IBM solidDB
IBM solidDB Universal Cache
バージョン 7.0

高可用性ユーザー・ガイド

SA88-4580-00
(英文原典：SC27-3843-00)

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、 227 ページの『特記事項』に記載されている情報をお読みください。

本書は、バージョン 7 リリース 0 の IBM solidDB (製品番号 5724-V17) および IBM solidDB Universal Cache (製品番号 5724-W91)、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

原典： SC27-3843-00
IBM solidDB
IBM solidDB Universal Cache
Version 7.0
High Availability User Guide

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 2011.10

目次

図

表

本書について

IBM® solidDB® 高可用性 (HotStandby) コンポーネントを使用すると、データベース・システムの信頼性が向上し、ダウン時間が減少します。 HotStandby は、「ホット・スタンバイ」アプローチを使用します。このアプローチでは、2 次データベース・サーバーが 1 次サーバーと並行して稼働し、データの正確な最新コピーを保持します。1 次データベース・サーバーに障害が発生すると、高可用性コントローラー (HAC) が 2 次側への切り替えを行います。この切り替えはアプリケーションに対して透過的に行われ、コミットされたトランザクションが失われることはなく、またパフォーマンスに対する影響も最小限にとどまります。切り替え時間は非常に短時間であり、ハードウェアおよびソフトウェアの環境の特性によって異なりますが、200 から 300 ミリ秒程度に収まります。

本書には、HotStandby コンポーネントのみに固有の情報が記載されています。solidDB データベースに関する一般的な管理および保守情報については、「*IBM solidDB 管理者ガイド*」を参照してください。

本書は、読者が、データベース管理システム (DBMS) に関する一般的な知識を持っていることと、SQL および solidDB に精通していることを前提としています。

図に関する規則

本書では、サーバーの図をいくつか使用して、HotStandby 環境におけるいろいろなシナリオについて説明します。

以下の表に、サーバーの図の凡例を示します。

表 1. HotStandby の図に関する規則

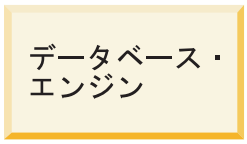
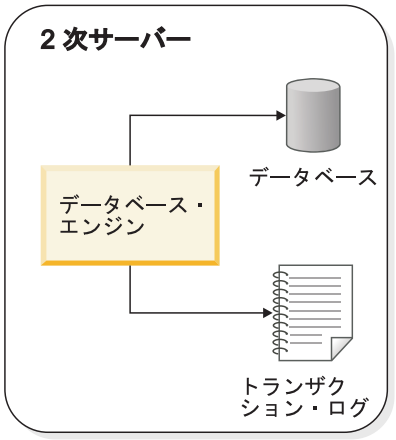
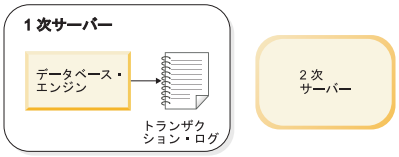
シンボル	説明
	長方形は、実行元のプログラム、つまりデータベース・サーバー (エンジン) 自体を表します。
 データベース	シリンダーは、データを表し、これは一般にはディスクに格納されています。あるいは、一部または全部がメモリーに格納されている場合もあります。
 トランザクション・ログ	このシンボルは、データベース・リカバリーと HotStandby の両方で使用されるトランザクション・ログ (Txn ログ) を表します。

表 1. HotStandby の図に関する規則 (続き)

シンボル	説明
	角丸長方形は、データとトランザクション・ログを伴う完全なサーバーを表します。角丸長方形の内部に **2 次サーバー**または **1 次サーバー**という語句がある場合、そのサーバーは HotStandby サーバーです。
	簡略化のために、データベース内のデータを表すシリンダーが省略される場合もあります。場合によっては、シンボルがさらに簡略化されて角丸長方形のみを示す場合もあります。どちらのアイコンも、サーバーの簡略化された表現です。

書体の規則

solidDB の資料では、以下の書体の規則を使用します。

表 2. 書体の規則

フォーマット	用途
データベース表	このフォントは、すべての通常テキストに使用します。
NOT NULL	このフォントの大文字は、SQL キーワードおよびマクロ名を示しています。
`solid.ini`	これらのフォントは、ファイル名とパス式を表しています。
`SET SYNC MASTER YES;` `COMMIT WORK;`	このフォントは、プログラム・コードとプログラム出力に使用します。SQL ステートメントの例にも、このフォントを使用します。
`run.sh`	このフォントは、サンプル・コマンド行に使用します。
TRIG_COUNT()	このフォントは、関数名に使用します。
java.sql.Connection	このフォントは、インターフェース名に使用します。
`LockHashSize`	このフォントは、パラメーター名、関数引数、および Windows レジストリー項目に使用します。
argument	このように強調されたワードは、ユーザーまたはアプリケーションが指定すべき情報を示しています。

表 2. 書体の規則 (続き)

フォーマット	用途
管理者ガイド	このスタイルは、他の資料、または同じ資料内の他の章の参照に使用します。新しい用語や強調事項もこのように記述します。
ファイル・パス表示	特に明記していない場合、ファイル・パスは UNIX フォーマットで示します。スラッシュ (/) 文字は、インストール・ルート・ディレクトリーを表します。
オペレーティング・システム	資料にオペレーティング・システムによる違いがある場合は、最初に UNIX フォーマットで記載します。UNIX フォーマットに続いて、小括弧内に Microsoft Windows フォーマットで記載します。その他のオペレーティング・システムについては、別途記載します。異なるオペレーティング・システムに対して、別の章を設ける場合があります。

構文表記法の規則

solidDB の資料では、以下の構文表記法の規則を使用します。

表 3. 構文表記法の規則

フォーマット	用途
`INSERT INTO` *`table_name`*	構文の記述には、このフォントを使用します。置き換え可能セクションには、この フォントを使用します。
`solid.ini`	このフォントは、ファイル名とパス式を表しています。
[ ]	大括弧は、オプション項目を示します。太字テキストの場合には、大括弧は構文に組み込む必要があります。
|	垂直バーは、構文行で、互いに排他的な選択項目を分離します。
{ }	中括弧は、構文行で互いに排他的な選択項目を区切ります。太字テキストの場合には、中括弧は構文に組み込む必要があります。
...	省略符号は、引数が複数回繰り返し可能なことを示します。
. . .	3 つのドットの列は、直前のコード行が継続することを示します。

1 IBM solidDB HotStandby の概要

高可用性 (HA) システムの目標は、システム障害を許容できる程度にすることです。高可用性を実装するため、solidDB HotStandby コンポーネントは 2 次サーバー (スタンバイ・サーバー) を 1 次サーバー (アクティブ・サーバー) と並行して実行し、2 次サーバーに 1 次サーバー内のデータの最新コピーを保持させることができます。

HotStandby コンポーネントは内部の状態マシンを実装し、これによりサーバーは HA 状態を認識します。HA 状態マシンにより、データベースの整合性を維持することが可能になります。例えば、サーバーが 2 次サーバーの状態である場合 (つまり、1 次サーバーからトランザクション・ストリームを受け取っている場合)、2 次データベースに対する更新は無効になります。

サーバーの可用性または状態は、*watchdog* と呼ばれるエンティティーで制御できます。solidDB には、solidDB 高可用性コントローラー (HAC) と呼ばれる Watchdog 実装が用意されています。

内部で、HAC は一連の HotStandby コマンド (HotStandby API) を使用してサーバーの状態を制御します。このようなソリューションでは、より信頼性の高いシステムの実装が可能です。データベース・サーバーに障害が起きても、サイトが完全に停止することはなくなります。solidDB でサポートされているエンジン構成 (マスター・サーバーとレプリカ・サーバーによる solidDB の拡張レプリケーションなど) では、わずか数百ミリ秒のうちに、HotStandby によって、障害のあるデータベースを 2 次データベースに置き換えることができます。

HotStandby アーキテクチャー

HotStandby に含まれているコンポーネントは、以下のとおりです。

- Watchdog アプリケーション (HAC など)
- HotStandby API (HSB 管理コマンド)
- 1 次および 2 次 solidDB サーバー

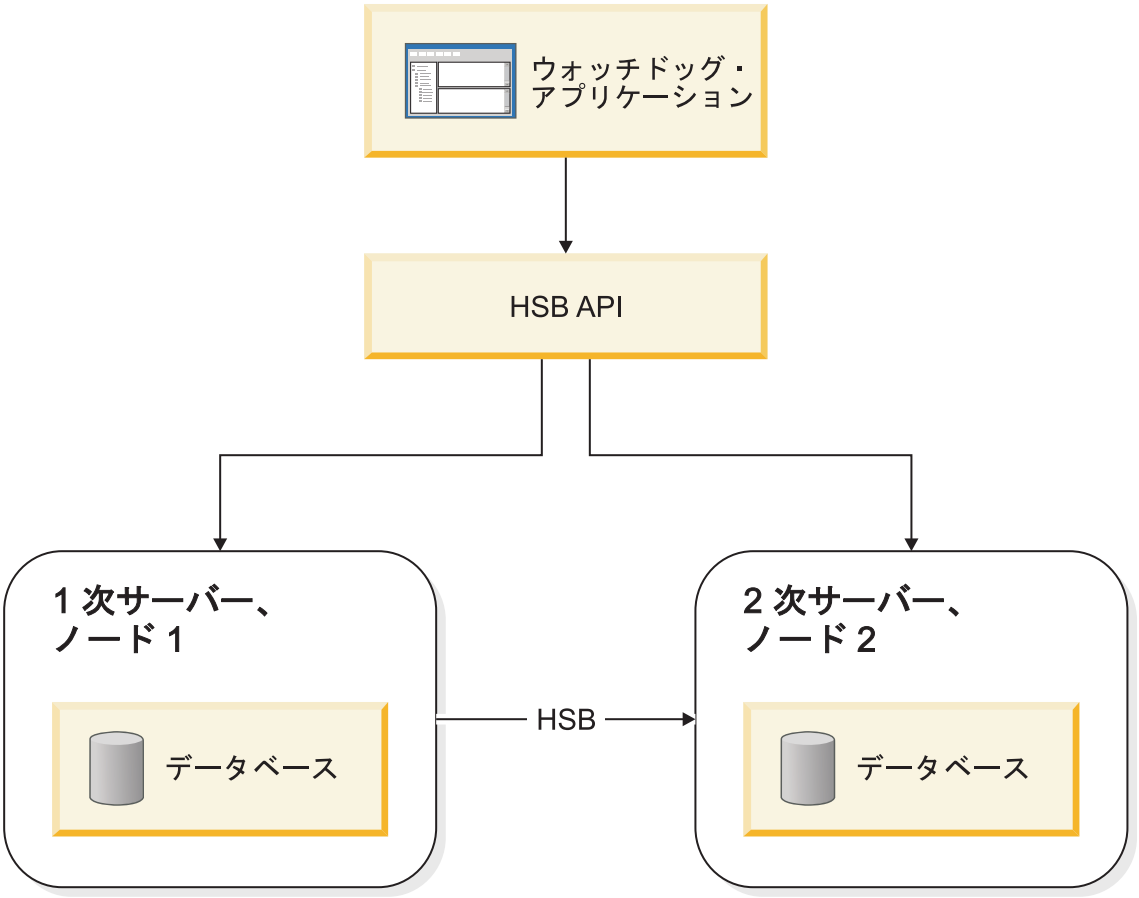


図 1. HotStandby アーキテクチャー

操作の原則

HotStandby (HSB) は、1 次サーバーと 2 次サーバーという 2 つのノード間で、同期トランザクション・レプリケーションを行います。

1 次サーバー・ノード (*1 次側*) には、アクティブ・データベースが入っています。2 次サーバー・ノード (*2 次側*) には、アクティブ・データベースの正確な最新コピーが入っており、このノードは 1 次サーバーに障害が起きた場合、1 次サーバーに取って代わることができます。

2 次サーバーは 1 次サーバーから更新情報を受け取り、元の 1 次サーバーが障害を起こした場合は、いつでも 1 次サーバーとして後を引き継ぐことができる状態になっています。それ以外の、2 次サーバーを使用することの利点は、クライアントからの読み取り専用の要求 (例えば、SELECT ステートメント) に 2 次サーバーも応答できることです。これにより、ワークロードの一部を 1 台でなく 2 台のサーバーに分散できます。

注: 「ホット・スタンバイ (hot standby)」という用語 (すべて英小文字の 2 ワード) は、最初のサーバーが障害を起こしたときに、2 番目のサーバーがいつでもその機能を引き継げるようにしておく一般的な技法を指しています。「HotStandby」(このように英大文字を使用した 1 ワード) は、この一般的な技法の solidDB 固有の実装を指しています。HotStandby の省略形は、HSB です。

同様に、Watchdog は 2 つのデータベースの状態を監視し、必要な場合に状態を切り替えることができるテクノロジーを指しています。HAC は、solidDB の Watchdog 実装です。また、HSB API を使用して独自の Watchdog アプリケーション・プログラムを作成することもできます。solidDB パッケージには、開始点として使用できる Watchdog のサンプルも含まれています。

1.1 HotStandby の主な機能

このセクションでは、HotStandby の主な機能について説明します。

1.1.1 HotStandby API (HSB 管理コマンド)

HotStandby サーバーの高可用性動作は、solidDB 管理コマンドのサブセットに基づいた API によって制御されます。

このコマンドのサブセットは、コマンド接頭部 **hotstandby** (省略形は **hsb**) で識別されます。これらのコマンドは、(**solsql** のような) 任意の SQL 対応ツールか、ODBC や JDBC などのプログラマチック・インターフェースを使用して発行できます。HotStandby 管理コマンドの構文は、以下のとおりです。

```
admin command 'hotstandby hsb-command options';
```

または

```
admin command 'hsb hsb-command options';
```

hotstandby 管理コマンドでは、solidDB HSB サーバーの高可用性状態の制御、または状態情報の取得を行います。これらのコマンドを手動で、またはプログラムで発行できます。solidDB HA 管理ツール、高可用性コントローラー (HAC)、および Watchdog サンプルでは、これらのコマンドをプログラムで使用します。

1.1.2 基本的な HotStandby サーバー・スキーム

基本的な HotStandby サーバー・スキームでは、2 つのデータベース・サーバー、つまり *1 次サーバー* と *2 次サーバー* が存在します。それぞれが独自のディスク・ドライブを持ち、データベースをそこに保管し、それぞれが独自のトランザクション・ログ (*Txn ログ*) を持ちます。

4 ページの図 2 は、基本的な HotStandby サーバー・スキームを示した図です。1 次サーバーは、それ自体のトランザクション・ログに書き込みを行い、そのログを 2 次サーバーへ転送して、2 次サーバーがそれ自体のデータベース・コピーに同じ変更を加えることができるようにします。2 次サーバーのトランザクション・ログは HSB にアクティブには関与しませんが、コミットされた後にメイン・データ表にまだ書き込まれていないデータを 2 次サーバーがリカバリーできるよう、保守されます。

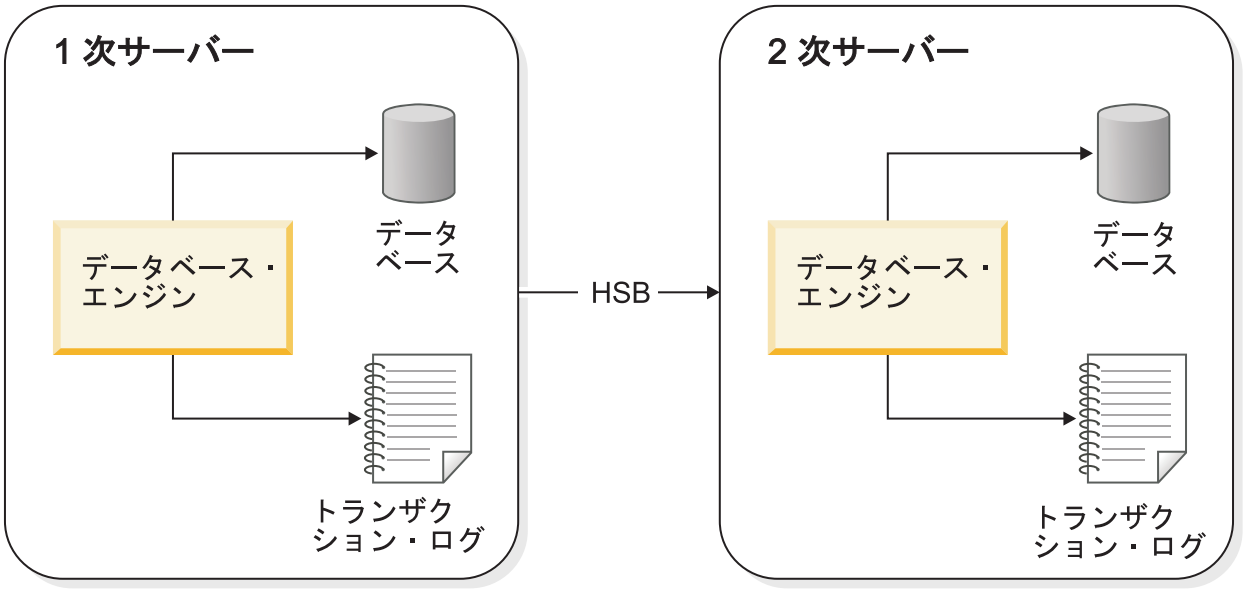


図 2. HotStandby サーバー・スキーム

ハートビート

solidDB HSB は、内部的にはハートビート と呼ばれる技法を使用してサーバー間の接続をモニターします。

アクティブ・サーバーとスタンバイ・サーバーとの間では、キープアライブ・メッセージのシーケンスが送信されます。両方のサーバーは、自己の存在を示すそれらの単一方向のメッセージを、絶えず相手側のサーバーへ送信します。メッセージは一定の時間間隔で送信されます。相手側のサーバーからのメッセージは、事前定義された時間枠内に到着することが予期されています。solidDB では、ハートビートの技法は *ping* と呼ばれます。

重要: solidDB では、ハートビートの技法は *ping* と呼ばれますが、ping 要求は送信されません。TCP/IP ネットワークで使用される Ping プロトコルと混同しないようにしてください。

トランザクション・ログと HotStandby

HotStandby は、1 次サーバーのトランザクション・ログを使用します。このログには、サーバー上でコミットされたトランザクションのコピーが入っています。非 HotStandby サーバーでは、このトランザクション・ログはサーバーが異常なシャットダウンを起こした場合に、データをリカバリーするために使用されます。

HotStandby 1 次サーバーでは、更新する必要があるデータを 2 次サーバーに知らせるために、ログ・データが 2 次サーバーへも送信されます。2 次側のデータベースは絶えずロールフォワード・プロセスを実行し、このプロセスにより、ログ・データを受信して 2 次側のデータ・コピーと 1 次側のコピーとの同期を保ちます。

1 次サーバーに障害が起きると、Watchdog アプリケーションは 2 次サーバーに 1 次サーバーになるよう指示します。新しい 1 次サーバーが稼働すると、クライアントはそれに接続して作業を続行できます。クライアントには、1 次サーバーがダウ

ンする前にコミットしたすべてのデータが表示されます。(クライアントは、元の 1 次サーバーがダウンした時点で開始されていて終了していなかったトランザクションを再開する必要があります。)

フェイルオーバーと切り替えの際にクライアントが HSB 環境で動作できるようにするため、*透過接続 (TC)* と呼ばれる特殊なタイプのクライアント接続を使用できます。詳細については、81 ページの『4, アプリケーションでの HotStandby の使用』を参照してください。

2 次サーバーに障害が起きた場合は、1 次サーバーが動作を続行できます。1 次サーバーはトランザクション・ログへのデータの書き込みを続行し、そのトランザクション・ログを、1 次サーバーと 2 次サーバーが互いに再接続して 1 次サーバーが 2 次サーバーへログを送信し終わるまで保持します。1 次サーバーがログを保持する時間の正確な長さは、solid.ini 構成パラメーター **General.CheckpointDeleteLog** および **General.BackupDeleteLog** の設定によって異なります。

1. **General.CheckpointDeleteLog**=Y の場合、1 次サーバーは、2 次サーバーがダウンした時刻と最新のチェックポイントのどちらか古い方の時点以降、すべてのトランザクション・ログを保持します。
2. **General.CheckpointDeleteLog**=N で **General.BackupDeleteLog**=Y の場合、1 次サーバーは 2 次サーバーがダウンした時刻と最新のバックアップのどちらか古い方の時点以降、すべてのトランザクション・ログを保持します。
3. **General.CheckpointDeleteLog**=N で **General.BackupDeleteLog**=N の場合、サーバーはログを無期限に保持します。

障害を起こしたデータベース・サーバーが再び使用可能になった後、そのサーバーを構成して、新しい 2 次データベース・サーバーにすることができます (障害を起こしていないサーバーは、既に現行の 1 次サーバーとして機能しています)。

1 次サーバーが障害を起こしているサーバーならば、各サーバーはそれぞれの責任を逆にして、元の 2 次サーバーが 1 次サーバーとして動作を引き継ぎ、元の 1 次サーバーは、修復された後に新しい 2 次サーバーとしてシステムに復帰します。これらの逆転は、障害が起きるたびに発生する可能性があります。どちらのサーバーも 1 次側になりえることから、システムは長期にわたって複数の障害に耐えて存続でき、事実上無期限に動作を続行できます。

注: 1 次サーバーが 2 次サーバーと長い期間接続できなくなった場合は、トランザクション・ログが、使用可能なすべてのディスク・スペースを満たしてしまう可能性があります。これは、構成パラメーターの適切な設定によって回避できます。75 ページの『3.6.3, トランザクション・ログのスペース不足』を参照してください。

HSB は、ハードウェアおよびソフトウェアのアップグレード時に、ダウン時間の削減のために使用することができます。一方のサーバーを 1 次サーバーとして動作させたままにしておき、その間にもう一方をアップグレードします。

HotStandby は、速度と安全のバランスをカスタマイズするために役立てることもできます。HSB パラメーター **HotStandby.SafenessLevel** および **HotStandby.2SafeAckPolicy** は、2 次サーバーによるトランザクションの確認応答の方法を制御します。ロギング関連の **Logging.DurabilityLevel** パラメーターと一

緒に使用して、これらのパラメーターでは、速度と安全性の組み合わせを指定できます。いくつかのパラメーター設定は、非 HSB サーバー上のパフォーマンスを高めることができます (詳細については、 72 ページの『3.5, パフォーマンスのチューニング』で持続性レベルと安全性の各パラメーターについての説明を参照してください)。

その一方で、**HotStandby.SafenessLevel=auto** 設定を使用することにより、持続性レベルとの関連で安全性レベルを動的に変更することもできます。

フェイルオーバー

フェイルオーバーでは、2 次サーバーが新しい 1 次サーバーに切り替えられます。

2 次サーバーを新しい 1 次サーバーに切り替えるのには、いくつかの理由があります。

1. 1 次側に障害が起きたとき
2. 1 次側を管理したいとき
3. システム上に既存の 1 次側が存在しないときに 1 次側を選択する必要があるとき

2 次サーバーを新しい 1 次サーバーになるよう切り替えるには、2 次サーバー上で次のコマンドを発行します。

```
ADMIN COMMAND 'hotstandby set primary alone';
```

フェイルオーバーの場合、新しい 1 次側は、古い 1 次側データベースからの最新のコミットされたデータを含んでいます。1 次側のデータベース内でコミットされたすべてのものは、2 次側のデータベースから検出できることが保証されます。透過接続 (TC) を使用している場合は、フェイルオーバー時でも接続が失われません。ただし、進行中のトランザクションは打ち切られるので、再実行する必要があります。詳細については、 95 ページの『4.2.2, 透過接続での障害透過性』を参照してください。新しい 1 次側は単独で動作でき、引き続きトランザクションとデータを自身のデータベースとトランザクション・ログに書き込みます。

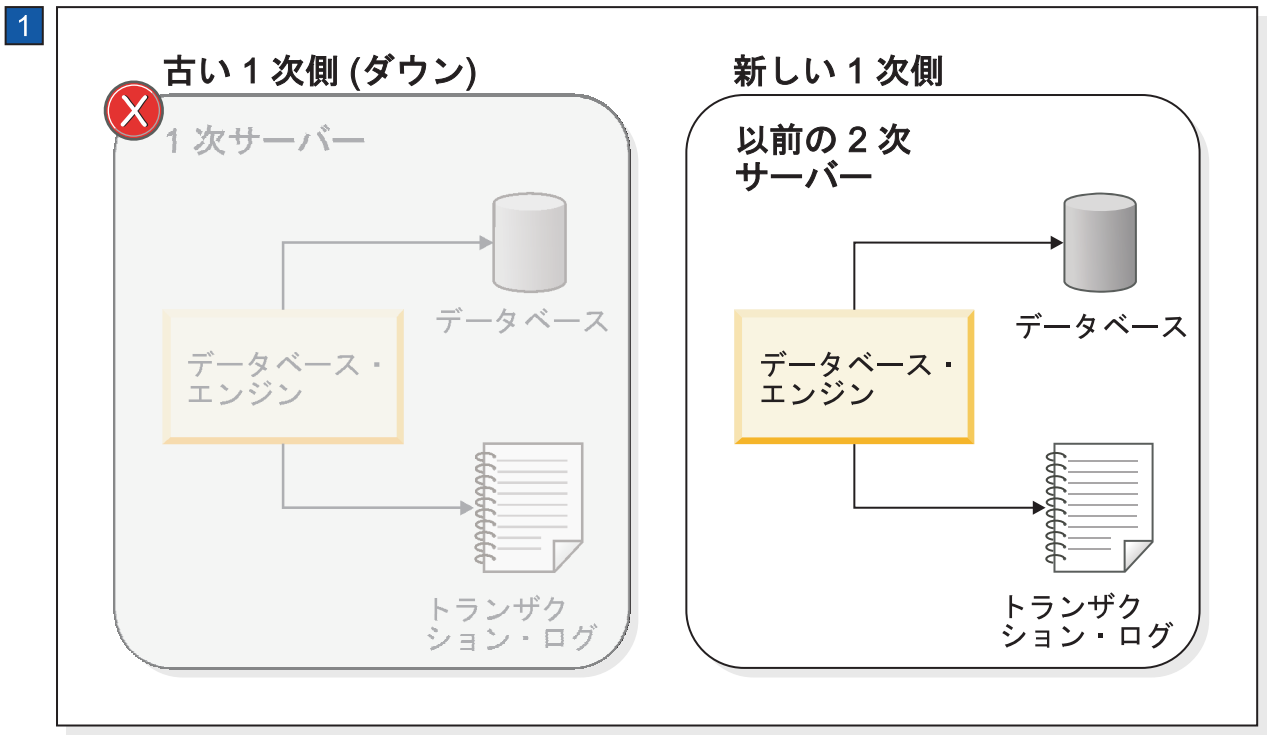


図 3. 新しい 1 次側 (古い 2 次側) への HotStandby 切り替え

1. 当初 2 次側だったサーバーは、古い 1 次サーバーに障害が起きると新しい 1 次サーバーになります。

サーバー・キャッチアップ

古い 1 次側は、オンラインに戻った後、(既存の 1 次側が存在する場合は) 新しい 2 次側になります。この段階で、新しい 2 次側内の情報は新しい 1 次側の情報より後れています。新しい 1 次側のデータベースには新しいトランザクションがコミットされているからです。新しい 2 次側を最新の状態にするために、サーバー同士が接続された後に、新しい 1 次側のトランザクション・ログ・データが自動的に新しい 2 次側へ送信されます。保留中のすべての変更がトランザクション・ログから新しい 2 次側へ書き込まれるため、2 次側は 1 次側との同期を保持できます。サーバー・キャッチアップの図を 8 ページの図 4 に示します。

