・センター* の URL (http://publib.boulder.ibm.com/infocenter/db2luw/v8/) を参照してください。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

ローカルにインストールした DB2 インフォメーション・センターは、定期的に更新する必要があります。

始める前に

DB2 バージョン 10.1 インフォメーション・センターが既にインストール済みである必要があります。詳しくは、「*DB2 サーバー機能 インストール*」の『DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール』のトピックを参照してください。インフォメーション・センターのインストールに適用されるすべての前提条件と制約事項は、インフォメーション・センターの更新にも適用されます。

このタスクについて

既存の DB2 インフォメーション・センターは、自動で更新することも手動で更新することもできます。

- 自動更新は、既存のインフォメーション・センターのフィーチャーと言語を更新します。自動更新を使用すると、手動更新と比べて、更新中にインフォメーショ

ン・センターが使用できなくなる時間が短くなるというメリットがあります。さらに、自動更新は、定期的に実行する他のバッチ・ジョブの一部として実行されるように設定することができます。

- 手動更新は、既存のインフォメーション・センターのフィーチャーと言語の更新に使用できます。自動更新は更新処理中のダウン時間を減らすことができますが、フィーチャーまたは言語を追加する場合は手動処理を使用する必要があります。例えば、ローカルのインフォメーション・センターが最初は英語とフランス語でインストールされており、その後ドイツ語もインストールすることにした場合、手動更新でドイツ語をインストールし、同時に、既存のインフォメーション・センターのフィーチャーおよび言語を更新できます。しかし、手動更新ではインフォメーション・センターを手動で停止、更新、再始動する必要があります。更新処理の間はずっと、インフォメーション・センターは使用できなくなります。自動更新処理では、インフォメーション・センターは、更新を行った後に、インフォメーション・センターを再始動するための停止が発生するだけで済みます。

このトピックでは、自動更新のプロセスを詳しく説明しています。手動更新の手順については、『コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新』のトピックを参照してください。

手順

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センターを自動更新する手順を以下に示します。

1. Linux オペレーティング・システムの場合、次のようにします。
   a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、/opt/ibm/db2ic/V10.1 ディレクトリーにインストールされています。
   b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
   c. 次のように update-ic スクリプトを実行します。

      ```
      update-ic
      ```
2. Windows オペレーティング・システムの場合、次のようにします。
   a. コマンド・ウィンドウを開きます。
   b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、<Program Files>¥IBM¥DB2 Information Center¥バージョン 10.1 ディレクトリーにインストールされています (<Program Files> は「Program Files」ディレクトリーのロケーション)。
   c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。
   d. 次のように update-ic.bat ファイルを実行します。

      ```
      update-ic.bat
      ```

タスクの結果

DB2 インフォメーション・センターが自動的に再始動します。更新が入手可能な場合、インフォメーション・センターに、更新された新しいトピックが表示されます。インフォメーション・センターの更新が入手可能でなかった場合、メッセージがログに追加されます。ログ・ファイルは、doc¥eclipse¥configuration ディレクトリーにあります。ログ・ファイル名はランダムに生成された名前です。例えば、1239053440785.log のようになります。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

このタスクについて

ローカルにインストールされた *DB2 インフォメーション・センター* を手動で更新するには、以下のことを行う必要があります。

1. コンピューター上の *DB2 インフォメーション・センター* を停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。DB2 インフォメーション・センターのワークステーション・バージョンは、常にスタンドアロン・モードで実行されます。を参照してください。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールしなければならない更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

   注: ご使用の環境において、インターネットに接続されていないマシンに *DB2 インフォメーション・センター* の更新をインストールする必要がある場合、インターネットに接続されていて *DB2 インフォメーション・センター* がインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングしてください。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシーを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。
   更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。
3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の *DB2 インフォメーション・センター* を再開します。