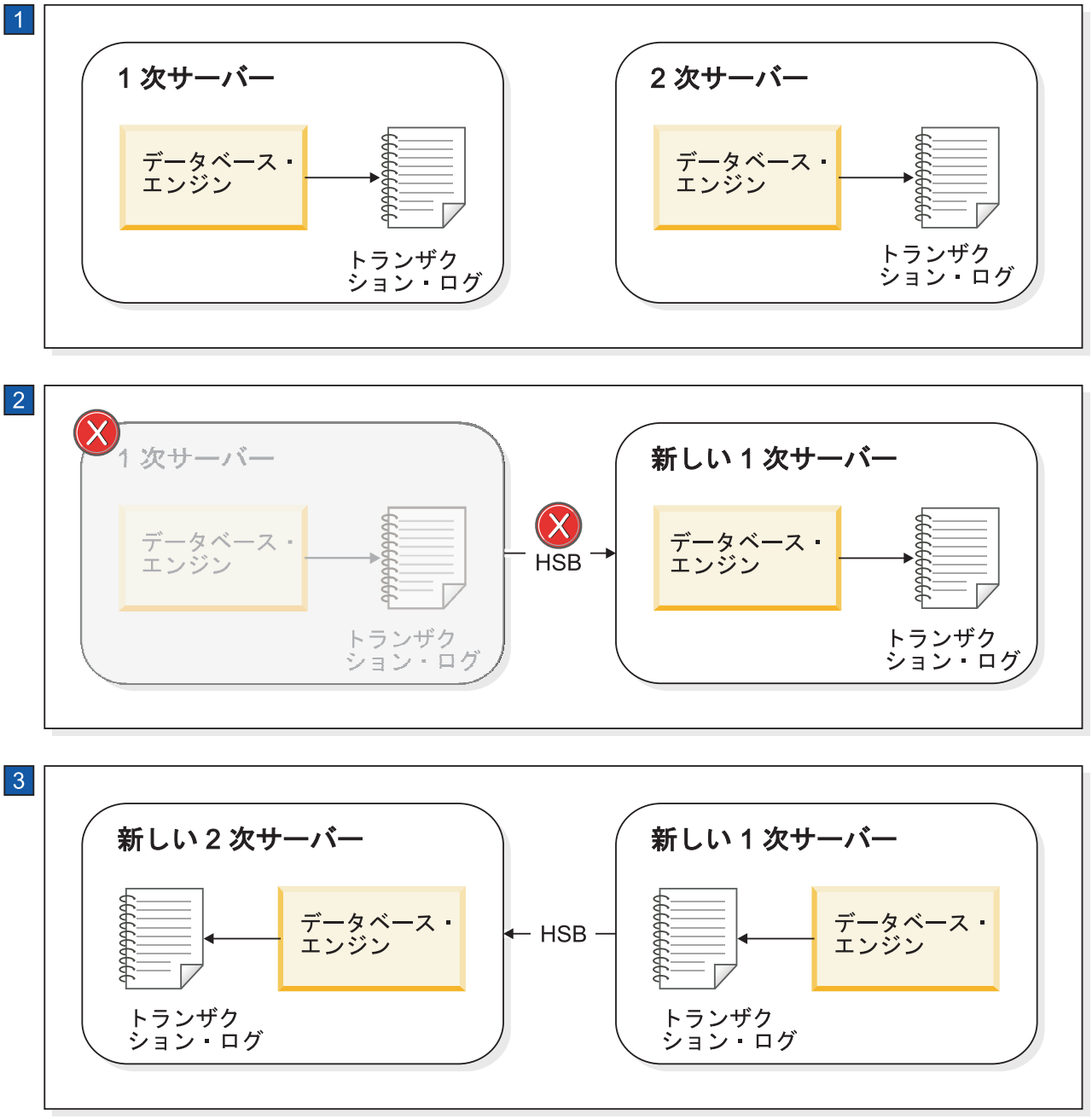


図 4. サーバーのフェイルオーバーとキャッチアップの例

1. 通常の動作: 1 次サーバーが 2 次サーバーにデータを送信します。
2. 1 次サーバーで障害が発生すると、2 次サーバーが新しい 1 次サーバーとして後を引き継ぎます。新しい 1 次サーバーは、後で新しい 2 次サーバーにデータを送信できるように、そのトランザクション・ログにトランザクション情報を保存します。
3. 古い 1 次サーバーが新しい 2 次サーバーとして立ち上げられると、新しい 2 次側がキャッチアップできるように、新しい 1 次側のトランザクション・ログの情報が 2 次側に送信されます。

1.1.3 サーバー HotStandby 状態

HotStandby システムでは、各サーバーは可能な複数の状態のうち、そのサーバーの現行動作を説明しているいずれかの状態に置かれます。

例えば、1 次サーバーと 2 次サーバーが通信して同期を保っている場合、それらはそれぞれ、PRIMARY ACTIVE 状態と SECONDARY ACTIVE 状態にあります。別の例として、1 次サーバーと 2 次サーバーとの接続が失われた場合、1 次サーバーは自動的に PRIMARY UNCERTAIN 状態に切り替わります。その状態では、新しいトランザクションを受け入れません。ユーザーは (または、より一般的には HAC は) サーバーを、独立したサーバーとして動作する PRIMARY ALONE 状態に切り替えます。この状態では、サーバーは新しいトランザクションを受け入れ、後で 2 次サーバーへ送信するためにそのトランザクションを保管します。

サーバー状態の説明

HotStandby (HSB) ペアの両方のサーバーは、照会と操作が可能な特定の状態を公開します。

状態とその意味は、以下のリストに示すとおりです。

表 4. サーバー状態の説明

状態	説明
PRIMARY ACTIVE	サーバーは接続され、このサーバー (1 次サーバー) は読み取り/書き込みトランザクションを受け入れており、データを 2 次サーバーへ送信しています。2 次サーバーは、SECONDARY ACTIVE 状態でなければなりません。
PRIMARY ALONE	ピア・サーバーが相互接続されていません。ピアは立ち上がっている可能性がありますが、接続されておらず、したがってトランザクションを受け入れません (つまり、SECONDARY ALONE 状態である可能性があります)。 このサーバー (1 次サーバー) は、読み取り/書き込みトランザクションをアクティブに受け入れて実行しており、後で 2 次サーバーへ送信するためにトランザクションを収集しています。

表 4. サーバー状態の説明 (続き)

状態	説明
PRIMARY UNCERTAIN	サーバー同士の接続が異常切断されており、**AutoPrimaryAlone** 構成パラメーターが「No」に設定されています。 PRIMARY UNCERTAIN 状態では、確認応答されていないすべてのトランザクションが保留状態のままです。つまり、サーバーは HAC がサーバーを別の状態に変更するまで、それらのトランザクションをコミットもロールバックもしません。 オペレーターに可能なアクションは、3 つあります。1 次サーバーを 2 次サーバーに再接続するか、1 次サーバーを PRIMARY ALONE 状態に設定するか、1 次サーバーを SECONDARY ALONE 状態に設定するかです。 1. サーバーを 2 次サーバーに再接続した場合、トランザクションは 1 次サーバー上でコミットされます。 2. 状態を PRIMARY ALONE に変更した場合、オープン・トランザクションは 1 次サーバー上でコミットされます。 3. 状態を SECONDARY ALONE に変更した場合、オープン・トランザクションは保留中のままになります。それらはサーバーが別の状態に変更された後に、最終的に解決されます。例えば、サーバーが SECONDARY ACTIVE 状態に移された場合、ブロックされていたトランザクションはキャッチアップの結果に応じて、打ち切られるかコミットされます。サーバー状態が STANDALONE または PRIMARY ALONE に変更された場合、ブロックされていたトランザクションはコミットされます。 2 次サーバーとの接続を失った 1 次サーバーを、PRIMARY UNCERTAIN でなく PRIMARY ALONE に自動的に移行させたい場合は、**AutoPrimaryAlone** 構成パラメーターの説明を参照してください。 **注:** HAC は、常にサーバーを PRIMARY UNCERTAIN から SECONDARY ALONE に切り替えることにより、最大の安全性を得ることができます。これにより、1 次サーバーの二重化が防止されます。ただし、ユーザーがサーバー上のデータを更新することもできなくなります。(74 ページの『3.6.2, ネットワーク分割と二重 1 次サーバー』を参照してください。)

表 4. サーバー状態の説明 (続き)

状態	説明
SECONDARY ACTIVE	ピア・サーバーは相互に接続されており、このサーバーは 1 次側からの着信トランザクション・ログ・データを受け入れています。それらのトランザクションは、2 次側が 1 次側と同じデータを持つよう、2 次側で実行されます (2 次側に障害が起きた場合でも、2 次側自体でデータをリカバリーできるよう、トランザクションも 2 次側自体のトランザクション・ログに書き込まれます)。さらに、クライアントは SECONDARY ACTIVE 状態のサーバー上で、読み取り専用トランザクションを実行できます。サーバーが SECONDARY ACTIVE 状態にある場合は、そのサーバーのピアは PRIMARY ACTIVE 状態にあることが必要です。
SECONDARY ALONE	2 次サーバーは、ピア・サーバーから切断されています。読み取り要求だけが受け入れられます。2 次サーバーか 1 次サーバー上で「HotStandby connect」コマンドを発行すると、サーバーをピアに接続できる場合があります。
STANDALONE	サーバーは HSB 状態 (1 次サーバーまたは 2 次サーバー) を持たず、通常のスタンドアロン・サーバーとして動作します。トランザクション・ログも通常の方法で処理および除去され、2 次サーバー用に保存されません。HSB 動作を再開するためには、サーバーを PRIMARY ALONE または SECONDARY ALONE に設定する必要があり、1 次サーバーは **netcopy** または **copy** 動作を実行して、2 次サーバーにデータベースの完全なコピーを送信する必要があります。
OFFLINE	サーバーは「netcopy listen モード」(「backupserver モード」とも呼ばれます) で始動しました。このモードでは、サーバーは PRIMARY ALONE 状態にあるサーバーからの着信 **netcopy** を待っています。サーバーは、**netcopy** を正常に完了すると、SECONDARY ALONE 状態に移行します。 OFFLINE 状態を直接監視することはできません。OFFLINE 状態のサーバーはクライアント接続を受け入れないからです。OFFLINE 状態のサーバーに接続しようとすると、エラー・コード 14552、「Server is in backup server mode, no connections are allowed」が返されます。OFFLINE 状態のサーバーは、**netcopy** 操作 (後述します) に対してのみ応答します。

1.1.4 HotStandby でのレプリケーション・モード

HotStandby レプリケーション・プロトコルの目的は、トランザクションの結果を 1 次サーバーから 2 次サーバーに安全に伝達することです。solidDB には、同期 (*2-safe*) と非同期 (*1-safe*) の両方のレプリケーション・プロトコルが用意されています。さまざまな持続性 (ロギング) レベルと組み合わせて、レプリケーション・プロ

トコルを使用することで、パフォーマンスと耐久性の必要なバランスに合わせて HotStandby システムを調整することができます。

- `1-safe`: トランザクションは、最初に 1 次側でコミットされてから、2 次側へ転送されます。
- `2-safe`: トランザクションは、2 次側による確認応答が完了するまでコミットされません (デフォルト)。

安全性レベルは、以下の 3 つのレベルで制御できます。

- グローバル - **`HotStandby.SafenessLevel`** パラメーター
- セッション - SET SAFENESS ステートメント
- トランザクション - SET TRANSACTION SAFENESS ステートメント

2-safe レプリケーションでの同期 HotStandby

1 次側と 2 次側が正確に同じデータを持つようにするために、solidDB は主に*同期 HotStandby* モデルを使用します。これは、*2-safe* レプリケーション方式と呼ばれ、データが 2 つの場所に書き込まれた後に、データがコミットされたことがユーザーに通知されます。

1 次サーバーは、トランザクションに対する変更を 1 次側のデータベース内でコミットする前に、トランザクション・データを 2 次サーバーへ送信します。 2 次サーバーは 1 次側に対し、データをコミットした、または少なくとも受信したことの確認応答を送信する必要があります。それがない場合、1 次サーバーはタイムアウトになり、状態が PRIMARY ACTIVE から PRIMARY UNCERTAIN に変化します。この場合、1 次サーバーはトランザクションのロールバックもコミットも行えません。HAC は 1 次サーバーを PRIMARY ALONE 状態に設定する場合があり、それによって 1 次側は、トランザクションの受信と、2 次側に依存しない動作を続行できます。1 次側は 2 次側へ送信した保留中のトランザクションをコミットし、新しいトランザクションの受け入れを再開します。

注: 2 次サーバーはトランザクション・ログ項目のコミット (または少なくとも受信) を完了すると同時に確認応答を送信します。この構成により、単一の障害が起きたときでも、トランザクションが失われずに済みます。さらに、システム全体の障害が発生した場合のデータベースのリカバリーを容易にするために、オプションとしてファイル・ベースのトランザクション・ログが保存されます。

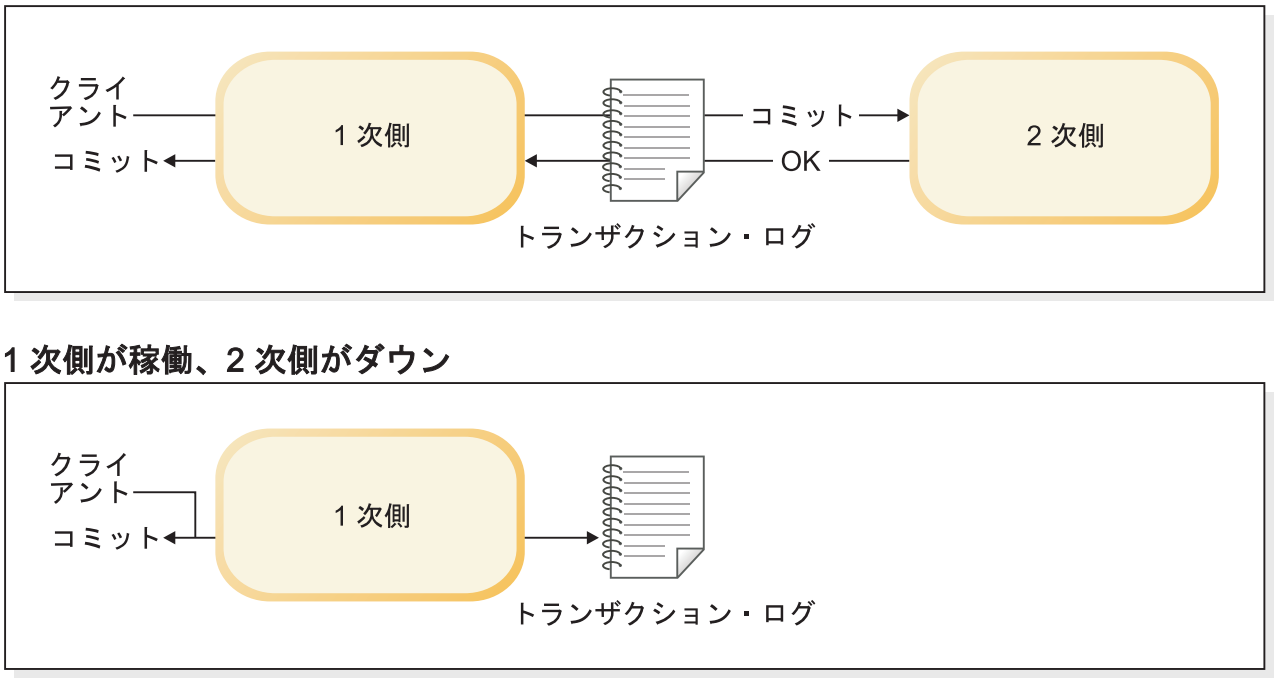


図 5. 同期 HotStandby 構成

データの送信での基本ステップ

以下に、同期レプリケーションでデータを送信する際のステップを示します。

1. 1 次サーバーは、1 次側ノードのトランザクション・ログにデータを (レコード・レベルの形式で) 書き込みます。
2. 1 次サーバーがコミット・ステートメントを検出すると、変更されたすべてのデータが 2 次サーバーへ送信されます。

   注: トランザクションの開始後、しかも 1 次サーバーがデータを送信する前に 2 次サーバーが障害を起こした場合、1 次側はトランザクションをロールバックします。
3. 2 次側は、コミット・メッセージの確認応答を行います。確認応答のタイミングは、**HotStandby.2SafeAckPolicy** 構成パラメーターの設定によって異なります。*2-safe received* という最高速の選択肢では、2 次側はコミット・メッセージの受信直後に 1 次側に確認応答を送信します。*2-safe durable* という最も安全な選択肢では、2 次側はトランザクションを実行して 2 次側自体のトランザクション・ログに永続的に書き込んだ後に、確認応答を送信します。

   1 次側は、2 次側の確認応答を受信すると、ユーザーにデータがコミットされたことを通知します。
4. 1 次側が 2 次側からの確認応答を (例えば、ネットワークやノードの障害のために) 受信できない場合、1 次側はタイムアウトになり、PRIMARY UNCERTAIN 状態に切り替わります。1 次側は、2 次側での最新のトランザクションの状態が分からないので、トランザクション自体のロールバックやコミットができません。 1 次側には、以下のどれが起きたかが分かりません。
   - トランザクションがコミットされる前に 2 次側がダウンした。

- 2 次側は、既にトランザクションをコミットしたが、1 次サーバーは例えばネットワーク障害のために、確認応答を受信しなかった。

サーバーが PRIMARY UNCERTAIN 状態にある間、ユーザーがコミットを試みる現行トランザクションと新規トランザクションはブロックされ、ユーザーにはサーバーが無応答のように感じられる場合があります。

5. HAC は、2 次側がダウンしていること、またはネットワークが障害を起こしたことを検出した場合、1 次サーバーを PRIMARY ALONE 状態に切り替えることができます。 1 次サーバーは、PRIMARY ALONE に設定されるとすぐに、2 次側へ送信した保留中のトランザクションをコミットし、新しいトランザクションの受け入れを再開します。
6. 変更は、2 次サーバーが復帰するか 1 次サーバーのディスク・スペースがなくなるまで、トランザクション・ログ・ファイルに蓄積されます。サーバーがトランザクション・ログのディスク・スペース不足になった場合、1 次側は読み取り専用モードに変わります。
7. 2 次サーバーが長時間にわたって稼働せず、サーバーがトランザクション・ログのディスク・スペース不足になる可能性がある場合は、1 次サーバーを PRIMARY ALONE から STANDALONE の状態に切り替えるとよいでしょう。そのような場合には、トランザクション・ログは、2 次側との接続が失われた時点以降のすべてのトランザクションを保管しなくなり、したがって、単に 1 次側からトランザクション・ログを読み取るだけでは 2 次側がキャッチアップすることができなくなります。トランザクション・ログによって 2 次側を最新の状態にできない場合、2 次側と 1 次側の同期をとる唯一の方法は、1 次側のデータベース・ファイルを 2 次側にコピーすることです。これは、**hotstandby netcopy** コマンドによって行うことができます。
8. **hotstandby netcopy** を実行するには、1 次側が PRIMARY ALONE 状態でなければなりません。**hotstandby netcopy** の後、1 次サーバーはコマンドが成功したか失敗したかに関係なく、PRIMARY ALONE 状態のままです。
9. 1 次側が 2 次側へのトランザクションの送信を開始するためには、1 次サーバーを 2 次サーバーへ明示的に接続するために、**hotstandby connect** コマンドを使用する必要があります。 1 次サーバーが 2 次サーバーに接続した後、1 次サーバーは PRIMARY ACTIVE 状態で動作します。

   サーバー同士が接続した後、キャッチアップ (1 次サーバーとの同期をとるために、保留中のすべての変更がトランザクション・ログから 2 次サーバーへ自動的に書き込まれること) の実行が開始されます。サーバー・キャッチアップの前に、1 次サーバーと 2 次サーバーは情報を交換し、トランザクションが 2 次サーバー上で 2 回コミットされないよう、キャッチアップの開始位置を決定します。

関連タスク

60 ページの『ネットワークを介した 1 次データベースの 2 次側へのコピー』
データベース・ファイルのコピーを 1 次サーバーから 2 次サーバーへ送信するには、**netcopy** コマンドを使用します。2 次サーバーは、既に稼働している必要があります。

64 ページの『1 次サーバーから指定されたディレクトリーへのデータベース・ファイルのコピー』
2 次側のデータベースに使用されるディレクトリーが 1 次側に可視の場合は、**hotstandby copy** コマンドを使用してデータベースを 1 次側のディレクトリーから 2 次側のディレクトリーにコピーできます。

65 ページの『3.4.6, HotStandby サーバーの接続』

1-safe レプリケーションでの非同期 HotStandby

オプションとして、1 次側から 2 次側への非同期レプリケーションを使用できます。これは、*1-safe レプリケーション* と呼ばれます。

1-safe レプリケーションでは、トランザクションが 1 次側でコミットされると、直ちにその確認応答が行われます。これにより、パフォーマンスが大幅に向上します。コミットの後、トランザクションは 2 次側へ非同期方式で送信されます。そのトレードオフとして、1 次側で障害が発生すると、転送中だった少数のトランザクションが失われる可能性があります。

2 つのレプリケーション方式のどちらも、動的に選択でき、セッションごと、またはトランザクションごとに選択することさえできます。1-safe レプリケーションに関連するレプリケーション遅延も、制御することができます。

1.1.5 持続性とロギング

持続性レベルは、solidDB がトランザクション・ロギングを処理する方法を制御します。 solidDB サーバーは、ストリクト (安全)、リラックス (高速)、およびアダプティブの 3 つの持続性レベルをサポートしています。リラックス持続性では最適なパフォーマンスが得られるのに対し、ストリクト持続性ではトランザクションの損失が最小限に抑えられます。アダプティブ持続性レベルは、HotStandby 構成でのみ使用可能です。

- *ストリクト持続性*: トランザクションがコミットされると同時に、そのトランザクションがトランザクション・ログに書き込まれる場合、それを「ストリクト持続性」と呼びます。このタイプの持続性は、安全性を最大にします。
- *リラックス持続性*: トランザクションの書き込みを、サーバーがビジーでなくなるまで、または複数のトランザクションをまとめて書き込めるまで遅延できる場合、それを「リラックス持続性」(または「リラックス・ロギング」) と呼びます。HSB ペアに含まれていないサーバーでは、リラックス持続性を使用することは、サーバーが異常終了した場合、少数の最新のトランザクションを失う危険を冒すことを意味します。しかし、サーバーが HSB ペアに含まれている場合は、トランザクションのコピーがもう一方のサーバー (2 次サーバー) にあり、しかも、1 次サーバーがトランザクションをログに記録する前に障害を起こしたとしても、トランザクションは失われません。このため、リラックス持続性を HSB と一緒に使用した場合、リラックス持続性が原因で安全性が低下することは、ま

ずありません。逆に、リラックス持続性は、特にサーバーの負荷が多数の小さな書き込みトランザクションからなっている状況では、システムのパフォーマンスを向上させます。

- *アダプティブ持続性*: アダプティブ持続性は、HotStandby 1 次サーバーにのみ適用されます。アダプティブ持続性とは、単にサーバーが Primary Active 状態 (2 次サーバーへトランザクションを送信中) の場合はリラックス持続性を使用し、それ以外の状態ではストリクト持続性を使用することを意味します。これにより、HSB がアクティブのときは高いパフォーマンスを (安全性をほとんど失うことなく) 得ることができ、一方のサーバーのみが動作する場合でも、高い安全性が維持されます。アダプティブ持続性は、2-Safe レプリケーション (デフォルト) が使用されているときにのみ、効果的に機能を発揮できます。

アダプティブ持続性は、パフォーマンスを大幅に向上させる一方、障害のある状態でも高いデータの安全性を保証します。これはシステム全体のスループットを増し、待ち時間 (つまり、トランザクションがコミットされたことを通知されるまでユーザーが待たなければならない時間) を減らすことができます。

持続性レベルは、**Logging.DurabilityLevel** パラメーターを使用してサーバーのデフォルトとして設定するか、SET [TRANSACTION] DURABILITY ステートメントを使用してセッション単位またはトランザクション単位で設定できます。

レプリケーション・プロトコルに関連して説明すると、STRICT は 2-safe に対応し、RELAXED は 1-safe に対応します。

1-Safe レプリケーション

1-safe レプリケーションでは、1 次サーバーでコミット処理が完了した直後に、コミット・ステートメントに対する確認応答が行われます。コミットされたトランザクションは、アプリケーションへ制御が戻された後に、2 次サーバーへ非同期に伝送されます。

トランザクションの伝送にかかわる遅延は、数ミリ秒から数百ミリ秒の範囲です。1-safe レプリケーションを使用すると 1 次サーバーでの待ち時間が著しく減少するため、パフォーマンスが大幅に向上します。1-safe の不利な面は、障害が起きたとき、フェイルオーバーの際に少数のトランザクションが失われる可能性があることです。

1-safe レプリケーションは、サーバーに対してパラメーター **HotStandby.SafenessLevel=1safe** と一緒に設定できます (**HotStandby.SafenessLevel** パラメーターの有効な値は、1safe、2safe、および auto です。デフォルトは 2safe です)。

安全性のレベルは、SET コマンドで動的に制御することもできます。

```
SET SAFENESS {1SAFE| 2SAFE| DEFAULT}
```

SET SAFENESS は、現行セッションの安全性レベルを、変更されるまでの間、設定します。

```
SET TRANSACTION SAFENESS {1SAFE| 2SAFE| DEFAULT}
```

SET TRANSACTION SAFENESS は、現行トランザクションの安全性レベルを設定します。コミットの後、安全性レベルはセッションについて設定された値か開始値、またはシステム・デフォルト (これは 2-safe です) に戻ります。

オプション DEFAULT は、セッションの現行の設定を意味します。

HotStandby.SafenessLevel パラメーターを auto (つまり「自動」) に設定した場合は、プログラムによる持続性の制御 (例えば、SET DURABILITY RELAXED) を使用して安全性レベルを制御できます。

2-safe 確認応答ポリシー

2-safe レプリケーションが有効 (デフォルト) の場合、1 次サーバーは 2 次サーバーからトランザクションを取得したことの確認応答を受信するまで、トランザクションが正常にコミットされたことをクライアントに通知しません。

確認応答ポリシーは 3 つあります。

- 2-safe received: 2 次サーバーは、データを受信したときに確認応答を送信します (デフォルト)。
- 2-safe visible: 2 次サーバーはデータ・コピーの更新を完了し、変更が「可視」になりました。言い換えると、2 次サーバーに接続しているクライアント・アプリケーションは、更新を認識することができます。
- 2-safe durable: 2 次サーバーはデータを永続的にした時点で、つまり、データのコミットとディスクへのデータの書き込みを完了した時点で、確認応答を行います。

2-safe 確認応答ポリシーの選択

2-safe received は、より高速です。2-safe durable は、より安全です。これらの確認応答ポリシーは 1 次と 2 次のサーバーが両方ともアクティブである (つまり、両方ともトランザクションを適用している) ときにのみ適用されるので、2-safe received でも安全であると考えられます。両方のサーバーが事実上同時に (互いに 1 秒以内に) 障害を起こした場合にのみ、トランザクションを失う危険があります。

2-safe received を使用すると、待ち時間 (コミットが開始されてから、ユーザーがコミットの確認を受け取るまでの時間) が少なくなります。2-safe received ポリシーは、全体のスループットにほとんど影響しません。

1.1.6 HotStandby と SMA

SMA サーバー・ノードは、solidDB ホット・スタンバイ・コンポーネントを使用することにより、高可用性を実現することができます。

ホット・スタンバイ・セットアップを使用する SMA では、各ノードで 1 つ以上の SMA アプリケーションを使用することができます。アプリケーションからデータベースへの接続は、通常の SMA 接続 (SMA 基礎接続) として、または透過接続 SMA 接続 (SMA TC) として構成することができます。いずれの接続タイプを使用しても、 1 次ノード上のアプリケーションは SMA 接続を使用してローカルに読み取りと書き込みを実行し、2 次ノード上のアプリケーションは SMA 接続を使用してローカルに読み取りを実行します。また、 SMA TC 接続を使用した場合は、2 次サーバー上のアプリケーションからの書き込みトランザクションを、ネットワー

ク接続を使用して 1 次サーバー上で実行することができます。さらに、ロード・バランシング・オプションが SMA TC 接続で使用可能な場合、アプリケーションはアクティブ - アクティブ方式で動作することができます。すなわち、各ノード上で、データベース・アクセスの全機能が使用可能です。

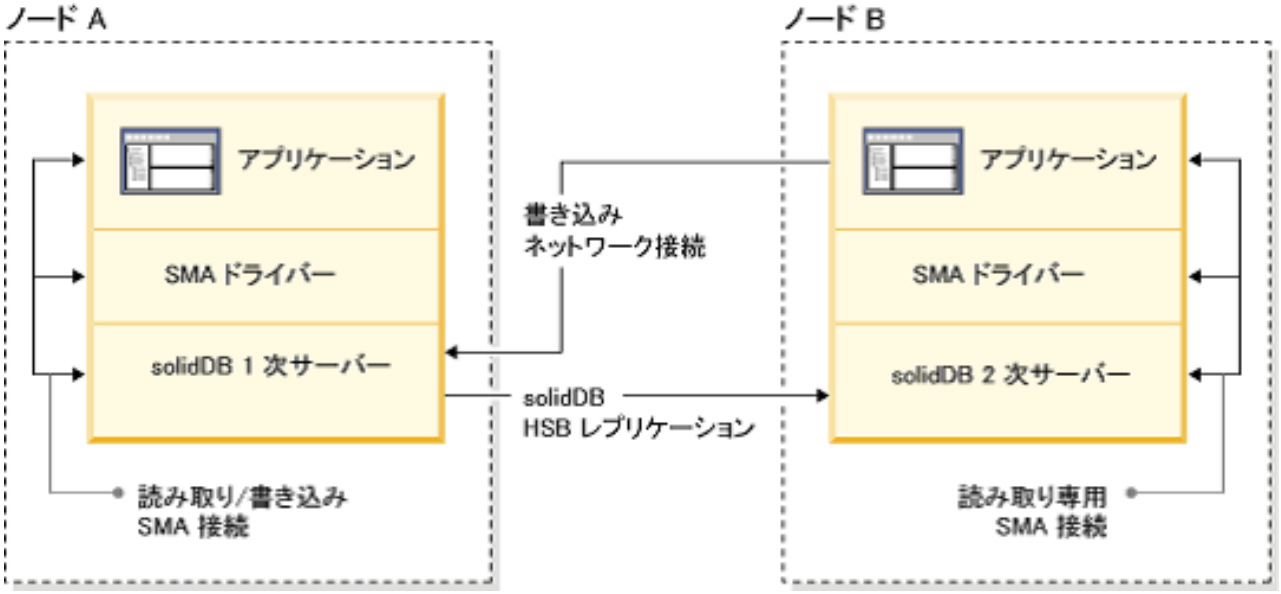


図 6. ホット・スタンバイを使用する SMA 透過接続性のアーキテクチャー

SMA TC を使用する場合、各ノード上のアプリケーションは SMA 接続を使用してローカル・サーバーに接続可能になっている必要があります。また、リモート・サーバーへは、ネットワーク・ベースの接続を使用して接続可能になっている必要があります。

フェイルオーバーおよび切り替えの処理

- いずれか 1 つのサーバーの状態が PRIMARY ACTIVE、PRIMARY ALONE または STANDALONE であれば、接続ハンドルは切り替えおよびフェイルオーバーによって維持されます。
- SMA サーバーに障害が発生すると、アプリケーションも失敗します。こうした障害のシナリオにおいて高可用性を保つには、システムにアプリケーション・レベルのフェイルオーバー・メカニズムを組み込む必要があります。これは、アプリケーションが提供するサービスを、障害が発生したアプリケーション・インスタンスから他のアプリケーション・インスタンスへ移動するメカニズムです。

1.1.7 HotStandby と拡張レプリケーション

solidDB HotStandby コンポーネントは、solidDB 拡張レプリケーションと組み合わせて使用できます。拡張レプリケーションは、双方向の定期的に発生するデータ同期機能を提供し、これによりユーザーは、マスター・サーバーとレプリカ・サーバーが含まれている分散システムを作成できます。 HotStandby を使用すると、分散システムのあらゆるデータベース・サーバーの可用性を高めることができます。

19 ページの図 7 は、1 つのマスター・データベースと 2 つのレプリカ・データベースを含んでいる単純な分散システムを示しています。各レプリカには、少なくと

もマスター・データベースのデータのサブセットが格納されています。それぞれのデータベース・サーバーは、HotStandby レプリケーションによってフォールト・トレラントになっています。拡張レプリケーションは、データベース・サーバー階層の 1 次サーバー間で発生します。いずれかの 1 次データベース・サーバーで問題が起きた場合、障害を起こしたノードは HotStandby フェイルオーバーを実行して、そのノードの 2 次サーバーを新しい 1 次サーバーにすることができます。これにより、新しい 1 次サーバーを使用して拡張レプリケーションを続行できるようになります。

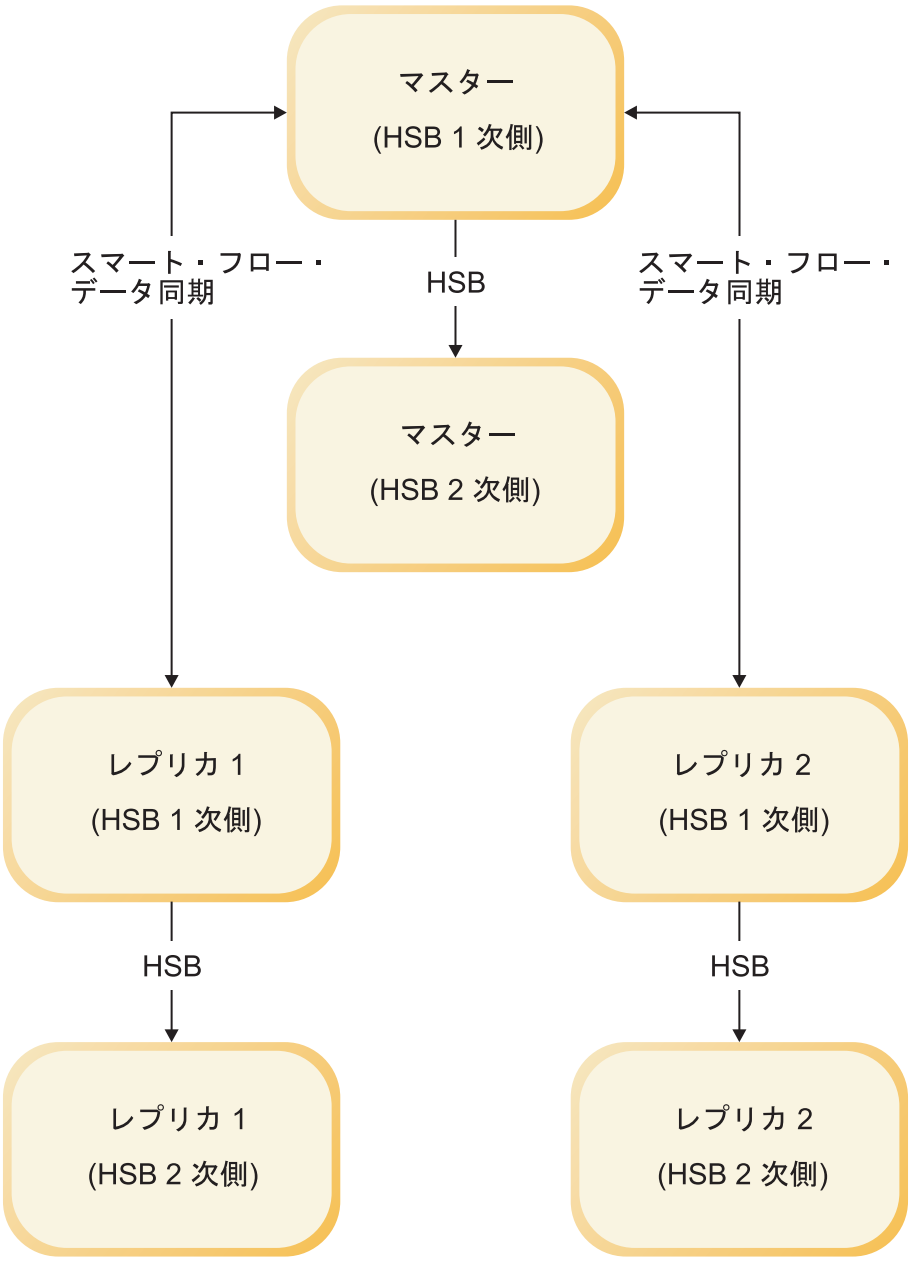


図 7. マスターおよびレプリカ・サーバー・スキームでの HotStandby

1.2 パフォーマンスと HotStandby

HotStandby システムのパフォーマンスを最適化することを考える場合は、パフォーマンスには複数の面があり、それらを検討することが必要になります。HA (冗長) システムでは、変更したデータを信頼性および一貫性のある方法で別のノードに複製する処理の影響が重要な役割を果たします。

このセクションでは、HotStandby 固有のパフォーマンスの考慮事項について説明します。ただし、以下で説明するトピックに加えて、パフォーマンスの最適化の取り組みでは、主に SQL 照会とスキーマ設計の最適化に焦点を当てるべきです。なぜなら、パフォーマンスの低下は多くの場合、SQL 照会、索引などを修正することによって改善できるからです。実際のスループットと応答時間は多数の要因に依存しています。例えば、それらの要因には、ネットワークの速度、ネットワーク上の他のトラフィックの量、SQL ステートメントの複雑さ、および 1 トランザクション当たりの SQL ステートメント数が含まれますが、それだけに限りません。例えば、メモリー量とディスク速度などの一般的な環境要因もパフォーマンスに影響します。

HotStandby のセットアップで考慮すべき主要なパフォーマンス要素は、次のとおりです。

- 待ち時間または応答時間 - 1 回の読み取り操作または書き込み操作はどのくらいの速さで完了しますか。
- スループット - 2 ノード・システムで処理できる照会またはトランザクションの容量は全体でどのくらいですか。
- データの安全性 - 各トランザクションが同じノード上で (ディスクに) または次のノードに (ネットワーク経由で) 安全に持続されることがシステムで保証されていますか。
- フェイルオーバー時間 - 単一ノードの障害後に、エラー検出時間を含め、システムはどのくらいの速さでそのサービスの提供を続行することができますか。
- リカバリー時間 - 障害が解決された後、システムはどのくらいの速さで、どの程度自動的に、HA 状態にリカバリーしますか。

solidDB HotStandby のセットアップでは、以下の構成およびセットアップの各オプションを使用して HotStandby のパフォーマンスを最適化できます。

- 単一の障害でトランザクションを保存する必要がある場合には、アダプティブ持続性が有効です。
- 障害での小規模なトランザクションの損失が許容可能な場合は、1-Safe レプリケーション・プロトコルが有効です。
- 2-Safe レプリケーション・プロトコルおよび適切な 2-Safe 確認応答ポリシーでは、最大限の安全性が確保されます。
- 2 次サーバーで読み取り専用トランザクションを実行することによるロード・バランシング。

  solidDB HotStandby では、クライアントは 2 次サーバーに接続して読み取り専用操作を行うことができます。状況によっては、読み取り専用クライアントを 2 次サーバーに接続して、そこで読み取りを実行させることによって「負荷を分散」し、システム全体のパフォーマンスを向上させることができます。これは特に、

多数のレコードを読み取る必要があり、レコードを変更する必要がない、報告書作成や「データウェアハウジング」の照会などの作業に便利です。

- 内部並列処理

  システムで並列処理の利点を活用したい場合は、すべてのトランザクションを同じ接続によってサブミットするのでなく、複数の接続にトランザクションを分散することを考慮してください。

  HotStandby (HSB) コンポーネントを使用した場合、書き込み操作を含んでいるすべてのトランザクションは、2 回 (1 回は 1 次サーバー上で、もう 1 回は 2 次サーバー上で) 実行されます。このため、状況によっては単一のトランザクションが、HSB を使用した場合に、HSB を使用しない場合の約 2 倍の時間を要することがあります。しかし、これは全体のスループットが 50% 低下することを意味するわけではありません。各サーバーの並列処理の度合いは高く、2 次サーバーが 1 つのトランザクションを処理している間に、1 次サーバーは別のトランザクションを処理します。

  注: 並行して実行する照会の数が多いほどサーバーに必要なメモリーの量が増えるため、接続を追加して照会を並列に実行することが常にスループットを高めるわけではありません (特に、大量のメモリーを備えていないシステムの場合)。一度に実行する照会の最適の数を知るために、実際に試行が必要になる場合もあります。

要約すると、可能な最高の安全性レベルを必要とする場合を除いて、以下を行うことにより、パフォーマンスを高めることができます。

- **Logging.DurabilityLevel** を 2 に設定してアダプティブ・ロギングを使用する。
- **HotStandby.2SafeAckPolicy** を 1 に設定して 2-safe received モードを使用する。

安全性が低い設定 (アダプティブ持続性および 2-safe received モード) を使用する場合でも、障害が 2 つ以上にならない限り (例えば、両方のサーバーがほぼ同時にダウンしない限り)、HotStandby によって保護されます。最低でも、それぞれのサーバーを別々の無停電電源装置 (UPS) に接続し、電源障害から保護してください。さらに、どのデータベース・システムでも同じですが、重要なデータはバックアップを作成し、なるべく別の場所に保存してください。HotStandby は、データのバックアップの代わりにはなりません。

ヒント: バックアップは、HSB ペアのどちらのサーバー上でも (**ADMIN COMMAND backup** コマンドを使用して) 実行できます。多くの場合、バックアップの作成に使用可能なリソースをより多く備えているのは、2 次サーバーです。

1.3 高可用性コントローラー (HAC)

高可用性コントローラー (HAC) は、solidDB HotStandby 構成のための自動冗長性管理プログラムです。このプログラムはデータベース・サービスの可用性を維持するために、障害を検出し、スタンバイ装置へのフェイルオーバーを行い、必要なときは障害を起こしたプロセスを再開します。

障害の原因としては、データベース・ノードのハードウェアの問題、データベース・プロセスの失敗、または HSB リンクの切断が考えられます。HAC はサーバーの HSB 状態をモニターし、障害が起きた場合には障害の影響を受けていないサーバーが 1 次サーバーのロールを保持し、そのサーバーがトランザクションの負荷をいつでも受け入れられるようにします。

言い換えれば、HAC は Watchdog プログラムの役割を果たします。その実装では、サーバー状態をモニターするために、solidDB のイベント・メカニズムが使用されます。HSB サーバーの状態が変化すると必ず、そのサーバーから HAC にイベントが送信され、HAC は、HotStandby 管理コマンド API を使用して実行するアクションの潜在的な必要性を推定します。

HAC のアーキテクチャーを下の図に示します。

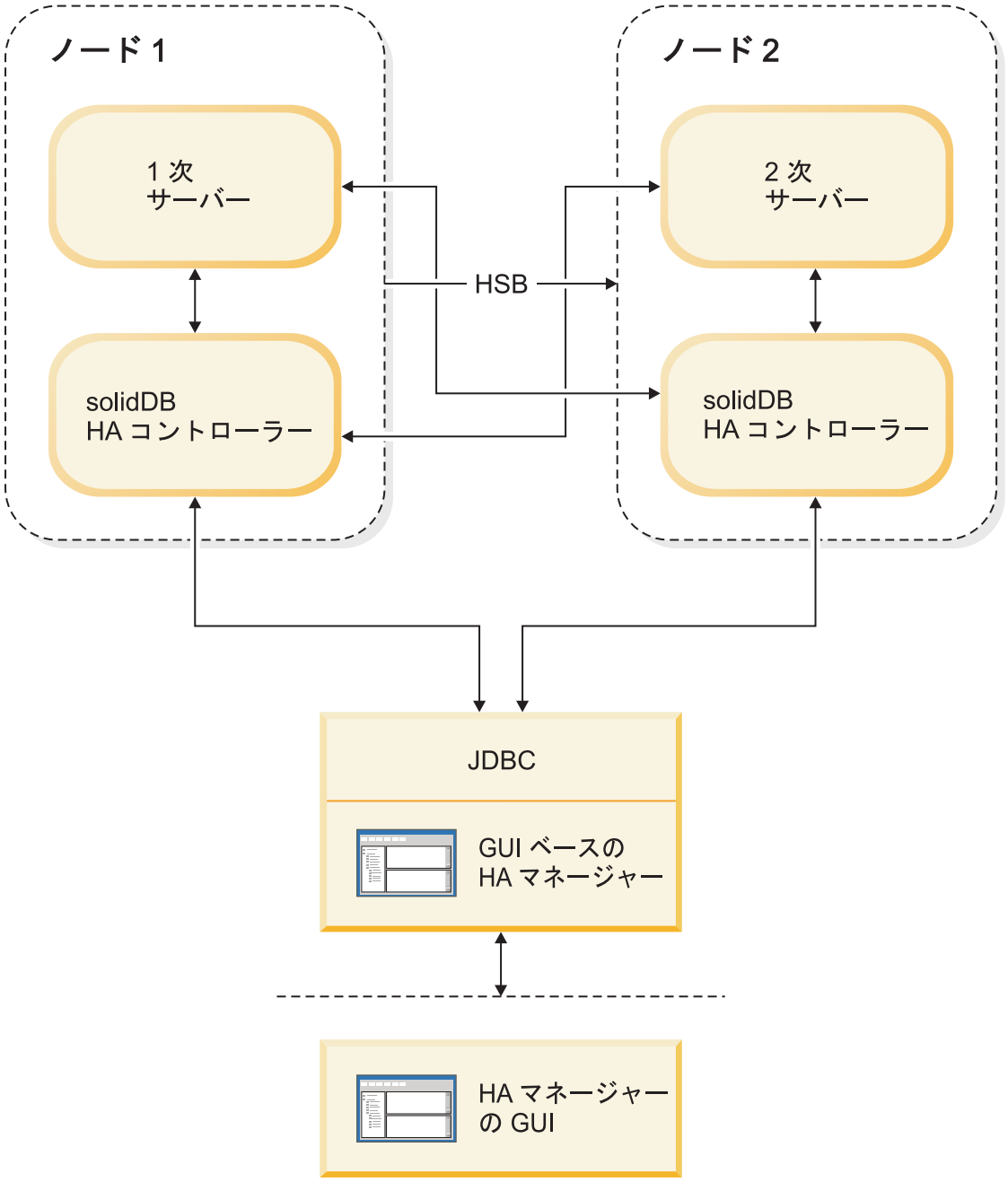


図 8. 高可用性コントローラーのアーキテクチャー

HAC には、主に以下の 2 つの目的があります。

1. HAC を Watchdog として使用して、データベース・サービスの可用性を自動的に維持することができます。この AUTOMATIC モードと呼ばれるモードでは、HAC は以下のアクションを実行します。
   - データベース・サーバー・プロセスを開始し、再開し、終了します (オプション)。
   - 各サーバーとそれらの間の HSB リンクの状態をモニターします。
   - 実行する必要があるアクションを推定します。
   - アクションを実行します。

2. HAC は、HSB システムのモニターの役割も果たし、(別の誰かによって起動された) HSB サーバーの状態変更をモニターし、システムの状況を報告します。これは、ADMINISTRATIVE モードと呼ばれます。このモードでは、HAC は HSB の状態遷移を実行せず、それ以外の HSB システムの変更も行いません。

高可用性コントローラーは、solidhac.ini 構成ファイルを通じて構成されます。HAC を開始する前に、このファイルを HAC の作業ディレクトリーに配置してください。作業ディレクトリーは、コマンド行オプション -c で指定できます。

solidDB パッケージには HAC 構成ファイル・テンプレートが含まれており、それに、使用可能なすべての構成パラメーター (コメント付き) と例が組み込まれています。solidhac_template.ini は、samples¥hac¥ ディレクトリーのルートに置かれています。

HAC では、高可用性マネージャー (HAM) と呼ばれるサンプルのグラフィカル・ユーザー・インターフェース (GUI) コンポーネントも使用できます。これは、サンプル・ディレクトリー samples¥hac¥ に含まれています。HA マネージャーは HA コントローラーと同じように構成されます。samples¥hac¥ ディレクトリーに、高可用性マネージャーのサンプル構成ファイル (HAManager.ini) が含まれています。

1.3.1 認識される障害

HAC は、HotStandby サーバーの正常性と状況をモニターします。データベース・プロセスの失敗やコンピューター・ノードの障害など、障害状態においては、HAC はフェイルオーバーやその他の必要な状態遷移を実行して、データベース・サービスの使用可能な状態をできる限り維持します。

考慮されるすべての障害について、それらの障害は、2 つの HSB サーバーの PRIMARY ACTIVE 状態および SECONDARY ACTIVE 状態によって表現される通常の完全な作動可能状態で発生することが想定されています。HAC は、単一の障害のみに対処します。言い換えれば、システムが前回の障害からリカバリーする前に障害が発生しないことを前提としています。ただし、HAC で処理できるいくつかの複数障害のシナリオが事前定義されています。

単一の障害に関する限り、HAC はデータベース・サービスをほとんど中断することなく維持します。複数の障害が発生した場合、HAC は、エラーのあるシステム状態 (二重 1 次サーバーなど) を回避しようと試みます。

HAC が処理できる障害は、以下のとおりです。

- 単一の障害
  - 1 次 (ACTIVE) データベース・サーバー・プロセスの失敗
  - 2 次 (ACTIVE) データベース・サーバー・プロセスの失敗
  - 1 次ノードの障害
  - 2 次ノードの障害
  - 外部参照エンティティーが使用されている場合、HAC は HotStandby リンクの障害、つまり 2 つの HotStandby データベース・プロセスの間で失われた接続も処理できます。外部参照エンティティーの詳細については、25 ページの『1.3.3, 外部参照エンティティー (ERE)』を参照してください。
  - サーバーが外部クライアントに応答しない

- 二重障害
  - 前回の障害からリカバリーする間、HAC は 1 次側と 2 次側のデータベース間の同期に関するエラーを認識します。
  - また、HAC は、もっと一般的でない障害にも対処します。例えば、サーバーが前回の障害の後、HSB リンクを確立しようとしているときに起きたサーバー・プロセスの失敗などです。

障害とリカバリーのシナリオについて、詳しくは、 115 ページの『5, 高可用性コントローラー (HAC) での障害処理』を参照してください。

1.3.2 データベース・サーバー・プロセスの制御

HAC は、データベース・プロセスを開始し、失敗したプロセスを再開するよう構成することができます。

HAC インスタンスは、ローカル・データベース・サーバーとの接続を失うと、`solidhac.ini` 構成ファイル内で指定された開始スクリプトを呼び出します。このスクリプトは、ユーザーが提供します。スクリプトの例はパッケージに添付されています。

重要: HAC インスタンスは、開始スクリプトが終了する時点でサーバーが稼働しており、対応できるものと想定します。HAC はサーバーの始動の際に発生する障害を処理しないため、スクリプトはサーバーが接続を受け入れた場合以外、終了してはなりません。

データベース・サーバーが障害を起こしたか、それ以外の何らかの理由で消失した場合、HAC はデータベース・サーバーを再始動しますが、以下の 2 つの場合は例外です。

- データベース・プロセスがオペレーティング・システムのプロセス・リストから消失していない場合、または
- データベース・サーバーが HA マネージャーを使用してシャットダウンされた場合。

1.3.3 外部参照エンティティー (ERE)

処理を必要とする最も難しい障害状態の 1 つは、データベース・ノード間の通信リンクに障害が起き、どちらのデータベース・サーバーも相手側がダウンしたと想定する可能性がある状態です。この状態は、二重 1 次サーバー (「*分離脳*」) 状態につながる場合があり、後でデータベースの同期がとられたときに、トランザクションを失うおそれがあります。できるだけ solidDB HAC による誤った判断を避けるために、外部参照エンティティー (ERE、つまりネットワーク参照デバイス) を使用して、ネットワークの正常性を検査することが推奨されます。例えば、1 台のコンピューターでネットワーク・アダプターの障害が起きた場合、HAC はその状態を検出でき、正しいデータベース・ノードが 1 次側のデータベースとして (また、もう一方が 2 次側として) 機能を続行するよう設定することができます。

ERE を使用した場合、HAC は HotStandby ノードと ERE デバイスの間の物理リンクの状況を検査するために、ERE に対して ping を行います。最も近い ERE への物理リンクが動作していない場合、ローカル HAC はローカル・サーバーを SECONDARY ALONE 状態に設定します。最も近い物理リンクが動作しており、他

のサーバーへの接続が使用可能でなければ、ローカル HAC はローカル・サーバーがサービスの提供を続行するサーバーであると判定して、ローカル・サーバーを PRIMARY ALONE に設定します。したがって、相手側の HotStandby ノードへの接続と、最も近い ERE への接続を失っている HotStandby ノードは、2 次サーバーになります。このようにして、ネットワーク障害が起きた場合でも、2 つの 1 次サーバー (分離脳) の状態が防止されます。