注: Windows 2008、Windows Vista (およびそれ以上) では、このセクションの後の部分でリストされているコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを開くには、ショートカットを右クリックしてから、**「管理者として実行」**を選択します。

手順

コンピューターまたはイントラネット・サーバーにインストール済みの *DB2* インフォメーション・センター を更新するには、以下のようにします。

1. *DB2* インフォメーション・センター を停止します。
   - Windows では、「**スタート**」 > 「**コントロール パネル**」 > 「**管理ツール**」 > 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**停止**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv10 stop
     ```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
   - Windows の場合:
     a. コマンド・ウィンドウを開きます。
     b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、*Program_Files*¥IBM¥DB2 Information Center¥バージョン 10.1 ディレクトリーにインストールされています (*Program_Files* は Program Files ディレクトリーのロケーション)。
     c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。
     d. 次のように help_start.bat ファイルを実行します。

        ```
        help_start.bat
        ```
   - Linux の場合:
     a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、/opt/ibm/db2ic/V10.1 ディレクトリーにインストールされています。
     b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
     c. 次のように help_start スクリプトを実行します。

        ```
        help_start
        ```

   システムのデフォルト Web ブラウザーが開き、スタンドアロンのインフォメーション・センターが表示されます。
3. 「**更新**」ボタン () をクリックします。(ブラウザーで JavaScript が有効になっている必要があります。) インフォメーション・センターの右側のパネルで、「**更新の検索**」をクリックします。 既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「**更新のインストール**」をクリックします。
5. インストール・プロセスが完了したら、「**完了**」をクリックします。
6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。
   - Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help_end.bat ファイルを実行します。

```
help_end.bat
```

注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して停止しないでください。

- Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

```
help_end
```

注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを停止しないでください。

7. *DB2 インフォメーション・センター* を再開します。
   - Windows では、「**スタート**」 > 「**コントロール パネル**」 > 「**管理ツール**」 > 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**開始**」を選択します。
   - Linux では、以下のコマンドを入力します。

```
/etc/init.d/db2icdv10 start
```

タスクの結果

更新された *DB2 インフォメーション・センター* に、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 データベース製品のさまざまな機能について学習するための支援となります。この演習をとおして段階的に学習することができます。

はじめに

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML ガイド*」の『**pureXML®**』
: XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 の資料

トラブルシューティング情報は、「*問題判別およびデータベース・パフォーマンスのチューニング*」または *DB2 インフォメーション・センター* の『データベースの基本』セクションにあります。ここには、以下の情報が記載されています。

- DB2 診断ツールおよびユーティリティーを使用した、問題の切り分け方法および識別方法に関する情報。
- 最も一般的な問題のうち、いくつかの解決方法。
- DB2 データベース製品で発生する可能性のある、その他の問題の解決に役立つアドバイス。

IBM サポート・ポータル

現在問題が発生していて、考えられる原因とソリューションを見つけるには、IBM サポート・ポータルを参照してください。 Technical Support サイトには、最新の DB2 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

IBM サポート・ポータル (http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows) にアクセスしてください。

ご利用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

適用度: これらのご利用条件は、IBM Web サイトのあらゆるご利用条件に追加で適用されるものです。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

権利: ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、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なしで提供されます。

IBM の商標: IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。IBM 以外の製品に関する情報は、本書の最初の発行時点で入手可能な情報に基づいており、変更される場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
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。サンプル・プログラムは、現存するままの状態で提供されるものであり、いかなる種類の保証も提供されません。 IBM は、これらのサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (*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

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds 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 Oracle やその関連会社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 の米国およびその他の国における登録商標です。
- インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Celeron、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社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。
- 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。

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］

［ワ行］

[数字]

A

B

C

D

H

I

L

M

N

O

R

S

T

U

V

W

Printed in Japan

SA88-4694-00

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

IBM DB2 10.1 for Linux, UNIX, and Windows

データ・リカバリーと高可用性 ガイドおよびリファレンス