ERE 用の HAC の詳しい構成方法については、47 ページの『3.3, HA コントローラーおよび HA マネージャーの構成』を参照してください。

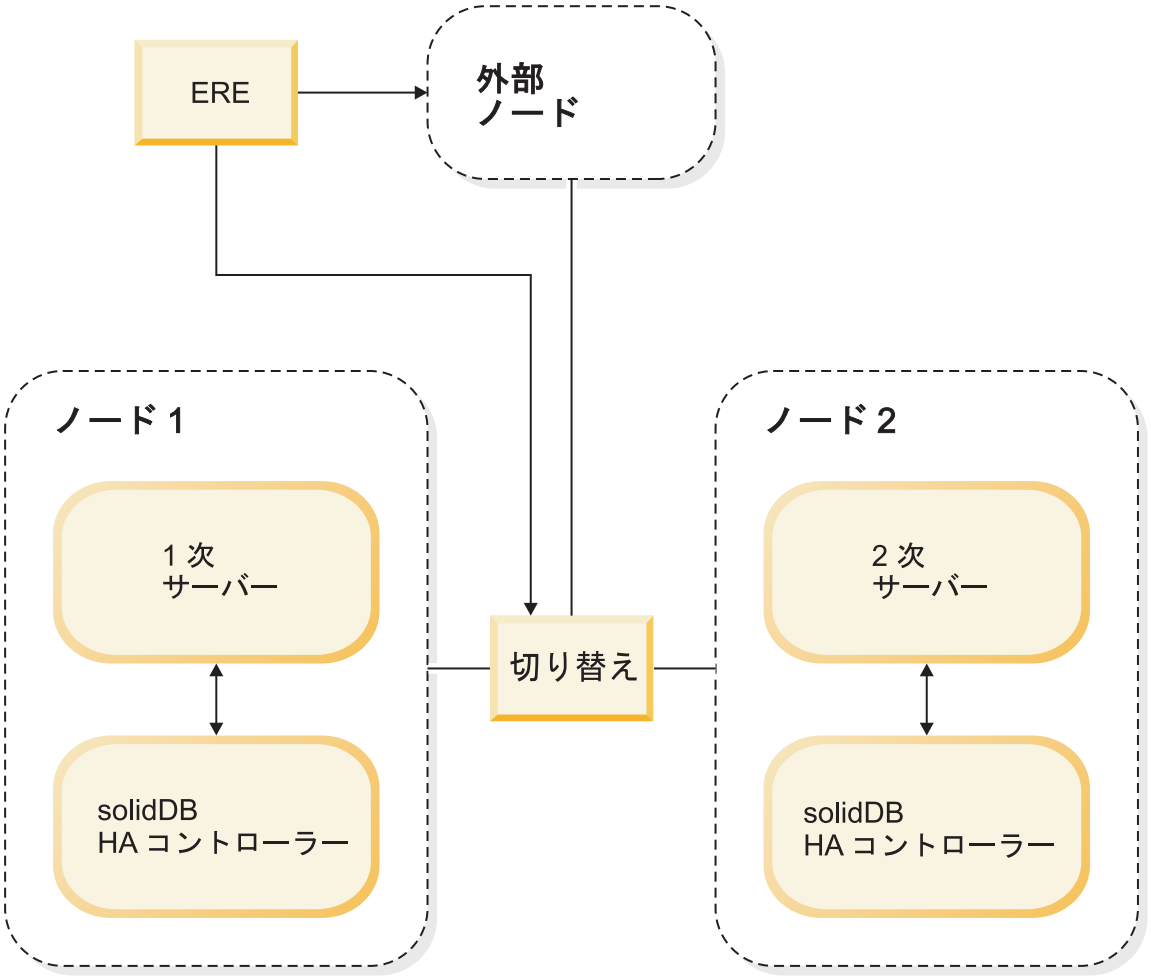


図 9. 外部参照エンティティーのコンポーネント

上の図は、ERE に可能な以下の 2 つのロケーションを示しています。

- クラスター・スイッチ
- クラスターの外部にある、ネットワーク内の任意のコンピューター。クラスター内で冗長ネットワーク (つまり、重複するネットワーク・コントローラー、ケーブルおよびスイッチ) を使用している場合は、ERE をクラスターの外部に定義することをお勧めします。

重要: HotStandby リンクが信頼できないと考えられる場合は (ERE を使用しているすべてのケースも含めて)、以下の HotStandby サーバー・パラメーターをファクトリー値に設定する必要があります。

HotStandby.AutoPrimaryAlone=no

ERE は、キープアライブ・メッセージが使用するのと同じ HSB リンクを使用する必要があります。

1.3.4 HAC でのネットワーキング

それぞれの HSB サーバーで、すべての通信に単一の論理ネットワーク・アクセス (IP アドレス) を使用することをお勧めします。

この推奨事項は、ネットワーク・インターフェース API レベルで透過的な、複数の (または予備の) 下位ネットワーク・コンポーネント (ネットワーク・インターフェース・カード、ケーブル、およびスイッチ) の使用を排除するものではありません。

サーバーは、さまざまなポート番号をさまざま目的に使用できます。

1 つの論理ネットワーク・アクセスだけを使用した場合、HAC (ERE を伴う) は HSB トランザクション・レプリケーションとデータベース・クライアント・アプリケーションの両方に影響を及ぼすネットワークの切断を検出します。別々のインターフェースを使用した場合、HAC は HSB リンクの正常性のみをモニターするため、データベース・クライアント通信での障害は HAC によって検出されません。

ただし、どのような理由であれ、ネットワーク・アクセスが信頼できないと考えられる場合は、下層にあるネットワーク・アクセスの実装に関係なく、HAC の ERE 機能を使用できます。

1.3.5 HAC ロギング

HAC は、ログ・レコードを HAC 作業ディレクトリーにある hacmsg.out ファイルに書き込みます。

ログには、以下に関する情報が入っています。

- 警告
- 致命的エラー
- 構成に関する情報
- 初期化に関する情報
- すべての入力イベント
- HotStandby の状態変更
- システム内の変更の原因となるユーザー・コマンド
- HAC の状態変更
- HAC のモード変更 (AUTOMATIC/ADMINISTRATIVE)
- システムの状態変更の原因となるイベント

HAC ログ・ファイルの最大サイズは 64 メガバイトです。サイズが限度を超えると、hacmsg.out は hacmsg.bak という名前に変更され、新しい hacmsg.out が作成されます。これらのファイルには、最大 128 メガバイトの最新のログが格納されます。

1.4 高可用性マネージャー (サンプル)

solidDB パッケージには、高可用性マネージャー (HAM) と呼ばれるサンプル・プログラムが含まれています。HAM は、HotStandby サーバーの状態と HAC の状態を表示する Java ベースのグラフィカル・インターフェース・ツールです。これはまた、例えば、HotStandby サーバーのロールの切り替えや HAC の一時中断と再開など、HAC を管理するための基本機能を提供します。

以下の図は、HAM インターフェースを示しています。

図 10. 高可用性マネージャー

高可用性マネージャーは、HAManager.ini 構成ファイルを通じて構成されます。

高可用性マネージャーを使用して、以下のアクションを実行できます。

- HotStandby サーバー間での切り替え
- AUTOMATIC と ADMINISTRATIVE の HAC モードの切り替え
  - ADMINISTRATIVE モードでは、HAC は HSB クラスターのみをモニターし、ユーザーは HSB サーバー上で管理タスクを実行できます。

  - AUTOMATIC モードでは、HAC は Watchdog として機能し、障害を処理し、データベース・サービスの可用性を維持します。
- HotStandby データベース・サーバー・プロセスのシャットダウン
- HotStandby データベース・サーバー・プロセスの開始
- 高可用性コントローラーの一時中断
- 高可用性コントローラーの再開

注:

データベース・サーバー・プロセスのシャットダウンと開始を実行できるのは、**HAController.EnableDBProcessControl** パラメーターが `solidhac.ini` ファイルの中で `Yes` に設定されている場合だけです。

2 HotStandby の概要

このセクションでは、2 つの solidDB HotStandby サーバー (1 次サーバーと 2 次サーバー) をセットアップする段階的な手順を示します。

このセクションでは、1 台または 2 台のコンピューターに solidDB が既にインストールされていることを前提とします。

1 台のコンピューターに HotStandby の評価構成をセットアップすることができます。1 台のマシンに solidDB の 2 つのインスタンスを稼働させ、一方のインスタンスを 1 次サーバーとして、もう一方のインスタンスを 2 次サーバーとしてセットアップできます。

2.1 HotStandby クイック・スタート手順

このセクションでは、HotStandby のクイック・スタート手順について説明します。この手順を実行すると、HSB システムでアプリケーションのサービスを開始できる状態に到達します。

このタスクについて

この方式では、HSB サーバー以外の solidDB コンポーネントを想定していません。例えば、HAC は必要ありません。HAC についての同様な段階的な手順は、34 ページの『2.2.1, HA コントローラーの開始と停止』で説明されています。また、サンプルの Watchdog アプリケーションが solidDB HotStandby パッケージに含まれています。サンプルの Watchdog を使用するには、そのための構成設定を行う必要があります。

HSB サーバー・ペアをセットアップして実行するには、ネットワークで接続した 2 台のコンピューターが必要です。HotStandby サーバーを (他の solidDB コンポーネントなしに) セットアップするには、以下の手順を実行します。

手順

1. 1 次ノードと 2 次ノードを構成します。

   HotStandby を使用するには、最小でも 1 次サーバーと 2 次サーバーの両方で `solid.ini` 構成ファイルの `[HotStandby]` セクションに以下のパラメーターを構成する必要があります。

   - **`HSBEnabled`**`=Yes`

     `HSBEnabled` パラメーターを省略するか、HotStandby 用に予定しているサーバーに `no` の値がある場合、サーバーは始動したときに非 HotStandby サーバーになります。

   - **`Connect`**`=`*connect string for the opposite HSB server*

Connect パラメーターを省略した場合、サーバーは HotStandby サーバーとして始動しますが、サーバーを接続するためには、事前に ADMIN COMMAND を使用して接続ストリングを指定する必要があります。

2. 他の solidDB サーバーと同じ方法で、両方の HSB サーバーを始動します。

   サーバーは、それぞれの独自の `solid.ini` ファイルから HotStandby 構成情報を読み取ります。始動後の両方のサーバーの状態は、SECONDARY ALONE です。

3. 1 次サーバーにするサーバーを選択し、以下のコマンドを発行することにより、選択したサーバーの状態を PRIMARY ALONE に切り替えます。

   ```
   ADMIN COMMAND 'hsb set primary alone';
   ```

   ヒント: HotStandby ADMIN COMMAND では、「hotstandby」の代わりに、省略形の「hsb」を使用してもかまいません。

4. 1 次サーバーを 2 次サーバーに接続するために、どちらかのサーバーで以下のコマンドを発行します。

   ```
   ADMIN COMMAND 'hsb connect'
   ```

   接続が成功したかどうかを検証するために、以下のコマンドを発行します。

   ```
   ADMIN COMMAND 'hsb state'
   ```

   1 次サーバーは、状態が「PRIMARY ACTIVE」であることを応答する必要があります。ただし、1 次サーバーの状態が予期される状態以外 (例えば、PRIMARY ALONE) の場合は、以下のコマンドを実行することによって **hsb connect** の状態を確認できます。

   ```
   ADMIN COMMAND 'hsb status connect'
   ```

   結果が ACTIVE である場合、接続プロセスは、まだアクティブです。結果が BROKEN の場合には、接続する前に、両方のサーバーのデータベースを同期させる必要があります。

5. データベースの同期をとるには、1 次サーバーで以下のコマンドを発行します。

   ```
   ADMIN COMMAND 'hsb netcopy'
   ```

   データベース・コピー・プロセスの状況は、次のコマンドを発行して検査します。

   ```
   ADMIN COMMAND 'hsb status copy'
   ```

6. **hsb status copy** の結果が SUCCESS になると同時に、両方のデータベースが同期し、サーバー同士を接続できます。どちらかのサーバーで、以下のコマンドを再発行してください。

   ```
   ADMIN COMMAND 'hsb connect'
   ```

   ステップ 4 で説明したように、成功したかどうかを検証します。

7. アプリケーションの使用を開始します。

関連概念

42 ページの『3.2.1, 1 次および 2 次ノード HotStandby 構成の定義』
少なくとも、HotStandby を構成するには、両方のノードで **HotStandby.HSBEnabled** パラメーターを yes に設定し、2 つのノード間の接続設定を定義する必要があります。

41 ページの『3.2, HotStandby の構成』
HotStandby は、1 次ノードと 2 次ノードの両方で solid.ini ファイルを使用して構成されます。 [HotStandby] セクションには、HotStandby 固有の構成パラメーターが含まれています。**Com.Listen** パラメーターなど、その他のセクションとパラメーターも設定する必要があります。

2.2 HAC を伴う HotStandby のクイック・スタート手順

このセクションでは、solidDB 高可用性コントローラー (HAC) を伴う HotStandby のクイック・スタート手順について説明します。

このタスクについて

この手順は、 31 ページの『2.1, HotStandby クイック・スタート手順』に似ています。しかし、単に 2 つの HSB サーバーをセットアップする代わりに、この手順では、高可用性コントローラー (HAC) によって可用性が保証された高可用性 HSB システムをセットアップする方法を説明します。 以下の手順を実行すると、障害に対する耐久力のある HSB システムでアプリケーションにサービスを提供できます。

高可用性 HSB システムをセットアップして実行するには、ネットワークで接続した 2 台のコンピューターが必要です。各 HSB サーバーに HAC の 1 つのインスタンスをセットアップします。

手順

1. 1 次サーバーと 2 次サーバーを構成するには、 31 ページの『2.1, HotStandby クイック・スタート手順』に従います。

   その結果、一方のサーバーは PRIMARY ACTIVE 状態で、もう一方のサーバーは SECONDARY ACTIVE 状態になります。
2. 1 次サーバーと 2 次サーバーで HAC を構成します。HAC は、作業ディレクトリーに置かれた solidhac.ini ファイルから、それ自体の構成を読み取ります。以下のリストには、必須構成パラメーターが含まれています。

   [HAController] セクション:

   - **Listen**=<*listen address, 'tcp' and chosen port #*>
   - **Username**
   - **Password**
   - **DBUsername**
   - **DBPassword**

   [LocalDB] セクション:

   - **Connect**=<*connect address, protocol, ip/hostname, port #*>

- **StartScript** (デフォルト値である **EnableDBProcessControl**=Yes の場合は必須)

[RemoteDB] セクション:

- **Connect**=<*connect address, protocol, ip/hostname, port #*>

3. 『2.2.1, HA コントローラーの開始と停止』で説明したように、両方のノードで HAC を開始します。

   HAC は自動的に、どちらのサーバーが新規 1 次サーバーになるべきかを各サーバーの前のロールに応じて判別し、ログの位置を検出します。HAC は、71 ページの『3.4.9, どちらのサーバーを 1 次サーバーにするかの選択』で説明されているメカニズムを使用します。一部の特殊な状況では、例えば、両方のサーバーが空のデータベースを使用して始動した場合などでは、いずれのサーバーも新規 1 次サーバーの候補として同等な適性を備えています。そのような状況では、HAC は solidhac.ini 内の [LocalDB] セクションで **PreferredPrimary** パラメーターが Yes に設定されている場合、ローカル・サーバーを選択します。

4. 2 番目のステップの後、一方のサーバーが PRIMARY ACTIVE 状態で、もう一方が SECONDARY ACTIVE 状態になるはずです。ロールを切り替えるために、2 次サーバーで以下のコマンドを発行します。

   ```
   ADMIN COMMAND 'hsb switch primary'
   ```

   あるいは、1 次サーバーで以下のコマンドを実行します。

   ```
   ADMIN COMMAND 'hsb switch secondary'
   ```

   この時点以降、HAC は必要な場合にサーバーのロールを切り替えます。つまり、常に少なくとも 1 つのサーバーが、読み取りと書き込みのトランザクションを実行します。

5. アプリケーションの使用を開始します。

関連概念

47 ページの『3.3, HA コントローラーおよび HA マネージャーの構成』
高可用性コントローラー (HAC) は、それぞれの HotStandby サーバー・ノードにデプロイされます。これは、solidhac.ini 構成ファイルを使用して構成されます。高可用性マネージャー (HAM) は、HAManager.ini 構成ファイルを使用して構成されます。

関連資料

134 ページの『A.3, 高可用性コントローラー (HAC) パラメーター』
このセクションでは、solidhac.ini 構成ファイルにある、高可用性コントローラー (HAC) の構成パラメーターを説明します。

142 ページの『A.5.2, solidhac.ini 構成ファイル』
以下は高可用性コントローラー (HAC) 構成ファイル (solidhac.ini) の例 (抜粋) です。

2.2.1 HA コントローラーの開始と停止

HAC を開始するには、solidhac.ini 構成ファイルを事前に HAC 作業ディレクトリーに入れておく必要があります。HAC の構成の簡単な説明については、33 ページの『2.2, HAC を伴う HotStandby のクイック・スタート手順』を参照してください。HAC の構成に関する詳しい説明は、47 ページの『3.3, HA コントローラーお

よび HA マネージャーの構成』にあります。パラメーターおよびパラメーターの詳しい説明の完全なリストについては、 134 ページの『A.3, 高可用性コントローラー (HAC) パラメーター』、および 142 ページの『A.5.2, solidhac.ini 構成ファイル』を参照してください。

注: 使用しているプラットフォームに応じて、HAC バイナリーには、solidhac または solidhac.exe という名前が付いています。例の中では、分かりやすいよう、solidhac.exe を使用します。同様に、サンプルの solidDB 開始スクリプトの名前は、start_solid.sh または start_solid.bat です。例の中では、start_solid.bat を使用します。

HAC の開始

HAC の開始の構文は以下のとおりです。

```
solidhac.exe [-c working_directory | -?]
```

? 引数、または **-c** を除く他のすべての引数は、使用法のメッセージを出力します。

HAC は、開始すると、solidhac.ini 構成ファイルの **HAController.Listen** パラメーターで指定されたポートの listen を開始します。このポートは、例えば、HAC と HAManager 間のコマンドの転送や、HAC 固有の ADMIN COMMAND の実行に使用されます。

HAC の停止

コマンドを実行するには、まず、solidhac.ini 構成ファイルの **HAController.Listen** パラメーターで定義されたポートを使用して HAC に接続する必要があります。例えば、**solsql** または ODBC インターフェースを使用して HAC に接続することができます。

次のコマンドを実行して、HAC を停止 (強制終了) することができます。

```
ADMIN COMMAND 'hacontroller shutdown'
```

例: HAC の開始

solidhac.exe が現行ディレクトリーでもある c:¥solid¥hac に置かれていて、c:¥solid¥run¥server1 を HAC の作業ディレクトリーとして使用する場合は、HA コントローラーを次のコマンドで開始します。

```
solidhac.exe -c c:¥solid¥run¥server1
```

または

```
solidhac.exe -c ..¥run¥server1
```

関連資料

174 ページの『C.2, 高可用性コントローラーのコマンド (ADMIN COMMAND)』

2.3 開始シーケンスの要約

以下の図と表は、HAC を伴うインストールと HAC を伴わないインストールのシーケンスを横並びに表したものです。図では、HAC のインストール・シーケンスは左側に、HAC を伴わないシーケンスは右側に示されています。図の後の表では、両方のインストール・タイプについて、図中で番号が付いている各ステップを説明しています。

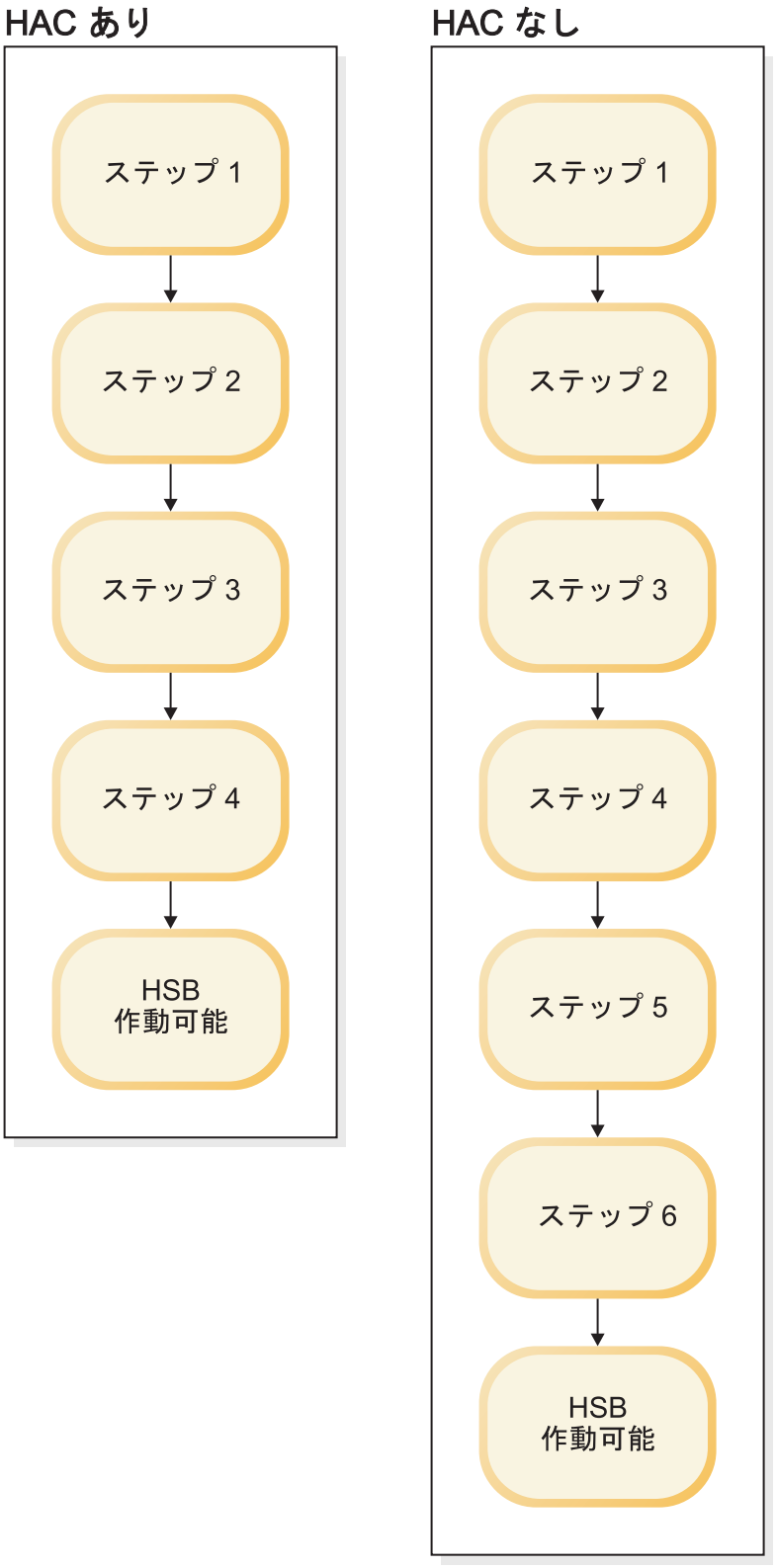


図 11. 開始シーケンスの要約

表 5. インストール・シーケンスのステップ

HAC を伴うインストール	HAC を伴わないインストール
ステップ 1. HSB サーバーを構成します。	
ステップ 2. HA コントローラーを構成します。 ステップ 3. 両方のノードで HA コントローラーを開始します。 ステップ 4. 必要であれば、HSB のロールを切り替えます。	ステップ 2. 両方のノードで HSB サーバーを始動します。 ステップ 3. データベース・サーバーの状態を PRIMARY ALONE に切り替えます。 ステップ 4. 1 次サーバーを 2 次サーバーに接続します。 ステップ 5. 接続に失敗した場合、1 次サーバーから 2 次サーバーへのデータベース・コピーを開始します。 ステップ 6. **netcopy** の後、再接続します。
HSB が作動可能になります。アプリケーションの使用を開始できます。	

2.4 HotStandby のサンプル

solidDB サーバー・パッケージには、HAC またはサンプルの Watchdog のインプリメンテーションを使用して HotStandby コンポーネントのデモンストレーションを実行するためのサンプルが含まれています。

HAC を伴う HotStandby

高可用性コントローラー (HAC) を伴う HotStandby のサンプルは、samples¥hac ディレクトリーにあります。サンプルの詳しい説明は、ファイル samples¥hac¥readme.txt に記載されています。

Watchdog サンプルを伴う HotStandby

Watchdog サンプルを伴う HotStandby は、samples¥hsb ディレクトリーにあります。サンプルの詳しい説明は、ファイル samples¥hsb¥readme.txt に記載されています。

3 HotStandby の管理および構成

このセクションでは、HotStandby 関連のパラメーターと ADMIN COMMAND を使用して、HSB サーバーと HAC インスタンスを含め、HotStandby インストール済み環境を保守する方法について説明します。

この説明は、セクションに記載されている情報の補足です。

3.1 HotStandby 管理の基本

HotStandby 関連のコンポーネントは、構成パラメーターおよび ADMIN COMMAND を使用して管理します。

構成パラメーター

パラメーターは、solid.ini 構成ファイル内のセクション・カテゴリーに従ってグループ化されます。HotStandby を使用する場合は、solid.ini 構成ファイルの [HotStandby] セクションを主に使用します。その他のセクションは、solidDB サーバーの全体的な動作の構成に使用できます。

高可用性コントローラー (HAC) は、solidhac.ini 構成ファイルを使用して構成します。solidhac.ini 構成ファイル内のパラメーターも、さまざまなセクション・カテゴリーに従ってグループ化されます。

高可用性マネージャー (HAM) は、HAManager.ini 構成ファイルを使用して構成されます。 HAManager.ini 内のパラメーターは、主に HA コントローラー・インスタンスの識別に使用されます。これにより、高可用性マネージャーはそれらのインスタンスにアクセスできます。

構成パラメーターは、以下の方法で変更できます。

- 手動で solid.ini、solidhac.ini、HAManager.ini の各構成ファイルを編集する。

  注: サーバーは始動のときにのみ構成ファイルを読み取ります。このため、構成ファイルに加えた変更は、対応するプログラムを次回に開始するときまで効果がありません。
- 動的に以下の ADMIN COMMAND 構文を使用して、稼働中の solidDB サーバーの設定を変更する。

  ```
  ADMIN COMMAND 'parameter section_name.param_name=value';
  ```

  以下に例を示します。

  ```
  ADMIN COMMAND 'parameter hotstandby.2SafeAckPolicy=2';
  ADMIN COMMAND 'parameter com.listen="tcp sf_server 1315"';
  ```

注: 構成パラメーターはすべて動的に変更できません。

HotStandby ADMIN COMMAND (HotStandby API)

HSB API は、solidDB SQL の構文拡張として提供され、次のような ADMIN COMMAND の形式を取ります。

```
ADMIN COMMAND hotstandby hsb-command options
```

または

```
ADMIN COMMAND hsb hsb-command options
```

HSB コマンドは、SQL に対応した任意の対話式ツール (**solsql** など) を介して発行するか、または ODBC や JDBC を通じてプログラマチックに発行することができます。

HotStandby とアクセス権限

HotStandby には固有のアクセス権限がありません。通常の管理者アクセス権限で十分です。HotStandby 管理コマンドを実行するには、SYS_ADMIN_ROLE アクセス権限または SYS_CONSOLE_ROLE アクセス権限が必要です。

HotStandby と solidDB ツール

HotStandby サーバーでは、solidDB データ管理ツールを使用できます。

ADMIN COMMAND を実行するためのコンソール・ツール

HotStandby 固有の管理コマンド (**ADMIN COMMAND 'hotstandby <option>'**) は、solidDB SQL エディター (**solsql**)、および solidDB リモート制御 (**solcon**) で実行できます。

solsql を使用している場合、コマンド名は引用符とともに指定する必要があります。以下に例を示します。

```
ADMIN COMMAND 'hotstandby status connect';
```

solcon で使用する場合、コマンド名は ADMIN COMMAND の接頭部と引用符を使用せずに指定する必要があります。以下に例を示します。

```
 hotstandby status connect
```

インポートおよびエクスポート・ツール

インポートおよびエクスポート・ツールでは、solidDB Speed Loader (**solloado** または **solload**)、solidDB エクスポート (**solexp**)、および solidDB データ・ディクショナリー (**soldd**) も、HotStandby とともに使用することができます。

高可用性マネージャー (HAM) のサンプル

solidDB パッケージには、高可用性マネージャー (HAM) というサンプル・ツールも含まれています。このツールでは、solidDB HotStandby の状態をモニターし、HSB サーバーおよび高可用性コントローラー (HAC) を制御することができます。HAM は、HA コントローラーとのみ連動させることができます。

3.1.1 HotStandby 構成パラメーターの照会

標準パラメーター操作コマンドを使用して、HotStandby パラメーターの値およびプロパティーを照会できます。

コマンドは以下のとおりです。

```
ADMIN COMMAND '[describe] parameter[section_name[.parameter_name]]';
```

以下に例を示します。

```
ADMIN COMMAND 'parameter logging.durabilitylevel';
RC TEXT
-- ----
0 Logging DurabilityLevel 3 3 2

ADMIN COMMAND 'parameter hotstandby.MaxLogSize';
RC TEXT
-- ----
0 HotStandby MaxLogSize 10000000 0 0
```

結果行に示されている 3 つの値は、左から以下のとおりです。

- *現行値* - 動的に設定されるか、デフォルト値またはファクトリー値から継承されます。
- *デフォルト値* - solid.ini ファイルから最初に読み取られるか、ファクトリー値から継承されます。
- *ファクトリー値* - 製品に事前設定されています。

3.1.2 HotStandby 構成パラメーターの変更

通常、パラメーターの値は、solid.ini 構成ファイルの値を変更し、サーバーを再始動することによって変更します。ただし、ほとんどの HotStandby パラメーターが ADMIN COMMAND でも変更可能です。

このタスクについて

パラメーター変更の構文は以下のとおりです。

```
ADMIN COMMAND 'parameter section_name.parameter_name=value [temporary]';
```

temporary オプションが使用されていなければ、パラメーターに対するすべての変更内容が次のチェックポイントで solid.ini ファイルに保存されます。以下のコマンドを使用すると、保存処理を効率よく行うことができます。

```
ADMIN COMMAND 'save parameters [file_name]';
```

デフォルトでは、このコマンドはデフォルトの solid.ini ファイルを書き換えます。 *file_name* オプションを使用することにより、異なる場所に出力を送信できます。

3.2 HotStandby の構成

HotStandby は、1 次ノードと 2 次ノードの両方で solid.ini ファイルを使用して構成されます。 [HotStandby] セクションには、HotStandby 固有の構成パラメーターが含まれています。**Com.Listen** パラメーターなど、その他のセクションとパラメーターも設定する必要があります。

3.2.1 1 次および 2 次ノード HotStandby 構成の定義

少なくとも、HotStandby を構成するには、両方のノードで **HotStandby.HSBEnabled** パラメーターを yes に設定し、2 つのノード間の接続設定を定義する必要があります。

[HotStandby] セクションに必要な最小の構成情報は、以下のとおりです。

- **HotStandby.HSBEnabled** パラメーターを yes に設定する必要があります。

  HotStandby.HSBEnabled を yes に設定しないと、サーバーは非 HotStandby サーバーとして始動し、HotStandby レプリケーションは使用されません。
- **HotStandby.Connect** 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、もう一方のサーバー (1 次サーバーまたは 2 次サーバー) に接続するために使用するネットワーク名を定義します。solid.ini ファイルにこのパラメーターが設定されていない場合、サーバーは接続を必要としない状態 (例えば、PRIMARY ALONE、SECONDARY ALONE、および STANDALONE) でのみ実行できます。サーバーが始動した後は、ADMIN COMMAND 'parameter' コマンドを使用してこのパラメーターを設定または変更できます。
- 各サーバーの **HotStandby.Connect** ストリングは、もう一方のサーバーの **Com.Listen** ストリングに一致する必要があります。

例: ノード 1 の最小限の構成

```
[Com]
Listen=tcp 2315

[HotStandby]
HsbEnabled=yes
;The server on Node 1 connects to the server on Node 2
;using the connect string 'tcp 2325':
Connect=tcp 2325
```

例: ノード 2 の最小限の構成

```
[Com]
Listen=tcp 2325

[HotStandby]
HsbEnabled=yes
;The server on Node 2 connects to the server on Node 1
;using the connect string 'tcp 2315':
Connect=tcp 2315
```

3.2.2 切断された接続または使用不可の接続の検出に役立つ、HotStandby サーバー待ち時間の設定

HotStandby サーバーは、タイムアウト・パラメーターを使用して、既存の接続が切断されたか新規接続が確立されなかったと結論付けるまでの待ち時間を制御します。

このタスクについて

タイムアウト・パラメーターは、以下のとおりです。

- **HotStandby.PingTimeout**
- **HotStandby.PingInterval**

- **HotStandby.ConnectTimeout**

PRIMARY ACTIVE 状態または SECONDARY ACTIVE 状態の HotStandby サーバーは、もう一方のサーバーに接続しようとして、指定された時間内に応答を受信しなかった場合、PRIMARY UNCERTAIN、PRIMARY ALONE、または SECONDARY ALONE に変わります。

手順

サーバーの待ち時間を制御するために、以下のことができます。

- **PingTimeout** パラメーターを設定して、サーバーが PRIMARY UNCERTAIN 状態に変わるまでの待ち時間を指定できます。
- **PingInterval** パラメーターを設定して、サーバーが正常であることを示す「ping」メッセージの送信間隔を指定できます。
- **ConnectTimeout** パラメーターを設定して、サーバーが (例えば、**ADMIN COMMAND 'hotstandby connect'** 操作などで) もう一方のサーバーに対して新規接続を確立しようと試みるときの、待ち時間を指定できます。します。

PingTimeout および PingInterval パラメーター [HotStandby]

[HotStandby] セクション内のオプションの **PingTimeout** パラメーターおよび **PingInterval** パラメーターは、ping 操作を制御します。

「ping」操作は、基本的に 1 つのデータベース・サーバーから別のデータベース・サーバーへ送信される「私は生きています」メッセージです。(一部のネットワーキング・ソフトウェアにも「ping」操作がありますが、[HotStandby] セクション内の solidDB**PingTimeout** 構成パラメーターは、solidDB サーバーの ping にのみ適用され、一般的なネットワーク ping には適用されません。)言い換えると、これは受動的なハートビート・システムを指しています。このパラメーターを設定した場合、1 次と 2 次の両方の HotStandby サーバーは定期的な間隔で、互いに「ping」メッセージを送信します。 4 ページの『ハートビート』も参照してください。

- **PingTimeout** は、もう一方のサーバーがダウンしているかアクセス不能であると結論付けるまでのサーバーの待ち時間を指定します。デフォルトは 4000 (4 秒) です。
- **PingInterval** は、2 回の ping の間の間隔をミリ秒単位で指定します。デフォルトは 1000 (1 秒) です。

例えば、**PingInterval** が 10 秒の場合、各サーバーは互いに 10 秒ごとに ping メッセージを送信します。**PingTimeout** が 20 秒で、一方のサーバー (S1) が 20 秒以内に、もう一方 (S2) からの応答を聴かなかった場合、S1 は S2 がダウンしているかアクセス不能であると結論付けます。その後、サーバー S1 は別の状態に (例えば「PRIMARY ACTIVE」から「PRIMARY UNCERTAIN」に) 切り替わります。

上記のパラメーターの値が異なる場合は、1 次サーバー内で「hsb connect」コマンドの実行中に設定された値が優先されます。値は、切り替え時に変更されません。しかし、ADMIN COMMAND「パラメーター」コマンドで動的に変更できます。

PingTimeout をゼロに設定した場合、ping は使用不可になります。

ping は、ほとんどオーバーヘッドを必要とせず、solidDB サーバーは、欠落した ping メッセージに迅速に対応するようセットアップされています。**PingInterval** 値は、1 秒またはそれ以下のかなり短い間隔に設定できます。

サーバーに障害が起きたときに、フェイルオーバーを素早く検出することが重要な場合は、**PingTimeout** 値を比較的に短い時間に設定します。ただし、値が小さいほど、「誤報」が発生する可能性も高くなります。ネットワークに多量のトラフィックがあり、そのために ping メッセージを受信するまでに遅延が生じる場合は、誤報を避けるために、**PingTimeout** に大きな値を設定することが必要になる場合もあります。

ConnectTimeout パラメーター [HotStandby]

一部のネットワーク実装環境では、接続操作が不定の期間、応答しないことがあります。考えられる理由の 1 つは、リモート・マシンが既知のノードであっても、接続の試行中に使用不可であることです。接続タイムアウト値を指定することにより、HotStandby 接続操作でリモート・マシンに接続するまでの最大待ち時間を、秒単位で設定できます。

ConnectTimeout パラメーター (一部のプラットフォームでのみ有効) は、特定の管理コマンドとの組み合わせでのみ使用します。これに該当するものは以下のとおりです。

- **hotstandby connect**
- **hotstandby switch primary**
- **hotstandby switch secondary**

接続タイムアウト値は、solid.ini ファイルの [HotStandby] セクションで、**ConnectTimeout** パラメーターを使用して、ミリ秒単位で設定します。単位はミリ秒です。デフォルトは 0 で、これはタイムアウトなしを意味します。これは、別の値に設定できます。以下に例を示します。

```
[HotStandby]
 ; Set ConnectTimeout to 20 seconds (20000 milliseconds).
ConnectTimeout=20000
```

3.2.3 トランザクション持続性レベルの定義

トランザクション持続性レベルは、**Logging.DurabilityLevel** パラメーターで設定します。このパラメーターには 3 つの値があり、リラックス (1) 持続性、アダプティブ (2) 持続性、ストリクト (3) 持続性にそれぞれ対応します。

アダプティブ持続性は、HotStandby でのみ使用されます。アダプティブ持続性の意味は、以下のとおりです。

- 1 次サーバーと 2 次サーバーが接続されていて、正常に機能している場合 (それぞれ、PRIMARY ACTIVE 状態と SECONDARY ACTIVE 状態にある場合)、サーバーはリラックス持続性を使用します。
- それ以外のすべての状態 (例えば、PRIMARY ALONE、STANDALONE など) では、サーバーはストリクト持続性を使用します。

ストリクトとリラックスの持続性の相違に関する説明、または **Logging.DurabilityLevel** パラメーターの詳細については、「*IBM solidDB 管理者ガイド*」を参照してください。

3.2.4 HotStandby データベース・コピー操作の名前と場所の定義

オプションの **HotStandby.CopyDirectory** パラメーターは、HotStandby **copy** 操作のコピー先となるディレクトリーの名前と場所を定義します。

HotStandby コピー操作は、以下のコマンドで指定します。

```
ADMIN COMMAND 'hotstandby copy [directory_name]';
```

HotStandby.CopyDirectory パラメーターにはデフォルト値がないので、solid.ini ファイル内でディレクトリーを指定しなかった場合は、コピー・コマンドの中でそれを指定する必要があります。**HotStandby.CopyDirectory** パラメーターに相対パスを指定する場合は、1 次サーバーの solid.ini ファイルを保持するディレクトリーからの相対パスにします。

HotStandby.CopyDirectory パラメーターは、**ADMIN COMMAND 'hotstandby netcopy'** コマンドを使用して HotStandby データベース・コピー操作を実行する場合は、必要ありません。これら 2 つの選択肢のうち、推奨されるのは、機能がより柔軟な **netcopy** コマンドです。

3.2.5 2 次サーバー障害時の 1 次サーバーの動作の定義

[HotStandby] セクションの **AutoPrimaryAlone** パラメーターを使用して、2 次サーバーとの接続が失われた後の 1 次サーバーを PRIMARY ALONE 状態に自動的に切り替えるか、それとも PRIMARY UNCERTAIN 状態に留めるかを制御できます。

AutoPrimaryAlone が Yes に設定されている場合、1 次サーバーは 2 次サーバーとの接続が失われたとき、PRIMARY ALONE 状態に自動的に切り替わります。これにより、1 次サーバーはトランザクションの受け入れを続行できます。
AutoPrimaryAlone が No に設定されている場合、2 次サーバーとの接続が失われた 1 次サーバーは、PRIMARY UNCERTAIN 状態に自動的に切り替わります。

デフォルトでは、**AutoPrimaryAlone** は No に設定されます。

```
[HotStandby]
AutoPrimaryAlone =  No
```

PRIMARY UNCERTAIN 状態では、1 次サーバーは新規トランザクションの受け入れや現在アクティブなトランザクションのコミットを行うことができません。1 次サーバーは、HAC、Watchdog、またはシステム管理者が指示するまで、PRIMARY ALONE 状態に切り替わりません。

AutoPrimaryAlone が No に設定されている場合は、**ADMIN COMMAND 'hotstandby set primary alone'** コマンドを実行することにより、サーバーを PRIMARY ALONE 状態に設定できます。このコマンドは、構成ファイル内の **AutoPrimaryAlone** の値を変更しないことに注意してください。

デフォルトを Yes に変更した場合、1 次サーバーの状態は PRIMARY ACTIVE から PRIMARY ALONE に変更され、PRIMARY UNCERTAIN には変更されません。

3.2.6 1 次および 2 次パラメーター値の調整の確認

このセクションでは、1 次および 2 次サーバーでどのパラメーターを同一にする必要があるか、およびどのパラメーターを異ならせる必要があるかについて説明します。

1 次側および 2 次側の両方で特定のパラメーターを同一にする必要があります。この理由は、フェイルオーバーの後で、元の 2 次側が新規の 1 次側になり、それが古い 1 次側と同じ動作をしなければならないからです。注意すべき点は、同じ値を使用することが絶対要件ではないことです。異なる値が使用されたときに、サーバーで障害が起きるのではなく、クライアントで認識する動作が異なることがあります。

[HotStandby] セクションにないが、間接的に関係している一部のパラメーターも、1 次サーバーおよび 2 次サーバーの両方で同一にする必要があります。例えば、**DurabilityLevel** パラメーターは一般に、1 次側および 2 次側で同じでなければなりません。

1 次サーバーおよび 2 次サーバーで特定のパラメーターを異ならせる必要があります。この理由は、両サーバーを一意的に識別できるように、かつ相互に対話できるようにするためです。

以下の HotStandby パラメーターが 1 次側および 2 次側の両方で同じでなければなりません。

- [HotStandby]
  - **2SafeAckPolicy**
  - **AutoPrimaryAlone**
  - **ConnectTimeout**
  - **HSBEnabled**
  - **PrimaryAlone** (推奨されないが、使用した場合は同じにする必要がある)
- [IndexFile]
  - **FileSpec** は「互換性」が必要です。つまり、**FileSpec** パラメーターの数を同じにする必要と、対応する **FileSpec** パラメーターのサイズを一致させる必要があります。
  - **BlockSize**
- [Logging]
  - **BlockSize**

以下のパラメーターが異なっていなければなりません。

- [HotStandby]
  - **Connect**

以下のパラメーターは、コンピューター上のディスク・ドライブ構成などの環境に基づいて、同じであるか異なる可能性があります。

- [General]
  - **BackupDirectory**
- [HotStandby]

– **CopyDirectory**

HSB パフォーマンスに影響する、「非 HSB」パラメーターの設定もいくつかあります。例えば、solid.ini ファイルの [Logging] セクションにある **DurabilityLevel** パラメーターには、HotStandby でパフォーマンスを最適化できる設定があります。 15 ページの『1.1.5, 持続性とロギング』と、「*IBM solidDB 管理者ガイド*」の **DurabilityLevel** の説明を参照してください。

3.2.7 1 次側の設定が 2 次側の設定より優先されるかどうかの決定

パラメーターによっては、1 次サーバーおよび 2 次サーバーの両方で同じである必要があります。同じ値を設定しないと、それぞれのサーバーが、そのサーバーの solid.ini ファイルに定義されている値を使用するようになることが予想されます。ただし、これは必ずしも当てはまりません。

2SafeAckPolicy のような、2 次側の動作を制御する一部のパラメーターの場合であっても、1 次側の値によって、その動作が決まります。原則的に、安全性および持続性のためのすべてのパラメーターが 1 次側で制御されます。例えば、1 次側は **2SafeAckPolicy** の値を読み取って、その値を 2 次側に使用するように送信します。2 次側の solid.ini ファイルに格納されている値は、2 次側が 1 次側になった場合にのみ使用されます。

1 次側の値が優先されるパラメーターには以下のパラメーターが含まれます。

- **HotStandby.SafenessLevel**
- **HotStandby.2SafeAckPolicy**
- **Logging.DurabilityLevel**
- **HotStandby.NetcopyRpcTimeout**

コマンド '**hsb connect**' の実行時点で、1 次側にある以下のパラメーターが優先されます。

- **HotStandby.PingTimeout**
- **HotStandby.PingInterval**

3.3 HA コントローラーおよび HA マネージャーの構成

高可用性コントローラー (HAC) は、それぞれの HotStandby サーバー・ノードにデプロイされます。これは、solidhac.ini 構成ファイルを使用して構成されます。高可用性マネージャー (HAM) は、HAManager.ini 構成ファイルを使用して構成されます。

HA コントローラー

HAC 構成ファイル solidhac.ini は、HAC 作業ディレクトリーに配置する必要があります。solidhac.ini 構成ファイル内のパラメーターは、以下のセクションの下にグループ化されています。

- HAController
- LocalDB

- RemoteDB
- ERE

パラメーターについて詳しくは、 134 ページの『A.3, 高可用性コントローラー (HAC) パラメーター』を参照してください。

構成パラメーターはすべて、 142 ページの『A.5.2, solidhac.ini 構成ファイル』の solidhac.ini サンプル・ファイルの中にも記載されています。

HA マネージャー

HAM 構成ファイル HAManager.ini は、HAM 作業ディレクトリーに配置する必要があります。

パラメーターについて詳しくは、 140 ページの『A.4, 高可用性マネージャー (HAM) の構成パラメーター』を参照してください。

構成パラメーターはすべて、 145 ページの『A.5.3, HAManager.ini 構成ファイル』の HAManager.ini サンプル・ファイルの中にも記載されています。

3.4 ADMIN COMMAND (HotStandby API) による HotStandby の管理

solidDB HotStandby の高可用性を管理するすべてのソフトウェアで、サーバー・プロセスのモニターと制御を行うために、HotStandby API (HSB API) が使用されます。製品に組み込まれている HA コントローラーは、そのようなプログラムの一例です。もう 1 つの例は、Watchdog サンプル・プログラムです。

HSB API は、solidDB SQL の構文拡張として提供され、次のような ADMIN COMMAND の形式を取ります。

```
ADMIN COMMAND hotstandby hsb-command options
```

または

```
ADMIN COMMAND hsb hsb-command options
```

HSB コマンドは、SQL に対応した任意の対話式ツール (**solsql** など) を介して発行するか、または ODBC や JDBC を通じてプログラマチックに発行することができます。

このセクションを使用して独自のアプリケーションをプログラムし、solidDB の高可用性を管理できます。これは、例えば、外部クラスター管理ソフトウェアへの統合を実装する場合などに必要になります。

3.4.1 管理タスクの概要

このセクションでは、HotStandby の使用時に実行しなければならない場合がある管理タスクについて説明します。

このセクションのトピックは、以下のとおりです。

表 6. 管理タスク

トピック	説明	ページ
HotStandby のリカバリー・タスクと保守タスクの実行	システム障害 (通信リンクの切断または動作不能のホット・スタンバイ・サーバーが原因で起きたもの) の場合の HotStandby タスクについて説明します。これらのタスクには以下が含まれます。 • サーバー状態の切り替え • HotStandby 操作のシャットオフ • 1 次サーバーと 2 次サーバーの同期 • HotStandby サーバーの接続	『3.4.2, HotStandby のリカバリーと保守の実行』
ネットワークを介した 1 次側データベースの新規 2 次側へのコピー	リモート・サーバーが HotStandby 構成に新規に追加されたものである (つまり、新規 2 次サーバーである) 場合、またはリモート・サーバーのデータが破損し、置き換える必要がある場合の、データベースのリモート (ネットワーク) コピーの作成方法を説明します。	60 ページの『ネットワークを介した 1 次データベースの 2 次側へのコピー』
HotStandby 状況の検査	1 次サーバーおよび 2 次サーバーの HotStandby 状況情報を検査する方法を説明します。	66 ページの『3.4.7, HotStandby 状況の検査』
HotStandby サーバー状態の検証	HotStandby サーバーの状態 (1 次、2 次、スタンドアロン) を検査する方法を説明します。	69 ページの『3.4.8, HotStandby サーバー状態の検証』
HotStandby サーバーから非 HotStandby サーバーへの変更	HotStandby 用に構成したサーバーを、通常の非 HotStandby サーバーに設定する方法を説明します。	72 ページの『3.4.10, HotStandby サーバーから非 HotStandby サーバーへの変更』

3.4.2 HotStandby のリカバリーと保守の実行

システム障害 (通信リンクの切断または動作不能のホット・スタンバイ・サーバーが原因で起きたもの) またはサーバー保守の場合、HotStandby タスクを実行しなければならないことがあります。これらのタスクには、サーバー状態の切り替え、HotStandby 操作のシャットオフ、1 次サーバーと 2 次サーバーの同期、および HotStandby サーバー同士の接続が含まれます。

手順

1. 以下の操作の一部または全部を実行します。
   a. サーバー状態を切り替えます。

      これには、1 次サーバーの PRIMARY ALONE 状態 (後で 2 次サーバーへ送信できるよう、トランザクション・ログへのトランザクションの蓄積を続行します) への切り替え、または HotStandby のシャットダウンが含まれます。
   b. 1 次側と 2 次側のデータベースが同一になるよう、サーバー同士の同期をとります。
   c. 何らかの理由で通信リンクが切断されている場合は、1 次サーバーを 2 次サーバーに接続します。
2. 以下のように、HAC と HA マネージャーで同じステップを実行できます。
   a. HA マネージャーで「**Switch**」ボタンを押します。サーバーをシャットダウンする必要がある場合は、HA マネージャーで「**Shutdown**」ボタンを押します。
   b. HAC インスタンスを ADMINISTRATIVE モードに設定するため、HA マネージャーで「**Administrative**」ボタンを押します。
   c. HAC インスタンスを AUTOMATIC モードに設定するため、HA マネージャーで「**Automatic**」ボタンを押します。

次のタスク

1 次側または 2 次側データベースへのアプリケーションの再接続の詳細については 103 ページの『4.3.1, アプリケーションから 1 次サーバーへの再接続』を参照してください。

重要:

HAC を使用する場合は、HA マネージャーを使用して管理手順を実行するか、HAC インスタンスを ADMINISTRATIVE モードに設定してから、管理作業を開始します。

3.4.3 サーバー状態の切り替え

HotStandby コンポーネントは、必要な場合、自動またはユーザーによる手動でのサーバー状態の切り替えを必要とします。

実動では、サーバー状態は自動的な状態切り替えによって (つまりフェイルオーバーの実行によって) 選択されます。 自動的な状態切り替えは、例えば、solidDB 高可用性コントローラー (HAC) で設定することができます。

切り替え とは、1 次サーバーと 2 次サーバーが、稼働中にロールを逆にすることを意味します。これは、さまざまな保守の目的で必要になる場合があります。

フェイルオーバー とは、1 次サーバーで障害が発生した場合に 1 次サーバーのロールを引き継ぐための 2 次サーバーによるアクションです。

切り替えの実行

2 次サーバーで `ADMIN COMMAND 'hotstandby switch primary';` コマンドを発行するか、1 次サーバーで `ADMIN COMMAND 'hotstandby switch secondary';` コマンドを発行することにより、サーバーのロールを逆にすることができます。

switch コマンドは、2 つのサーバーが接続されている場合でも接続されていない場合でも使用できます。サーバー同士が接続されている場合は、単に状態が逆になり、旧 2 次サーバーが新規 1 次サーバーに、旧 1 次サーバーが新規 2 次サーバーになります。サーバー同士が接続されていない場合は、旧 2 次サーバーが新規 1 次サーバーになり、もう一方のサーバーの状態は変更されません。

下の図は、サーバー同士が接続されているときに、コマンド **hsb switch secondary** または **hsb switch primary** を発行するとどうなるかを示したものです。コマンド **hsb switch primary** は SECONDARY 状態 (例えば、SECONDARY ACTIVE) にあるサーバー上でのみ発行でき、コマンド **hsb switch secondary** は PRIMARY 状態 (例えば、PRIMARY ACTIVE) にあるサーバーでのみ使用できます。

状態切り替え

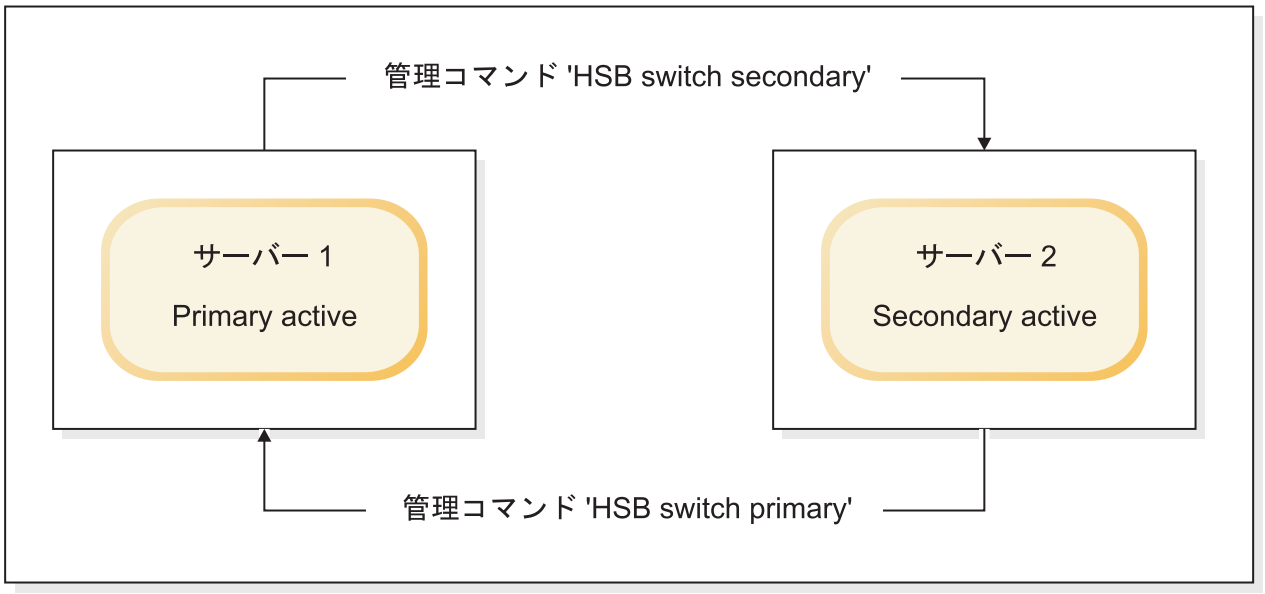


図 12. 状態切り替え

コマンド **hotstandby switch primary** を実行して 2 次サーバー (サーバー 2) を 1 次サーバーに切り替える場合、2 次サーバー (サーバー 2) がもう一方のサーバー (サーバー 1) に接続していないと、エラーが返されます。

2 つのサーバーが接続されている場合は、サーバー同士の状態が切り替わります。つまり、旧 1 次サーバー (サーバー 1) は新規 2 次サーバーになり、旧 2 次サーバー (サーバー 2) は新規 1 次サーバーになります。

旧 2 次サーバー (サーバー 2) がもう一方のサーバー (サーバー 1) に接続できない場合は、両方のサーバーが SECONDARY ALONE に切り替わります。**AutoPrimaryAlone** 構成パラメーターが `yes` に設定されている場合でも、新規 1 次サーバーは PRIMARY ALONE でなく SECONDARY ALONE に切り替わります。

2 次サーバーの 1 次サーバーへの切り替え:

コマンド `ADMIN COMMAND 'hotstandby switch primary';` を発行することにより、2 次サーバーを PRIMARY 状態に切り替えることができます。

`hotstandby switch primary` コマンドを実行すると、状態を切り替えるプロセスが開始されます。切り替えプロセスが正常に開始された場合は、以下のメッセージが表示されます。`Started the process of switching the role to primary.`切り替えの状況は、コマンド **`hotstandby status switch`** を使用してモニターできます。

`hotstandby switch primary` コマンドの後にコミットを発行すると、コミットは `replicated transaction is aborted` というエラーで失敗します。

すべてのトランザクションは、切り替えのときに強制終了されます。ただし、ADMIN COMMAND (管理コマンド) は、トランザクション・コマンドでないので、ロールバックできません。

注: 管理コマンドは、新規トランザクションがまだ開かれていない場合、強制的に新規トランザクションを開始します。オープン・トランザクションが終了しないようにするには、またはトランザクションの開始時刻を予期したものと異なる時刻にするには、管理コマンドの後に COMMIT WORK を実行します。

両方のサーバーが PRIMARY 状態 (例えば、両方が PRIMARY ALONE) になる構成エラーが起きた場合は、コマンド **`hotstandby switch secondary`** を使用して、どちらかのサーバーを SECONDARY 状態に戻すことができます。

- サーバーが同じデータを持っている場合は、両方のサーバー上で通常の操作が再開されます。
- サーバーが同じデータを持っていない場合、1 次サーバーは 2 次サーバーからの接続操作を拒否し、以下のメッセージを発行します。`14525: HotStandby databases are not properly synchronized.`

  HotStandby レプリケーションは開始されません。その場合は、2 次サーバー側に 1 次データベースの完全コピーが必要です。最初に、どちらのデータベースが正しいかを判断する必要があります。

  また、`14525` エラーが発生した場合、データベース状態は変化しません。両方のサーバーはコマンド発行以前と同じ状態のままになります。

関連タスク

67 ページの『切り替え状況情報の表示』
2 つの HotStandby サーバー間で、状態切り替えが発生したかどうかを検証する必要が生じる場合があります。

1 次サーバーの 2 次サーバーへの切り替え:

コマンド `ADMIN COMMAND 'hotstandby switch secondary';` を発行することにより、1 次サーバーを SECONDARY 状態に切り替えることができます。

これは、2 つのサーバーの状態が既に切り替えられており、再び元の状態に切り替えたい場合に特に便利です。例えば、新規 2 次サーバーがサービスに復帰したときに、その状態を切り替えて 1 次サーバーに戻し、新規 1 次サーバーを 2 次サーバーに再び切り替えることができます。

hotstandby switch secondary コマンドを実行すると、状態を切り替えるプロセスが開始されます。切り替えプロセスが正常に開始された場合は、以下のメッセージが表示されます。`Started the process of switching the role to secondary.`切り替えの状況は、コマンド **hotstandby status switch** を使用してモニターできます。

hotstandby switch secondary を実行する場合、サーバー同士がまだ接続されていなければ、旧 1 次サーバーは旧 2 次サーバーに接続しようとします。

2 つのサーバーが接続されている場合は、サーバー同士の状態が切り替わります。つまり、旧 1 次サーバーは新規 2 次サーバーになり、旧 2 次サーバーは新規 1 次サーバーになります。

関連タスク

67 ページの『切り替え状況情報の表示』
2 つの HotStandby サーバー間で、状態切り替えが発生したかどうかを検証する必要が生じる場合があります。

フェイルオーバーの実行

フェイルオーバーは、2 次サーバー側で、コマンド `ADMIN COMMAND 'hotstandby set primary alone';` を実行することによって行われます。

サーバーは、以前に 1 次サーバーから受信した保留中のすべてのトランザクションを処理した後に、新しい状態になります。これにより、失われるトランザクションがないこと、およびデータベース状態に障害直前の 1 次サーバーの状態が反映されることが保証されます。ただし、使用している安全性レベルが 1-safe の場合は、フェイルオーバーの際にいくつかのトランザクションが失われることがあります。

PRIMARY ALONE 状態での新規 1 次サーバーの実行

PRIMARY ALONE 状態では、2 次サーバーへの接続は切断されていますが、1 次サーバーを実行してトランザクション・ログの更新を継続できます。2 次サーバーが復帰した後、PRIMARY ALONE 状態のサーバーは 2 次サーバーへのトランザクションの送信を再開できます。

手順

サーバーを PRIMARY ALONE 状態に設定するには、以下の 3 とおりの方法があります。

- 以下のコマンドを発行します。

  ```
  ADMIN COMMAND 'hotstandby set primary alone';
  ```

- 次のコマンドを使用して、制御された方式での切断を、1 次サーバーと 2 次サーバーのどちらかで実行します。

  ```
  ADMIN COMMAND 'hotstandby disconnect';
  ```

制御されたシャットダウンを行うために、ADMIN COMMAND 'shutdown'; を 2 次サーバー上で実行すると、2 次サーバーはシャットダウンの前に暗黙に切断され、1 次サーバーは安全に PRIMARY ALONE 状態に切り替わります。

- **HotStandby.AutoPrimaryAlone** パラメーターを yes に設定すると、デフォルトで PRIMARY ALONE 状態になります。

  HotStandby.AutoPrimaryAlone=yes の場合、サーバーは 2 次サーバーへの接続が切断されたとき、自動的に PRIMARY ALONE 状態になります。それ以外の場合は、サーバーに障害が起きた後、サーバーの状態は PRIMARY UNCERTAIN のままになります。ただし、HAC、管理者、Watchdog プログラムのいずれかがコマンド ADMIN COMMAND 'hotstandby set primary alone' を発行した場合は除きます。デフォルトでは、solid.ini ファイルの [HotStandby] セクションの **AutoPrimaryAlone** パラメーターは no に設定されます。これは、PRIMARY ACTIVE 状態で作動している 1 次サーバーが、2 次サーバーに障害が起きた場合、自動的に PRIMARY UNCERTAIN に切り替えられることを指定します。

タスクの結果

PRIMARY ALONE 状態は、以下のいずれかが発生するまで持続します。

- 2 次サーバーへの接続が成功した。
- サーバーにトランザクション・ログ用のスペースがなくなった。
- ログ・サイズの限度 (**MaxLogSize**) を超えた。
- 別のコマンドによって、サーバーが STANDALONE など、別の状態に切り替えられた。
- 1 次サーバーがシャットダウンされた。

注意:
2 次サーバーに PRIMARY ALONE 状態を命じるのと同時に 1 次サーバーのシャットダウンを行わないよう注意する必要があります。この 2 つの操作は矛盾しているため、2 次サーバーは SECONDARY ALONE 状態になります。これが現実の操作で偶然起きることは、ほとんどありません。ただし、システムのテスト中に、シャットダウンで 1 次サーバーの障害をシミュレートしたくなる場合があります。しかし、それはしないでください。シャットダウンは障害の代わりにはなりません。それは、1 次と 2 次の両方のサーバーが関与した複雑な分散操作になります。それをしないもう 1 つの理由は、1 次サーバーは、シャットダウンされて新規 2 次サーバーとして始動された後、新規 1 次サーバーにキャッチアップできなくなるからです。1 次サーバーのシャットダウンが本当に必要な場合、正しい手順は以下のとおりです。

1. **切り替えを実行します。**
2. **新規 2 次サーバーをシャットダウンします。**
3. **新規 1 次サーバーが自動的に PRIMARY ALONE 状態に切り替わります。**

2 次サーバーのオンラインへの復帰

手順

2 次サーバーをオンラインに復帰させるには、1 次サーバーを 2 次サーバーに接続します。 詳細については、 65 ページの『3.4.6, HotStandby サーバーの接続』を参照してください。

2 次サーバー・ノードをオンラインに復帰した後、キャッチアップが必要になる場合があります。1 次サーバー内の変更は、長期間にわたって蓄積されています。1 次サーバーが PRIMARY ALONE 状態に設定されていた間、1 次サーバーはトランザクション・ログにトランザクションとデータを書き込んでいました。
2 次サーバーが再び 1 次サーバーに接続すると、同期をとるために、1 次サーバーに保留中の変更がトランザクション・ログから 2 次サーバーへ書き込まれます。変更が 2 次サーバーに書き込まれている間、2 次サーバーは SECONDARY ALONE 状態になり、1 次サーバーは PRIMARY ALONE 状態になります。(コマンド ADMIN COMMAND "hsb status connect' を発行すると、サーバーがキャッチアップを実行中であるかどうかを知らせるメッセージが表示されます。)

注: 1 次サーバーが、コマンド **hotstandby set standalone** を使用して STANDALONE 状態に設定されていた場合、2 次サーバーを SECONDARY ACTIVE 状態にするには、事前に 1 次サーバーから 2 次サーバーにデータベース全体をコピーしておく必要があります。 56 ページの『3.4.5, 1 次サーバーと 2 次サーバーの同期』を参照してください。

タスクの結果

2 次サーバーが、保留中の変更の処理を正常に終了した後、1 次サーバーと 2 次サーバーの状態は、それぞれ PRIMARY ACTIVE と SECONDARY ACTIVE に自動的に変更されます。

3.4.4 HotStandby 操作のシャットオフ

1 次サーバーの HotStandby 操作を一時的にシャットオフする必要が生じる場合があります。例えば、2 次サーバーのサービスをアップグレードのために休止しようとしているとき、1 次サーバーに、2 次サーバーのサービス休止中に蓄積されるトランザクション・ログを保管する十分なディスク・スペースがない場合などです。

このタスクについて

(詳細については、 75 ページの『3.6.3, トランザクション・ログのスペース不足』を参照してください。)

1 次サーバーで HotStandby をシャットオフするには、以下の手順を実行します。

手順

1. サーバーを切断します (現在、接続している場合)。
2. 以下の一連のコマンドを使用して、1 次サーバーを STANDALONE 状態に設定します。

   ```
   ADMIN COMMAND 'hotstandby disconnect'; -- if servers are connected
   ADMIN COMMAND 'hotstandby set standalone';
   ```

   これにより、1 次サーバーは非 HotStandby サーバーであるかのように操作を続行できます。

   注:

   2 次サーバーへ送信するトランザクション・ログの保管を停止すると、もはや単に 1 次サーバーと 2 次サーバーを再接続するだけでは、キャッチアップできな

くなります。代わりに、HotStandby 操作を再開するときに、手動でサーバー同士を同期させることが必要になります。詳細については、以下のセクションを参照してください。

次のタスク

このサーバーを、以後永続的に HotStandby サーバーとして使用しない場合は、 72 ページの『3.4.10, HotStandby サーバーから非 HotStandby サーバーへの変更』を参照してください。

3.4.5 1 次サーバーと 2 次サーバーの同期

各サーバーが HSB レプリケーションを開始できるようにするためには、各サーバーのデータベースが同一でなければなりません。言い換えると、2 次データベースは 1 次データベースの正確なコピーでなければなりません。HotStandby システムの各データベースを同じものにするプロセスを、HotStandby 同期と呼びます。

1 次側と 2 次側の同期をとる必要があるのは、以下の場合です。

- 2 次側が新規で、開始するための 1 次側のデータベースのコピーをまだ持っていない場合。
- 2 次側がしばらくの間実行されておらず、そのデータ・コピーが最新のものでない場合。
- 1 次側と 2 次側の両方が同時に「Primary Alone」状態で実行されていたため、データが矛盾している場合。
- 2 次側のディスク・ドライブが破損したか、ファイルが壊れたため、置き換える必要がある場合。

サーバー上のデータの同期化には、主に 2 つの方法があります。キャッチアップとフル・コピー です。

キャッチアップ

キャッチアップを使用できるのは、サーバー同士の接続が切断されていた間、2 次サーバーに「未着」のすべてのトランザクションのコピーを 1 次サーバーが保管していた場合で、また、その場合に限られます。1 次サーバーがそれらのすべてのトランザクションを保管していた場合、1 次サーバーは 2 次サーバーに再接続したとき、自動的にそれらのトランザクションを 2 次サーバーへ転送し、2 次サーバーが 1 次サーバーに「キャッチアップ」できるようにします。

solidDB サーバーは、PRIMARY ALONE 状態にある間だけトランザクションを (2 次サーバーへ転送するために) 保管し、STANDALONE 状態にある間や非 HotStandby サーバーとして作動している間は、保管しません。したがって、最後に 2 次サーバーと接続していたとき以降に STANDALONE 状態にあったか非 HotStandby サーバーとして作動していたサーバーは、すべてのトランザクションを持っているわけではなく、キャッチアップを行うことができません。代わりに、(後述する) 完全コピーを行う必要があります。

明示的な「キャッチアップ」コマンドは存在しません。以下のコマンドを使用してサーバー同士を接続すると、サーバーは自動的にキャッチアップを試みます。

```
ADMIN COMMAND 'hotstandby connect';
```

1 次サーバーと 2 次サーバーを接続すると、それらのサーバーは自動的に、1 次サーバーがトランザクション・ログ内に 2 次サーバーへ送信するデータを持っているかどうかを検査します。そのデータが存在する場合、サーバーは自動的にキャッチアップを試みます。

キャッチアップ・プロセスの間、1 次サーバーと 2 次サーバーは PRIMARY ALONE 状態と SECONDARY ALONE 状態のままです。クライアントは、照会のサブミットとトランザクションのコミットを続行できます。キャッチアップ・プロセスは、クライアント・アプリケーションに意識されることがなく行われます。

1 次サーバーから 2 次サーバーへトランザクションをコピーした後であっても、1 次サーバーと 2 次サーバーのデータベースが同一でないことをサーバーが認識した場合は、エラー・メッセージが出ます。

キャッチアップが失敗した場合 (または、例えば 1 次サーバーが STANDALONE 状態にあったなどの理由でキャッチアップが機能しないと前もって分かっている場合) は、完全コピーを実行する必要があります。

キャッチアップは、いずれかの時点で 2 次サーバーが既に SECONDARY ACTIVE 状態で稼働している場合にのみ適用されます。2 次サーバーが新品の場合は、たとえ 1 次サーバーがそれまで PRIMARY ALONE 状態で実行されており、その 1 次サーバー自体が始動した時点以降のすべてのトランザクションを保管している場合でも、2 次サーバーにデータベースの初期コピーを与えるために、完全コピーを行う必要があります。

キャッチアップ・プロセスの詳細は、 54 ページの『2 次サーバーのオンラインへの復帰』に説明があります。

完全コピー

完全コピーとは、その名前が示すとおり、すべてのデータを 1 次サーバーから 2 次サーバーにコピーすることです。これは、単数または複数のデータベース・ファイル自体をコピーすることによって行います。

完全コピーは、以下の状況で使用されます。

- 2 次サーバーが新品で、1 次サーバーのデータベースの初期コピーを取得しようとしている場合。
- 1 次サーバーが、PRIMARY ALONE 状態でないときにトランザクションを書き込んだために、キャッチアップが可能でない場合。
- 2 次サーバーのデータベースが壊れたか欠落している場合。
- 2 次サーバーがディスクレスで、障害を起こした場合。2 次ディスクレス・サーバーを障害後に始動したときは、**hotstandby netcopy** コマンドでデータベースの完全なコピーを行う必要があります。ディスク・ベースの 2 次サーバーと異なり、2 次ディスクレス・サーバーは、トランザクション・ログを読み取って、作動不能だった間に発生した変更を適用することができません。
- 1 次サーバーはキャッチアップに必要なすべてのデータを持っているが、キャッチアップを行うと、単に現在のデータ・ファイルをコピーするよりも長時間かかると予想される場合。

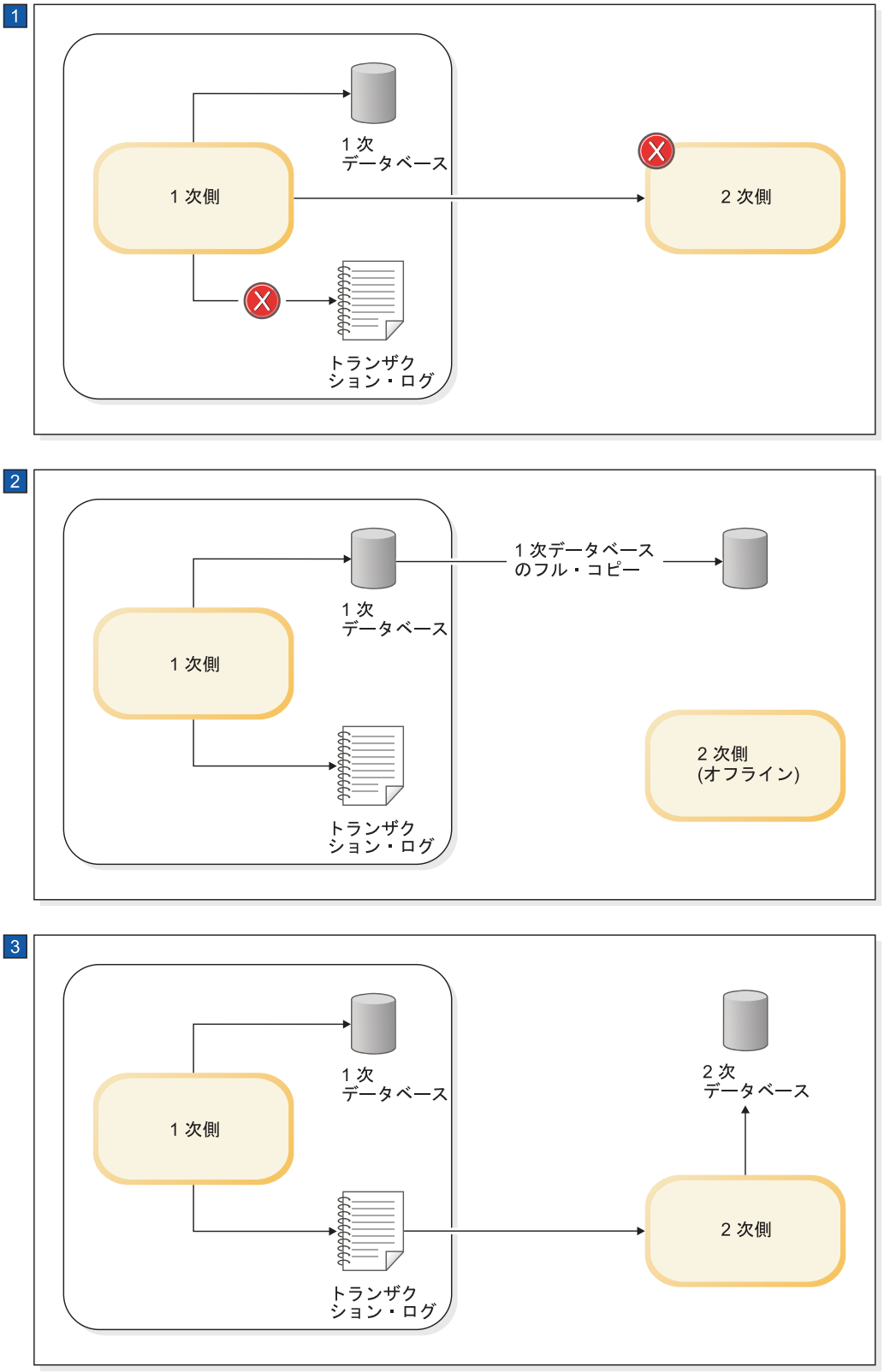
注意:
2 次サーバーに古いデータベース・ファイルがある場合、完全コピーを行うと、それらの古いファイルは上書きされます。何らかの理由で、1 次サーバー内になかったデータが 2 次サーバー上のファイルに含まれている場合 (例えば、両方のサーバーが同時に PRIMARY ALONE 状態で作動していた場合)、そのデータは失われます。

完全コピーを行うことができる (つまり、データベース・ファイルを 1 次サーバーから 2 次サーバーにコピーする) 2 つの HotStandby コマンドがあります。以下のどちらのコマンドを使用してもかまいません。

```
ADMIN COMMAND 'hotstandby netcopy';
ADMIN COMMAND 'hotstandby copy [<directory_name>]';
```

netcopy 操作は、ネットワーク上で稼働してファイルを受信できる 2 次サーバーに、ネットワークを通じてデータベースをコピーします。**copy** 操作は、1 次サーバーに可視の指定されたディスク・ドライブ・ディレクトリーに、データベース・ファイルをコピーします。2 次サーバーは **copy** 操作の間、稼働していてはなりません。通常、**netcopy** コマンドの方が **copy** コマンドより好ましいので、ほとんどの例で **copy** ではなく **netcopy** のみが示されています。

copy コマンドと **netcopy** コマンドについては、64 ページの『1 次サーバーから指定されたディレクトリーへのデータベース・ファイルのコピー』および 60 ページの『ネットワークを介した 1 次データベースの 2 次側へのコピー』に説明があります。



1. 2 次サーバーが長時間ダウンしているために、1 次サーバーが 2 次サーバー用のデータをトランザクション・ログに格納する処理を停止します。このログはまだローカル・リカバリーに使用されます。
2. 1 次データベースが 2 次ノードにコピーされ、1 次サーバーがトランザクション・ログへの書き込みを開始します。
3. データベースがコピーされ、1 次サーバーがトランザクション・ログ・ファイルの変更を 2 次サーバーに送信します。

図 13. 手動での完全コピー手順

注:

上の図では、トランザクション・ログの使用が過度に単純化されています。図の最初の部分で、1 次サーバーと 2 次サーバーが接続されていないとき、1 次サーバーは実際にトランザクション・ログへのデータの書き込みを続行しますが、リカバリーを行うのに十分なデータだけを保持し、2 次サーバーが接続の切断以降のすべての変更をキャッチアップするために十分なデータを保持するわけではありません。

Watchdog を使用したサーバーの同期

手動でサーバーの同期をとることができるコマンドを、Watchdog プログラムで使用して、自動的にサーバーの同期をとることもできます。

キャッチアップが十分であれば、Watchdog で行う必要があることは、2 次サーバーをモニターして 2 次サーバーが稼働したことを確認した後、1 次サーバーを 2 次サーバーに接続するコマンドを実行することだけです。完全コピーが必要な場合、Watchdog は 1 次サーバーに netcopy (または copy) 操作を実行するよう指示できます。完全コピーでは、2 次サーバー上のすべてのデータが上書きされることに留意してください。

ネットワークを介した 1 次データベースの 2 次側へのコピー

データベース・ファイルのコピーを 1 次サーバーから 2 次サーバーへ送信するには、**netcopy** コマンドを使用します。2 次サーバーは、既に稼働している必要があります。

始める前に

重要: コマンドを実行するには、1 次サーバーが PRIMARY ALONE 状態でなければなりません。

このタスクについて

netcopy を使用して 2 次サーバー用のデータベース・コピーを作成する状況には、主に以下の 2 つがあります。

- データベースを保持したことがない完全に新規の 2 次サーバー用に、データベースを作成する場合。(この方法は、ディスクレスの 2 次サーバーにデータベースをコピーするときにも使用されます。そのような 2 次サーバーは、障害を起こした後、データベース全体を失っており、完全に新規の 2 次サーバーとして取り扱う必要があるからです。)
- 既存の 2 次側のデータベース (例えば、破損したもの) を置き換える場合。

手順

netcopy を実行するためのコマンドは以下のとおりです。

```
ADMIN COMMAND 'hotstandby netcopy';
```

1 次側は、**netcopy** を行う場合、solid.ini の [HotStandby] セクションで指定された接続ストリングを使用します。
接続ストリングを定義する **Connect** パラメーターの詳細については、 42 ページの『3.2.1, 1 次および 2 次ノード HotStandby 構成の定義』を参照してください。
hotstandby netcopy コマンドを実行すると、1 次側のデータベースのコピーを送信

する前に、データベース・チェックポイントが実行されます。
1 次側は netcopy の間も、トランザクションと接続の受け入れを続行します (ただし、サーバーの状態を変更する ADMIN COMMAND は、すべてリジェクトされます)。 2 次側は、トランザクションと接続の受け入れを続行しません。**netcopy** が開始されると、2 次側は、オープンの接続またはトランザクションがある場合には、オープン・トランザクションをロールバックし、クライアントからの接続を切断してから、**netcopy** の受信を開始します。 2 次側は、**netcopy** を受信している間、1 次サーバーとだけ通信します。
netcopy が正常に完了すると、2 次サーバーの状態は SECONDARY ALONE に変更されます (まだその状態になっていない場合)。
1 次サーバーは、netcopy 操作の間、PRIMARY ALONE 状態のままです。 **netcopy** が正常に完了した後、1 次サーバーは引き続き同じ状態のままです。完全なホット・スタンバイ動作を再開できるようにするには、事前に 1 次サーバーと 2 次サーバーを接続する必要があり、それによって、1 次サーバーは PRIMARY ACTIVE 状態に設定されます。2 つのサーバーの接続方法については、65 ページの『3.4.6, HotStandby サーバーの接続』を参照してください。

2 次サーバー用の新規データベースの作成

通常、solidDB サーバーを始動すると、新規データベースを作成するかどうかを尋ねられます (まだデータベースが存在しない場合)。しかし、サーバーが 2 次サーバーである場合は、独自のデータベースを作成せずに、1 次サーバーのデータベースのコピーを使用してください。したがって、既存のデータベースがない 2 次サーバーを始動するときは、コマンド行パラメーターを指定して、1 次サーバーからデータベースのコピーを受信するまで待つよう指示する必要があります。そのコマンド行パラメーターは、**-x backupserver** です。例えば、以下のコマンドで 2 次サーバーを始動します。

```
solid -x backupserver
```

「-x」と「backupserver」の間のスペースは、なくてもかまいません。以下のようにしても同じことです。

```
solid -xbackupserver
```

-x backupserver コマンド行パラメーターは、サーバーに「ネットコピー *listen* モード」(「バックアップ *listen* モード」とも呼ばれます) に入るよう指示します。このモードでは、2 次サーバーに可能な唯一の操作は、1 次サーバーからデータベース・コピーを受信することです。2 次サーバーは、それ以外のどのコマンドにも応答せず、実際に、solsql などのクライアント・プログラム、アプリケーション、または Watchdog プログラムからの接続要求さえ受け入れません。

2 次データベースが存在する場合は、通常の方法でサーバーを始動でき、その場合、サーバーは SECONDARY ALONE 状態になります。

2 次サーバーを **-x backupserver** で始動した後、または 2 次サーバーが SECONDARY ALONE 状態にある場合は、1 次サーバー上で **netcopy** コマンドを実行できます。

最初に、1 次サーバーが PRIMARY ALONE 状態にあることを確認してください。次に、1 次サーバー上で以下のコマンドを発行します。

```
ADMIN COMMAND 'hsb netcopy';
```

1 次サーバー上で、**hotstandby netcopy** コマンドは solid.ini 構成ファイル内の **connect** パラメーターで定義された接続ストリングを使用して、2 次サーバーに接続します。接続後、ネットワーク・リンクを通じてデータベース・ファイルがコピーされます。ネットコピー listen モードでは、2 次サーバーは 1 次サーバーでの **hotstandby netcopy** コマンドを通じて新規データベース・コピーを受信した後、2 次側のデータベースを開くことだけを試みます。

以下に、2 次側のデータベース・コピーを作成する手順を示します。

1. solid.ini ファイルを、HotStandby 構成に有効になるように構成してあることを確認します。接続ストリングを定義する **Connect** パラメーターの詳細については、42 ページの『3.2.1, 1 次および 2 次ノード HotStandby 構成の定義』を参照してください。

   この接続ストリングは、1 次サーバーから 2 次サーバーに接続し、ネットワークを通じてデータベース・ファイルをコピーするために使用されます。
2. 1 次サーバーを始動します。
3. 以下のコマンドを実行して、2 次サーバーをネットコピー listen モードで始動します。

   ```
   solid -x backupserver
   ```

   あるいは、別の方法として、既存のデータベースを使用して 2 次サーバーを始動することもできます。
4. 1 次サーバーを PRIMARY ALONE 状態に設定します (まだその状態でない場合)。

   ```
   ADMIN COMMAND 'hotstandby set primary alone';
   ```
5. 1 次サーバーで、以下のコマンドを発行します。

   ```
   ADMIN COMMAND 'hotstandby netcopy';
   ```
6. **netcopy** が完了した後、2 つのサーバーを接続し、以下のコマンドで完全なホット・スタンバイ操作を開始 (または再開) できます。

   ```
   ADMIN COMMAND 'hotstandby connect';
   ```

2 次サーバーは、ネットワーク・リンクを通じてデータベースの新規コピーを受信すると、その 2 次側のデータベースを開きます。サーバー同士が (**hsb connect** コマンドで) 接続した後、2 次サーバーは通常の SECONDARY ACTIVE 状態で稼働し、1 次サーバーからユーザー・トランザクションを受け入れることができる状態になります。

HAC を使用した場合、1 次サーバーのデータベースを 2 次サーバーにコピーする手順は、以下のとおりです。

1. 各サーバーが適正な接続パラメーターを備えていることを確認します。接続ストリングを定義する **Connect** パラメーターの詳細については、42 ページの『3.2.1, 1 次および 2 次ノード HotStandby 構成の定義』を参照してください。
2. 1 次サーバー・ノードの HAC に、solidhac.ini 内で **PreferredPrimary**=Yes パラメーターが設定されており、2 次サーバー・ノードの HAC には設定されていないことを確認します。HAC の構成の詳細については、47 ページの『3.3, HA コントローラーおよび HA マネージャーの構成』を参照してください。

3. HAC のインスタンスを開始するか、オプションとして、HAC を AUTOMATIC モードに設定します。

注:

SECONDARY ALONE 状態にあるサーバーに **netcopy** が送信された場合、既存のデータベースはコピーされたデータベースによって上書きされます。このオプションは、データベース同士を再同期させる必要がある場合、または破損した 2 次側のデータベースを修復する必要がある場合に便利です。

2 次サーバー上の既存データベースの置き換え

netcopy は主に、それまでにデータベースがなかった 2 次サーバーにデータベースを送信するために使用されますが、それ以外の状況でも **netcopy** を使用できます。例えば、2 次サーバーのデータベースのコピーが (ディスク・ドライブの破損などによって) 壊れた場合、**netcopy** コマンドを使用して、2 次サーバーに 1 次サーバーのデータベースを送信できます。

始める前に

既存のデータベースを置き換える場合、2 次サーバーを「ネットコピー listen モード」にする必要はありません。つまり、2 次サーバーを **-x backupserver** で始動する必要はありません。

手順

1. 1 次サーバーが PRIMARY ALONE 状態で、2 次サーバーが SECONDARY ALONE 状態であることを確認します。
2. 1 次サーバーに以下のコマンドを発行します。

   ```
   ADMIN COMMAND 'hotstandby netcopy';
   ```

   netcopy が完了した後、1 次サーバーは引き続き PRIMARY ALONE 状態で、2 次サーバーは自動的に SECONDARY ALONE 状態に置かれる (まだその状態になかった場合) ことに注意してください。

   2 次サーバーが SECONDARY ALONE 状態にあるときに **netcopy** を実行した場合、2 次サーバーにクライアントが (読み取り専用照会を行うために) 接続していると、2 次サーバーはすべてのオープン・トランザクションをロールバックし、すべてのクライアント接続を切断します。 **netcopy** が完了した後、2 次サーバーは SECONDARY ALONE 状態のままです。
3. サーバーは自動的には接続しないので、以下のコマンドを発行する必要があります。

   ```
   ADMIN COMMAND 'hotstandby connect';
   ```

ネットコピー状況の検証

netcopy コマンドを開始すると、バックグラウンドで非同期にネットコピーが実行されます。サーバーは **netcopy** の完了時にメッセージを表示しません。代わりに、ユーザーが **netcopy** の状況をモニターする必要があります。

このタスクについて

サーバーはネットワーク・エラーなどの問題で **netcopy** が失敗した場合でも、メッセージを表示しません。

手順

1 次サーバーで、以下のコマンドを発行します。

```
ADMIN COMMAND 'hotstandby status copy';
```

注: このコマンドでは、キーワードに「netcopy」でなく「copy」を使用します。コピー操作とネットコピー操作の両方に同じコマンドを使用します。
このコマンドは、**netcopy** が成功したのか、まだ進行中なのか、あるいは、失敗したのかを知らせる状況メッセージを表示します。失敗の場合は、エラー・コードとエラー・メッセージで示します。

1 次サーバーから指定されたディレクトリーへのデータベース・ファイルのコピー

2 次側のデータベースに使用されるディレクトリーが 1 次側に可視の場合は、**hotstandby copy** コマンドを使用してデータベースを 1 次側のディレクトリーから 2 次側のディレクトリーにコピーできます。

始める前に

このタスクを実行できるのは、1 次サーバーと 2 次サーバーがいくつかの同じディスク・ドライブを認識でき、したがって、いくつかの同じディレクトリーの読み取りと書き込みができる場合だけです。

注意:
hotstandby copy コマンドを使用する前に、必ず 2 次サーバーをシャットダウンしてください。1 次サーバーがデータベース・ファイルに書き込みを行っている間、2 次サーバーは、そのデータベース・ファイルにアクセスを試みてはなりません。

注: hotstandby copy コマンドを発行するとき、1 次サーバーは PRIMARY ALONE 状態にあることが必要で、コマンドの実行中 (およびコマンドの後)、1 次サーバーはその状態のままになります。

このタスクについて

hotstandby copy コマンドと **hotstandby netcopy** コマンドの主な相違点の 1 つは、**netcopy** コマンドが 2 次サーバーの実行中にのみ使用できるのに対し、**copy** コマンドは 2 次サーバーの実行中以外にのみ使用する必要があることです。パフォーマンスの面では、この 2 つのデータベース・コピー方式に大きな違いはありません。

手順

1. **hotstandby copy** を使用してファイルをコピーするには、1 次サーバー側で次のコマンドを発行します。

   ```
   ADMIN COMMAND 'hotstandby copy[directory_name]';
   ```

ここで、*directory_name* はファイルのコピー先にしたいディレクトリーの名前です。ディレクトリー名の形式は、オペレーティング・システムによって異なります。ディレクトリー名はオプションです。ディレクトリー名を指定しなかった場合、サーバーは solid.ini 構成ファイル内の **CopyDirectory** パラメーターで指定された値を使用します。

hotstandby copy コマンドを実行すると、データベースにチェックポイントが作成され、次に 1 次側のデータベースのコピーが作成された後、そのコピーが 2 次側へ送信されます。

サーバーは PRIMARY ALONE 状態にあるため、1 次側でのトランザクション処理は **copy** コマンドの間も通常どおり続行され、1 次側はトランザクションをトランザクション・ログに保管して、後で 2 次側へ転送できるようにします。

2. コピー操作の後も、2 次側はまだダウンしたままです。それを復帰させた後に、**hotstandby connect** コマンドを発行して、2 つのサーバーを接続する必要があります。

   1 次サーバーのデータベースが管理コマンドの **hotstandby connect** を使用して 2 次側に接続されると、1 次と 2 次のサーバーは自動的に「キャッチアップ」を実行して、2 次サーバーを最新の状態にします。

2 次サーバーの始動とキャッチアップ:

コピーが完了したら、新規にコピーしたデータベースを使用して 2 次サーバーを始動する必要があります。

手順

1. 通常と同じ方法でサーバーを始動します。つまり、オペレーティング・システムのプロンプトでコマンド「**solid**」を発行します。
2. 2 次サーバーを再始動した後、1 次サーバーで **hotstandby connect** コマンドを使用して、1 次サーバーを 2 次サーバーに接続します。

   ```
   ADMIN COMMAND 'hotstandby connect';
   ```

   hotstandby connect コマンドの詳細については、『3.4.6, HotStandby サーバーの接続』に説明があります。

タスクの結果

1 次サーバーを 2 次サーバーに接続した後、1 次サーバーと 2 次サーバーは自動的にキャッチアップの実行を開始します。これは、1 次サーバーが 2 次サーバーのデータベースを最新のものにするために、1 次サーバーのトランザクション・ログを 2 次サーバーへコピーし、2 次サーバーは、トランザクション・ログをロールフォワードして、自身のデータベースのコピーを更新することを意味しています。

3.4.6 HotStandby サーバーの接続

このタスクについて

1 次サーバーが 2 次サーバーに接続するために使用する connect ストリングは、solid.ini 構成ファイルの [hotstandby] セクションで **Connect** パラメーターによって指定します。

1 次側ノードと 2 次側ノードの現行の接続設定は、次のコマンドを発行することによって表示できます。

```
ADMIN COMMAND 'hotstandby cominfo';
```

手順

1 次側と 2 次側の間の接続が切れているか、まだ確立されていない場合は、以下のコマンドを 1 次側または 2 次側のノードで発行する必要があります。

```
ADMIN COMMAND 'hotstandby connect';
```

例えば、**netcopy** を実行した後、通常はサーバー同士を接続します。
HotStandby サーバーには自動接続メカニズムがないため、サーバー間の接続が切れているときは、高可用性制御アプリケーションにこのコマンドを実行させてください。
このコマンドを発行した後、1 次サーバーと 2 次サーバーの接続が成功した場合は、確認メッセージが表示されます。1 次サーバーと 2 次サーバーが接続されていても、トランザクション・ログが 2 次側でまだ完全にはコピーされていない場合、ユーザーは 1 次サーバーから「`Started the process of connecting the servers`」というメッセージを受け取ります。
コマンドを実行したときに 1 次サーバーの状態が PRIMARY UNCERTAIN または PRIMARY ALONE であった場合、接続が成功すると、1 次サーバーの状態が PRIMARY ACTIVE に変わります。成功しないと、状態は PRIMARY UNCERTAIN または PRIMARY ALONE のままです。

次のタスク

1 次サーバーと 2 次サーバーで接続状況を照会する方法について詳しくは、67 ページの『接続状況情報の表示』を参照してください。アプリケーションを 1 次サーバーに再接続する方法の詳細については、103 ページの『4.3.1, アプリケーションから 1 次サーバーへの再接続』を参照してください。

3.4.7 HotStandby 状況の検査

このセクションでは、1 次サーバーと 2 次サーバーの両方に対して要求できる HotStandby 状況情報について説明します。

手順

状況を確認するには、1 次または 2 次サーバーで次のコマンドを発行します。

```
ADMIN COMMAND 'hotstandby status option';
```

ここで、*option* は、以下のいずれかを指定できます。

オプション	説明
catchup	サーバーがキャッチアップを実行するかどうかを指示します。キャッチアップは、1 次サーバーが 2 次サーバーに接続した後に行われます。キャッチアップの間、1 次サーバーは 2 次サーバーが変更を適用できるよう、蓄積したトランザクション・ログを送信します。可能な値は、「ACTIVE」と「NOT ACTIVE」です。
connect	前回サーバー同士を接続しようとして成功したかどうかを示します。
copy	前回の **copy/netcopy** の試みが成功したかどうかを示します。
switch	前回、サーバーを PRIMARY ACTIVE 状態または SECONDARY ACTIVE 状態に切り替えようとして成功したかどうかを示します。

例

```
ADMIN COMMAND 'hotstandby status catchup';
```

切り替え状況情報の表示

2 つの HotStandby サーバー間で、状態切り替えが発生したかどうかを検証する必要が生じる場合があります。

このタスクについて

HotStandby 切り替え状況情報を検査するには、以下のようにします。

手順

1 次または 2 次サーバーで、以下のコマンドを発行します。

```
ADMIN COMMAND 'hotstandby status switch';
```

- 以前に 2 つのサーバー間で切り替えが発生したことがない場合は、NO SERVER SWITCH OCCURRED BEFORE というメッセージが表示されます。
- 切り替えプロセスがまだアクティブである場合は、ACTIVE というメッセージが表示されます。
- 以前の切り替えプロセスのうち最新のものが正常に完了している場合は、SUCCESS というメッセージが表示されます。
- 最新の切り替えの試みが失敗している場合は、ERROR *number* というメッセージが表示されます。ここで、*number* は切り替えのときに発生したエラーのタイプを示しています。

接続状況情報の表示

1 次サーバーと 2 次サーバーの間の接続状況情報を照会できます。この機能は、アプリケーション・コード内で使用できる SQL 関数 HOTSTANDBY_CONNECTSTATUS と同じものです。

手順

接続状況を確認するには、1 次サーバーまたは 2 次サーバーで次のコマンドを発行します。

```
ADMIN COMMAND 'hotstandby status connect';
```

可能な戻り値は、以下のとおりです。

表 7. 接続状況の戻り値

エラー・コード	テキスト	説明
0	CONNECTED	接続アクティブ。1 次と 2 次の両方のサーバーから返されます。
14007	CONNECTING	1 次サーバーが 2 次サーバーに接続中です。1 次と 2 次の両方のサーバーから返されます。
14008	CATCHUP	1 次サーバーが 2 次サーバーに接続しましたが、トランザクション・ログはまだ完全にはコピーされていません。1 次と 2 次の両方のサーバーから返されます。
14010	DISCONNECTING	サーバーは切断処理中です。
14537	BROKEN	接続が切れています。1 次と 2 次の両方のサーバーから返されます。

通信情報の表示

他のサーバーに接続するために使用した通信情報を照会できます。これは、solidDB 構成ファイル (solid.ini) の [HotStandby] セクションにある **Connect** パラメーター設定の値です。この情報をクライアント・アプリケーションで使用して、他のサーバーに接続できます。

手順

通信情報を表示するには、1 次または 2 次サーバーで次のコマンドを発行します。

```
ADMIN COMMAND 'hotstandby cominfo';
```

ロール開始時刻の表示

場合によっては、いつサーバーが現在の 1 次サーバーまたは 2 次サーバーのロールに入ったかを知ることが重要になります。

この情報をリトリーブするには、以下のような **ADMIN COMMAND 'info'** の 2 つの対応するオプションを使用します。

```
admin command 'info primarystarttime';
       RC TEXT
       -- ----
        0 2005-06-09 14:22:18
```

```
admin command 'info secondarystarttime';
       RC TEXT
       -- ----
        0 2005-06-09 18:24:44
```

報告される時刻は、ロールが 1 次サーバーまたは 2 次サーバーになった時刻です。STANDALONE 状態は、1 次サーバー・ロールの状態であると考えられます。

具体的には、1 次サーバー開始時刻は、以下の遷移が発生したときに設定されます。

- SECONDARY ALONE => PRIMARY ALONE
- SECONDARY ALONE => STANDALONE
- SECONDARY ACTIVE => PRIMARY ACTIVE

2 次サーバー開始時刻は、以下の時点で設定されます。

- サーバーが SECONDARY ALONE 状態で始動されたとき
- OFFLINE (**-x backupserver** で始動) => SECONDARY ALONE
- PRIMARY ALONE => SECONDARY ALONE
- STANDALONE => SECONDARY ALONE
- PRIMARY ACTIVE => SECONDARY ACTIVE

現在のロールが照会と矛盾する場合、照会は空ストリングを返します。例えば、ロールが SECONDARY で、コマンド 'info primarystarttime' が発行された場合は、空ストリングが返されます。

3.4.8 HotStandby サーバー状態の検証

HotStandby の管理または保守を行うときは、多くの場合、HotStandby サーバーの状態を確認する必要があります。

手順

HotStandby サーバーの現在の状態を確認するには、そのサーバーで以下の HotStandby コマンドを発行します。

```
ADMIN COMMAND 'hotstandby state';
```

次のタスク

コマンドが返す可能性がある状態については、9 ページの『サーバー状態の説明』を参照してください。

注:

1. サーバーの `solid.ini` 構成ファイルが、そのサーバーを HotStandby サーバーにするように構成されている場合、そのサーバーは、始動された時点では SECONDARY ALONE 状態で始動します。
2. 上記の表に OFFLINE 状態がリストされていないことに注意してください。サーバーは、状態名「OFFLINE」に戻ることはできません。サーバーが OFFLINE 状態になると、それに接続して照会 (例えば、ADMIN COMMAND 'hotstandby state' など) を発行できなくなるからです。

3. HotStandby 用に構成されていないサーバー上で ADMIN COMMAND 'hotstandby state' を発行すると、以下のエラー・メッセージが返されます。

```
14527: This is a non-HotStandby Server
```

   HAC および HA マネージャーを使用している場合、HA マネージャーは両方のサーバーの HSB 状態を表示します。この情報は、秒刻みで定期的に照会されます。

管理操作およびトラブルシューティング操作の実行中に生じる HotStandby 状態遷移の概要については、177 ページの『付録 D. サーバー状態遷移』のセクションを参照してください。

サーバー状態の組み合わせ

サーバー状態のすべての組み合わせが可能なわけではありません。例えば、2 次サーバーは 1 次サーバーが PRIMARY ACTIVE 状態の場合、SECONDARY ACTIVE 状態にのみなることができます。

以下の表は、関連付けられているサーバーが特定の状態にあるときに、HotStandby サーバーに可能となるサーバー状態を示しています。

表 8. サーバー状態

サーバーの状態	関連付けられているサーバーに可能な状態
PRIMARY ACTIVE	SECONDARY ACTIVE
PRIMARY ALONE	PRIMARY ALONE * PRIMARY UNCERTAIN SECONDARY ALONE STANDALONE *
PRIMARY UNCERTAIN	PRIMARY ALONE PRIMARY UNCERTAIN SECONDARY ALONE STANDALONE
SECONDARY ACTIVE	PRIMARY ACTIVE
SECONDARY ALONE	PRIMARY ALONE PRIMARY UNCERTAIN SECONDARY ALONE STANDALONE
STANDALONE	PRIMARY ALONE * PRIMARY UNCERTAIN SECONDARY ALONE STANDALONE *

* 一方のサーバーが PRIMARY ALONE 状態または STANDALONE 状態の場合、もう一方のサーバーが PRIMARY ALONE 状態または STANDALONE 状態にならないようにしてください。その理由は、両方のサーバーに別々の変更が加えられた場合、2 つのデータベースを 1 つにマージする方法がないためです。

3.4.9 どちらのサーバーを 1 次サーバーにするかの選択

エラー・リカバリー状態によっては、1 次サーバーにする必要があるサーバーを判断できないことがあります。この場合は、**hsb logpos** コマンドを使用して、どちらのサーバーが 1 次サーバーになるかを判断することができます。

このタスクについて

この手順は、例えば、両方のデータベースで障害が発生した場合に使用できます。この場合、サーバー同士の相互の接続が失われる前に 1 次サーバーだったサーバーが、必ずしもその時点で 1 次サーバーになる必要があるサーバーとは限りません。

手順

1. 両方のサーバーが SECONDARY ALONE 状態であることを確認します。
2. 両方のサーバーに接続します。
3. 各サーバーで、以下を実行します。

   ```
   ADMIN COMMAND 'hsb logpos';
   ```

   このコマンドは、前回の操作時のログ操作 ID およびサーバーのロール (PRIMARY、SECONDARY、STANDALONE) を返します。 成功した管理コマンドは、エラー・コード 0 とストリング、およびサーバーの前のロールを返します。

   以下に例を示します。

   ```
   ADMIN COMMAND 'hsb logpos';
   RC TEXT
   -- ----
   0 00000000000000000871:PRIMARY
   ```

   アプリケーションは、このストリングを、定義された構造を持たず、内部が見えない値と見なす必要があります。
4. ストリング値を比較します。

   例えば、C では strcmp() 関数を使用します。

   原則として、ログ操作 ID の値が大きいサーバーがより多くのトランザクションを受け入れてきているため、これを 1 次サーバーにすべきです。しかし、HSB 接続で障害が発生した後に両方のサーバーを更新した場合には、ログ操作 ID の値を比較しても信頼性はありません。

   HSC が使用されていてストリングが等しい場合は、**LocalDB.PreferredPrimary** HAC パラメーターによって、ローカル・サーバーが 1 次サーバーになるべきかどうかが定義されます。
5. 1 次サーバーになるサーバー上で、以下のコマンドを使用して 1 次サーバーを選択します。

   ```
   ADMIN COMMAND 'hsb set primary alone';
   ```
6. 以下のコマンドを使用して、HotStandby サーバーを互いに接続します。

   ```
   ADMIN COMMAND 'hsb connect';
   ```

タスクの結果

- **hsb connect** コマンドが成功した場合、2 次サーバーは 1 次サーバーをキャッチアップし、HotStandby ペアが再び機能するようになります。
- **hsb connect** コマンドが失敗した場合は、ノードを別々に同期する必要があります。詳細については、56 ページの『3.4.5, 1 次サーバーと 2 次サーバーの同期』を参照してください。

注意:
この手順では、大きい方のストリング値を持つサーバーが他方のサーバーのスーパーセットになることは保証されません。2 つのサーバーのそれぞれが、他方が受け入れなかったトランザクションを受け入れた可能性が存在し、例えば、両方のサーバーが PRIMARY ALONE 状態で稼働していた可能性があります。各サーバーは、どちらのサーバーも他方のスーパーセットでない可能性を検出するために、hsb connect コマンドを実行するときに情報を比較します。どちらのサーバーもスーパーセットではないと、hsb connect コマンドは失敗し、該当するエラー・メッセージが出力されます。

3.4.10 HotStandby サーバーから非 HotStandby サーバーへの変更

solid.ini ファイルの [HotStandby] セクションを編集することにより、1 次サーバーまたは 2 次サーバーを通常の非 HotStandby サーバーに変更できます。

手順

1. **HSBEnabled** パラメーターを除去 (または、no に設定) します。
2. (オプション) **Connect** パラメーターを除去またはコメント化します。
3. solid.ini ファイル内のパラメーター設定を変更した後、変更を有効にするためには、サーバーを再始動する必要があります。

例

一時的に HotStandby サーバーとしての機能を停止し、後で HotStandby サーバーとしての機能を再開させたい場合は、solid.ini ファイルを変更しないでおき、代わりに、サーバーの状態を単に STANDALONE に変更することもできます。

3.5 パフォーマンスのチューニング

3.5.1 安全性レベルと持続性レベルによるレプリケーション・パフォーマンスのチューニング

通常操作時のデータ・レプリケーションのパフォーマンスは、持続性レベルと安全性レベルの設定によって異なります。また、2-safe レプリケーションを使用した場合、2-safe モードで使用される確認応答ポリシーは、アプリケーションによって感知される待ち時間に影響します。詳細については、20 ページの『1.2, パフォーマンスと HotStandby』を参照してください。

3.5.2 ネットコピーのパフォーマンスのチューニング (General セクション)

hotstandby **netcopy** コマンドを使用すると、1 次サーバーのデータベースをリモート 2 次サーバーにコピーできます。このコマンドは、一方または両方のサーバーがディスクレスであるときに、データベースを 1 次サーバーから 2 次サーバーにコピーするためにも使用されます。ネットコピー用に 2 次サーバーに接続するために使用する接続ストリングは、solid.ini の [HotStandby] セクションで指定されます。

1 次サーバーのデータベース・ファイルは、ネットワーク・リンクを通じてコピーされます。**netcopy** の詳細については、60 ページの『ネットワークを介した 1 次データベースの 2 次側へのコピー』を参照してください。

solid.ini ファイルの [General] セクションにある以下のパラメーターは、ネットコピーのパフォーマンスを向上させる効果があります。

General.BackupBlockSize パラメーター

General.BackupBlockSize パラメーターは、1 次サーバーのデータベース・ファイルを 2 次サーバーにコピーするときのブロック・サイズを増減することにより、ネットコピーのパフォーマンス (およびバックアップのパフォーマンス) をチューニングするために使用されます。一般的なルールとして、ブロック・サイズが大きいほどネットコピー/バックアップは高速になりますが、その代償として、ネットコピー/バックアップが行われている間、他の要求に対するサーバーの応答時間が低下する可能性があります。

デフォルトでは、**BackupBlockSize** パラメーターは 64K に設定されます。これは、別の値に設定できます。以下に例を示します。

```
[General]
BackupBlockSize = 32K
```

または

```
[General]
BackupBlockSize = 32768
```

接尾部「M」および「K」がサポートされます。例えば、32K や 1M などです。

注:

- **BackupBlockSize** の最小値はサーバーのブロック・サイズです。
- 最大値は、8MB です。solid.ini 内のパラメーター値がこの最大値を超える場合は、デフォルト値が使用されます (64K)。
- **BackupBlockSize** の値は、サーバーのデータベース・ブロック・サイズの倍数にする必要があります。

データベース・キャッチアップのパフォーマンスのチューニング

HotStandby では、障害を起こした 2 次サーバーがサービスに復帰し、1 次サーバーに接続した場合、「*HotStandby* データベース・キャッチアップ」と呼ばれる自動プロセスにより、HotStandby トランザクション・ログ・ファイルの内容が 1 次サーバーから 2 次サーバー・ノードへ引き続き送信されます。

solid.ini ファイルの [HotStandby] セクション内の **CatchupSpeedRate** パラメーターは、キャッチアップと現在のクライアント・データベース照会に対するサービスに使用するサーバーの時間を調整することにより、データベース・キャッチアップのパフォーマンスをチューニングするために使用されます。

CatchupSpeedRate に値 90 を割り当てた場合、サーバーはその時間の約 90% をキャッチアップに使用し、約 10% の時間をユーザーの照会に対する応答に使用します。例えば、以下のように指定します。

```
[HotStandby]
CatchupSpeedRate = 50
```

この数値が大きいほど、高速なキャッチアップが実行されますが、ユーザー照会など、他のアクティビティーに及ぼす影響が大きくなります。デフォルトでは、**CatchupSpeedRate** は 70 に設定されます。

3.6 HotStandby で solidDB を使用する場合の特別な考慮事項

HotStandby で solidDB を使用する場合は、トランザクション分離レベル、トランザクション・ログ・スペース、計画されていないネットワーク分割、およびスロットルに特に注意する必要があります。

3.6.1 トランザクション分離レベルとインメモリー表

2 次サーバーに接続している場合に、インメモリー表からデータを読み取ると、トランザクション分離レベルは、REPEATABLE READ または SERIALIZABLE として指定した場合でも、自動的に READ COMMITTED に設定されます。

さらに、ロード・バランシングを使用している場合、分離レベルはデフォルトで READ COMMITTED になります。

3.6.2 ネットワーク分割と二重 1 次サーバー

環境によっては、両方のサーバーが PRIMARY ALONE 状態で動作する可能性があります。*二重 1 次サーバー* があると、深刻なリカバリー不能エラーにつながることがあります。

この状況では、いずれかのトランザクションを一方のサーバーだけがコミットした場合、サーバー同士を再同期させることができません。各データベースを「マージ」して、正しい情報が入った単一のデータベースを作成する方法が存在しないからです。実際には、二重 1 次サーバーの状況で「間違った 1 次サーバー」のデータベースにコミットされたトランザクションは、失われます。二重 1 次サーバーがあると、その他のエラーにもつながります。

二重 1 次サーバーの問題の原因として最も可能性が高いものは、「*ネットワーク分割*」です。この状況では、ネットワーク接続の全部ではなく一部が失われ、単一のネットワークが事実上複数の分離された小部分に分割された状態となり、それぞれがその部分の中で通信でき、他の部分と通信できなくなります。このため、両方のサーバーが互いに接続を失いますが、どちらも依然として稼働中であり、場合によっては一部のクライアントとの通信が維持されています。

二重 1 次サーバーのシナリオは、HSB システムの外部にあるノードで単一インスタンスの Watchdog を使用すれば、回避できます。そのような Watchdog を使用すると、どちらのサーバーを 1 次サーバーに設定すればよいかを簡単に判断でき、クライアントが確実に 1 つの 1 次サーバーだけを認識するようにすることも簡単にできます。

HAC は、HSB サーバーを備えた同じノード内で稼働する 2 つのインスタンスから構成されていますが、HAC を ERE と一緒に使用している場合、二重 1 次サーバーの状態は事実上不可能です。

結果的に二重 1 次サーバーが発生した場合でも、誰かが元の 2 次サーバー上で (それが PRIMARY ALONE に切り替わった後に) 書き込み操作を実行できる場合を除いて、実際に矛盾するデータが生じることはありません。元の 2 次サーバーがネットワークのその他の部分から完全に切り離された場合、誰もそれに書き込むことはできず、元の 1 次サーバーは 2 次サーバーのスーパーセットとなるので、ユーザーは依然として、(サーバー同士を再接続して 2 次サーバーに元の 1 次サーバー上で行われた変更のキャッチアップを許可すれば) 整合性のある単一のデータ・セットを取得できます。

二重 1 次サーバーはまれなことではありますが、発生すると極めて危険なので、データが不整合にならないように最大の注意を払う必要があります。ERE を使用することを強くお勧めします。

一方または両方のサーバーの `solid.ini` ファイルで構成パラメーター **`AutoPrimaryAlone`**`=Yes` を設定してあると、二重 1 次サーバーが生じる可能性が高くなります。**`AutoPrimaryAlone`**`=Yes` を使用することは、システムが障害に対して、より迅速に対応できることを意味しますが、同時に、二重 1 次サーバーを防止するための独立した監視者 (HAC、Watchdog、または人間) がいなくなることも意味しています。ネットワークの信頼性にいくらかでも疑いがあるときは、**`AutoPrimaryAlone`** パラメーターをファクトリー値の `No` のままにしておいてください。

3.6.3 トランザクション・ログのスペース不足

HotStandby を使用する場合、サーバーを PRIMARY ALONE 状態にするときは、トランザクション・ログのディスク・スペースが不足しないように注意する必要があります。非 HotStandby サーバーでは、頻繁にチェックポイントを指定した場合、トランザクション・ログが極端に大きくなることはありません。サーバーは各チェックポイントの後に、そのつど「古い」トランザクション・ログを削除するからです。

特に、非 HotStandby サーバーは、チェックポイントの前に発生したデータ変更を持つログを削除します。チェックポイントの詳細については、「*solidDB 管理者ガイド*」を参照してください。

しかし、PRIMARY ALONE 状態で動作している HotStandby サーバーでは、サーバーは 1 次側と 2 次側との接続が失われた時点以降に蓄積したトランザクション・ログを保持する必要があります。2 次側が長い時間にわたってダウンした場合、サーバーは、正常に稼働していれば各チェックポイントの後に放棄するはずの大量のトランザクション・ログ・データを、保持し続ける可能性があります。最悪の場

合、2 次側が合理的な時間内に復帰できず、発生するすべてのトランザクションを保管するために十分なディスク・スペースがないと、1 次側のトランザクション・ログが、使用可能なすべてのディスク・スペースを埋め尽くす可能性があります。その場合、サーバーは読み取り専用モードに切り替わります。

これを防ぐために、[HotStandby] セクションでパラメーター **MaxLogSize** の適切な値を設定できます。指定した合計ログ・サイズに到達すると、サーバーは次のチェックポイントで自動的に STANDALONE 状態に切り替わります。(ただし、ディスクレス・サーバーではディスクへの書き込みが存在しないので、状態は PRIMARY ALONE のままです。)

STANDALONE 状態に設定されたサーバーは、1 次側と 2 次側の接続が失われた時点以降のすべてのトランザクション・ログを保持しなくなります。完全なトランザクション・ログがなければ、単に 1 次側を 2 次側に接続し、2 次側で古いログを読み取って「キャッチアップ」しても、システムの同期は取れません。**copy** または **netcopy** コマンドを使用して、データベース全体を 1 次側から 2 次側へコピーする必要があります。

HAC を使用している場合は、HAC が上記の状態を識別し、必要なアクションを自動的に実行して各サーバーを ACTIVE 状態に導きます。

3.6.4 2 次サーバーでのスロットルとマルチプロセッシング

1 次サーバーと 2 次サーバーのデータを最新の状態に維持するため、1 次サーバーは自身のトランザクション・ログに書き込みを行い、そのログを 2 次サーバーに転送して、2 次サーバーが自身のデータベースのコピーに同じ変更を加えられるようにします。 2 次サーバーが 1 次サーバーの処理ペースに対応できない場合は、solidDB のスロットル・メカニズムによって 1 次サーバーの処理速度が遅くなります。アプリケーションの観点から見れば、スロットルを行うと応答時間が長くなります。

V7.0 現在、2 次サーバーは書き込み負荷の処理に複数のスレッドを使用できます。2 次サーバーは、1 回のストリームで 1 次サーバーから複数の操作を受け取ります。操作が解析され、各トランザクションが実行キューにアタッチされます。つまり、オープンの並列トランザクションと同じ数の実行キューがあります。競合する操作が正しい順序で確実に実行されるように、行ロックが使用されます。

以下のパフォーマンス・カウンター (pmon) で、スレッドの使用をモニターできます。

表 9. 2 次サーバーのマルチプロセッシングをモニターするための pmon カウンター

Perfmon 変数	**説明**
HSB grpcommits	最新のグループ・コミット内のトランザクション数 トランザクション・コミットは、1 つのログ・バーストにまとめられ、1 つのパケットとして 2 次サーバーに送信されます。 このカウンターは 1 次サーバーでのみ使用できます。

表 9. 2 次サーバーのマルチプロセッシングをモニターするための *pmon* カウンター *(続き)*

Perfmon 変数	説明
HSB secondary ops in packet	最新のログ・レコード・パケットで 2 次サーバーが 1 次サーバーから受け取ったログ・レコードの数
HSB secondary trx count	2 次サーバーが 1 次サーバーから受け取ったオープン・トランザクションの数
HSB secondary locks	2 次サーバー上の行レベル・ロックの数
HSB secondary lock reqs	2 次サーバー上のロック要求数
HSB secondary lock waits	サーバーが開始してからの 2 次サーバー上のロック待機の数
HSB secondary op waits	2 次サーバー上の操作 (トランザクション) が実行の続行を待機した回数
HSB secondary buffers	2 次サーバーが 1 次サーバーから受け取ったバッファー内のログ・レコード・パケットの数
HSB secondary serial mode count	2 次サーバーの並行実行プログラムが並行に実行するのではなく逐次モードに切り替えた回数
HSB secondary dispatch queuelength	2 次サーバー上の最新のディスパッチ・スレッド (ディスパッチする操作) のサイズ

また、2 次サーバーのマルチプロセッシング・レベルは **HotStandby.SecondaryThreads** パラメーターで最適化することもできます。スレッドの最適な数は、環境によって異なります。環境に一般的な負荷で経験することが必要です。基本的に、コンピューターの処理能力が高い (コアが多い) ほど、**HotStandby.SecondaryThreads** パラメーターに高い値を設定できます。ただし、スレッドを使用しすぎると、パフォーマンスが向上しない可能性があります。

3.7 コスト削減のための構成と安全性向上のための構成の対比

solidDB の HotStandby ソリューションでは、1 次サーバーと 2 次サーバーのペアを使用して真の高可用性を提供します。ただし、サーバーのペアを使用しても、すべての使用シナリオが最適化されるとは限りません。瞬時に近いフェイルオーバーが必要ない場合は、すべての 1 次サーバーに 2 次サーバーを設けることは、その費用を正当化できないかもしれません。それとは反対に、ビジネス・ケースの中には特別な信頼性を必要とし、「スペアのスペア」を購入するための資金を認可することもあります。スペアのスペアとは、すべての 1 次サーバーに対して 2 次サーバーを購入するだけでなく、さらに 1 台以上の追加のスペア・サーバーを購入できる場合があるということです。このようにすれば、1 次サーバーで障害が発生し、対応する 2 次サーバーがそれに置き換わった場合に、元の 1 次サーバーを速やかに修復できなくても、スペアを「新しい 2 次サーバー」として使用できます。

コストを削減したり、信頼性を向上したりできるように、solidDB HotStandby は、いわゆる +N または 2N の標準的なホット・スタンバイ・モデルに代わる各種の代替モデルをサポートしています (標準モデルでは、1 次サーバーと 2 次サーバーの数は同一 (N) です)。代替モデルには、以下のものがあります。

- 「N + 1 スペア」または「N + M スペア」: これは、スタンドアロン用のスペア・ノードのシナリオです。この場合、N 台の「1 次サーバー」と 1 台以上のスペアが存在します。2 次サーバーは存在しません。障害を起こした「1 次」サーバーは、スペアによって置き換えられます。これは真の「ホット・スタンバ

イ」シナリオではなく、むしろ「ウォーム・スタンバイ」と呼ぶ方が当たっています。コンピューターは使用可能ですが、データベースのコピーを持っていないからです。

- 「2N + 1 スペア」または「2N + M スペア」: これは、HotStandby 用のスペア・ノードのシナリオです。この場合、N 個の HotStandby ペアが存在します。つまり、すべての 1 次サーバーごとに 1 台の 2 次サーバーが存在します。さらに、M 台のスペアが存在します。ここで、M は 1 以上で、通常は N 未満です。1 次サーバーまたは 2 次サーバーに障害が起きた場合は、スペアの 1 台が新規 2 次サーバーとして取り込まれます。これにより、1 次サーバーは、たとえ元のパートナーに障害が起きた場合でも、決して単独で長期間動作することはありません。

以下のトピックでは、N+M (または N+1) および 2 N+M (または 2N+1) の手法と、それらを実装するのに役立つ solidDB 機能について説明します。

3.7.1 コストの削減: N + 1 スペアと N + M スペアのシナリオ

これらのシナリオでは、N 台の「1 次」サーバーがあり、そのそれぞれがスタンドアロン状態で作動します。つまり、2 次サーバーに接続されていません。さらに、M 台のスペア・サーバーが存在します。ここで、M は 1 以上で、通常は N 未満です。「1 次」サーバーに障害が起きた場合は、スペアのうちの 1 台がそれに取って代わります。データは元のサーバーからスペアにコピーされ、その後、元のサーバーはオフラインにされ、スペアが元のサーバーとして機能するよう構成されます。すべてのスペアが、すべての 1 次サーバーに取って代わることができる点に注意してください (特定の 1 次サーバー専用のスペアは存在しません)。また、フェイルオーバーがほとんど瞬時に行われるわけではないことにも注意してください。

この手法を、ここでは「*N + 1*」(*単一スペア*) または「*N + M*」(*複数スペア*) シナリオと呼びます。

この手法では、どこかに元のサーバーのデータのコピーを持っている必要があるので、元のサーバーのディスク・ドライブが損傷していて、データのバックアップも存在しない場合、この手法は使用できません。この N+M の手法は、以下の状況で最も便利です。

1. 予期できない障害でなく、定期保守を処理するために、スペア・ノードを使用している場合。
2. スペア・サーバーに素早くコピーでき、信頼できるバックアップを持っている場合。
   a. 磁気テープ上か RAID ドライブ、またはその他の安全な場所にバックアップを持っている場合。または、
   b. solidDB の拡張レプリケーション・テクノロジーを使用しており、障害を起こしたサーバーの拡張レプリケーションの「マスター」または「レプリカ」から、十分なデータをコピーまたは再作成できる場合。
3. データの個々の部分は、重大データでないか固有でない場合。
   a. 例えば、本当に必要なものが特定のデータでなくコンピューターの「計算能力」(負荷分散機能) であるなら、標準または「シード」データベースをコピーするか、クライアントからデータを取得して実行を続けるだけでよい場合もあります。

b. 同様に、すべてのサーバーがほぼ同じデータを持っており、書き込み要求が少ないか 1 つもない「読み取り」要求のほとんどすべてに対応している場合 (例えば、多数のサーバーを実行していて、そのすべてのサーバーが同じインターネット・ルーティング表または電話帳情報を使用している場合) は、どのコンピューターからでも有用なデータベースをコピーできます。

3.7.2 信頼性の向上: 2N + 1 スペアおよび 2N + M スペアのシナリオ

通常の solidDB HotStandby 動作には高い信頼性があります。1 次と 2 次の両方のサーバーがほとんど同時に障害を起こす可能性は、別々の信頼できる電源機構を使用していれば、非常に低くなります。しかし、その危険さえも減らしたい場合、または障害を起こしたサーバーを迅速に修復できない場合は、どうでしょう。理想的には、1 次サーバーが障害を起こし、それを 2 次サーバーで置き換えた場合 (または 2 次サーバーが障害を起こした場合)、完全なサーバー・ペアで実行を続けられるよう、「旧」2 次サーバーに取って代わる「新規」2 次サーバーを持っていたいと考えるでしょう。

その状態を、*2N + 1* スペア (または *2N + M* スペア) のシナリオと呼びます。その場合、N 台の 1 次サーバーと N 台の 2 次サーバー、それに、障害を起こすか 1 次サーバーに変換された 2 次サーバーに取って代わる、少なくとも 1 台のスペアを使用します。各スペアは特定のサーバー (または HSB サーバー・ペア) 専用ではなく、障害を起こしたサーバーをスペアで置き換えられるようにするためには、事前に何らかの構成が必要になります。

3.7.3 solidDB HSB による N+1 (N+M) と 2N+1 (2N+M) の手法のサポート

スペア・サーバーは、置き換えられるサーバーに似たものにする必要があります。一般に、これは以下のことを意味します。

1. データをスペアにコピーする必要があります。
2. 置き換えられるサーバーと同じネットワーク・アドレスか、クライアント・プログラムが通信用に認識している別のアドレスで「listen」を行うよう、スペアに指示する必要があります。
3. さらに、2N+1 (2N+M) のシナリオでは、新規 2 次サーバーと現行 1 次サーバーに相互の通信方法を指示する必要があります。言い換えれば、それぞれのサーバーに、相手との接続に使用するアドレスを指示する必要があります。

これらのニーズをサポートするために、solidDB は次の 2 つの機能を備えています。

- solidDB では、スペア・サーバーをシャットダウンすることなく、スペア・サーバーにデータをコピーできます。
- solidDB では、特定の構成パラメーターを動的に設定できます。

これらについて、以下で詳しく説明します。

1. solidDB の構成パラメーターを設定するには、通常ではサーバーをシャットダウンし、`solid.ini` 構成ファイルを更新し、サーバーを再始動しますが、一部の構

成パラメーター (「com.listen」パラメーターや「hotstandby.connect」パラメーターなど) は、以下のような ADMIN コマンドを実行することによっても変更できます。

```
ADMIN COMMAND 'parameter com.listen="tcp SpareServer1 1315"';
ADMIN COMMAND 'hsb parameter connect "tcp srvr27 1316"';
```

これは、最初にシャットダウンしなくても、スペアを別のサーバーに取って代わるよう動的に構成できることを意味しています。同様に、1 次サーバーに新規 2 次サーバーの Connect ストリングを指示することができます。

注意:

これらのコマンドを実行しても、更新されたパラメーター値は solid.ini ファイルに書き込まれません。したがって、サーバーが次回の再始動時に新しい値を持つようにするには、上記のコマンドを実行するだけでなく、solid.ini ファイルも更新する必要があります。

重要:

スペア・サーバーは、1 次サーバーからのネットコピーを受信できるようにするために、**-x backupserver** コマンド行オプションを使用して始動してください。**-x backupserver** オプションの詳細については、61 ページの『2 次サーバー用の新規データベースの作成』を参照してください。また、「*solidDB 管理者ガイド*」のコマンド行オプションの説明も参照してください。

2. solidDB の「netcopy」コマンドを使用すると、既に稼働中のサーバーにデータベースをコピーできます。
   a. 「connect」パラメーターの新しい値を設定します。

      ```
      ADMIN COMMAND 'hsb parameter connect "tcp srvr27 1316"';
      ```

   b. **netcopy** コマンドを実行します。

      ```
      ADMIN COMMAND 'hsb netcopy';
      ```

   c. 以下のコマンドを実行して、現行の 1 次サーバーを新規 2 次サーバーに接続します。

      ```
      ADMIN COMMAND 'hsb connect';
      ```

3.7.4 スペアでの HAC の使用

HAC は、スペアのシナリオではサポートに制限があります。 HAC はスペア内で使用できますが、1 次サーバー・ノード内の HAC がスペア・サーバーの HSB 状態のモニターを開始するためには、事前に 1 次側の HAC の接続情報、**RemoteDB.Connect** を更新しておく必要があります。そのためには、solidhac.ini ファイルを更新し、1 次サーバー・ノード内の HAC を再開する必要があります。

同様に、スペア・ノード内の HAC は 1 次サーバーの接続情報を必要とします。それが事前に分かっていない場合は、その情報を solidhac.ini ファイルに追加する必要があります。その情報を solidhac.ini ファイルに追加した後、HAC を再開する必要があります。

4 アプリケーションでの HotStandby の使用

HotStandby を使用する場合は、アプリケーション設計によって、アプリケーションが HotStandby サーバーに接続する方法と、障害とフェイルオーバーをアプリケーションが認識して処理する方法に対応する必要があります。

アプリケーションが HotStandby サーバーのペアに接続する方法は、アプリケーションが (アプリケーションが使用しているデータベースと比較して) どこに存在するかに応じて、およびフェイルオーバー状態でのオートメーションの優先レベルに応じて異なります。

通常、アプリケーションはデータベースを使用する場合、1 つのデータベースに接続し、照会、読み取り、および書き込みのすべてにそのデータベースを使用します。そのデータベースが使用できなくなると、接続は切断され、アプリケーションは、同じデータベースに再び接続できるようになるまで待機する必要があります。HotStandby ペアでは、データベースは 2 つになります。この場合、どちらのデータベースも稼働し、相互にミラーリングして、データベースの内容は同一になります。そのため、アプリケーションは、使用できる選択肢が多くなり、データベース・ノードの一方で障害が発生しても、アプリケーションはそのユーザーに提供するサービスをさらに確実に続行できるようになります。

4.1 HotStandby サーバーへの接続

HotStandby 環境でアプリケーションをプログラミングする方法として、基本接続と透過接続 (TC) の 2 つの方法があります。基本接続では、クライアントは HSB サーバーそれぞれに明示的に接続します。透過接続では、クライアントは TC 接続と呼ばれる唯一の論理接続を規定します。

どちらの接続タイプも、solidDB ODBC ドライバーと JDBC ドライバー、および SMA サーバーでサポートされています。接続タイプは接続ストリング内で定義されます。基本接続では、標準の solidDB 接続構文が使用されます。透過接続では、TC 接続固有の構文が使用されます。

基本接続

基本接続では、HotStandby 構成の各サーバーへの接続を、アプリケーションが特定のサーバー・アドレスを使用して別々に処理する必要があります。フェイルオーバーが発生した場合、アクティブ接続は失われ、アプリケーションは新規 1 次サーバーに再接続する必要があります。

透過接続

透過接続を使用すると、アプリケーションは特定のサーバーへの接続を処理する必要がなく、フェイルオーバーの場合でも再接続の処理を行う必要がありません。アプリケーションは、*TC 接続* と呼ばれる論理接続 (ハンドル) を保守します。つまり、透過接続を使用すれば、アプリケーションは数多くのサーバーとそのアドレスについての処理から解放されます。

接続ハンドルは、指定されたサーバー・セット内に PRIMARY ACTIVE、PRIMARY ALONE、STANDALONE のいずれかの状態にあるサーバーが 1 つでもある限り、フェイルオーバーや切り替えが何回起きても維持されます。フェイルオーバーおよび切り替えのとき、ドライバーは*接続切り替え* と呼ばれる内部操作を実行します。アプリケーションは、(障害透過性レベルに応じて) セッション状態を再構成しなければならない場合もあるので、接続切り替えに関する通知を受けます。

ネットワーク・ベースの接続による透過接続

ネットワーク・ベースの接続では、アプリケーションとサーバーの配置場所は同じノード上でも異なるノード上でも構いません。

アプリケーションは、ロード・バランシング機能を使用して、読み取り専用負荷を 2 次サーバーに送信します。

SMA 接続による透過接続

SMA では、2 つのアプリケーション (各 HotStandby ノードに 1 つずつ) があります。1 次ノードのアプリケーションは、読み取りおよび書き込みに SMA 接続を使用します。2 次ノードのアプリケーションは、SMA 接続を使用して読み取りをローカルで実行します。2 次ノードのアプリケーションからの書き込みトランザクションは、ネットワーク接続を使用して 1 次サーバーで実行されます。

サーバーのフェイルオーバーと切り替えの処理方法は、原則としてはネットワーク・ベースの場合と同じです。ただし、SMA サーバーの障害のためにアプリケーションで障害が発生した場合は、アプリケーション固有の高可用性処理が必要です。

4.1.1 接続タイプの選択

手順

以下の互換性マトリックスは、選択された接続情報に対してサポートされている機能が示されているので、ユーザーが接続タイプを選択するのに役立ちます。

表 10. 接続タイプの選択

機能	スタンドアロン構成	HSB 構成
基本接続	可 (BC 情報)	可 (BC 情報)
透過的フェイルオーバー	不可	可 (TC 情報)
ロード・バランシング	不可	可 (TC 情報)

4.2 透過接続

solidDB 透過接続を使用した場合、solidDB ODBC または JDBC ドライバーは 2 つの HSB サーバーの存在を、アプリケーションに対して隠蔽します。透過接続では、1 次サーバーと 2 次サーバー間の透過的なロード・バランシングも提供されます。

このドライバーは、内部のアクティブ接続にマップされる単一の論理 TC 接続を提供します。1 次と 2 次の両方のサーバーがアクティブ状態で稼働している理想的な状況では、ドライバーは*スタンバイ接続* (つまり 2 次サーバーへの接続) も維持します。この接続はイベント待機モードに設定され、HSB の状態変更イベントをいつ

でも受信できる状態になります。それらのイベントは、フェイルオーバーと切り替えに関してドライバーが使用する基本的な情報源です。スタンバイ接続は (例えば、Primary Alone 操作の場合などに) 失われることがありますが、ドライバーは可能な限りそれを確立しようとします。

スタンバイ接続は、アプリケーションに対して完全に透過的に処理されます。また、接続切り替え (つまり、TC 接続から内部アクティブ接続へのマッピングの変更) が発生すると、それが特殊なエラー・コードによってアプリケーションへ通知されます。

重要: solidDB SQL エディター (**solsql**) などの solidDB ツールは、TC 接続をサポートしていません。

4.2.1 透過接続の定義

透過接続は、solidDB 専有の ODBC 透過接続情報設定または非標準 JDBC 接続プロパティーを使用して指定します。

透過接続情報の構文 – ODBC

solidDB 透過接続を使用した場合、クライアントは TC 接続と呼ばれる唯一の論理接続を規定します。ODBC では、TC 接続は、TC 情報の中で指定されます。TC 情報は、両方の HSB 構成で透過的なフェイルオーバーとロード・バランシングを規定します。

ODBC アプリケーションでは、以下の方法で TC 接続を指定することができます。

- SQLConnect 関数:

```
rc = SQLConnect(comHandle, "<solidDB_TC_Info>", ...
```

- クライアント・サイドの solid.ini ファイル:

```
[Com]
Connect = <solidDB_TC_Info>
```

solidDB TC 情報の構文は、以下のとおりです。

```
<solidDB_TC_Info>::= {[<failure_transparency_level_attribute>]
[<preferred_access_attribute>] [<encryption_attribute>]
<connect_target_list>} | <cluster_info>

failure_transparency_level_attribute ::= TF_LEVEL={NONE |
CONNECTION | SESSION}

preferred_access_attribute::= PREFERRED_ACCESS={WRITE_MOSTLY |
READ_MOSTLY | LOCAL_READ}

encryption_attribute::=USE_ENCRYPTION)={YES|NO}

connect_target_list::= [SERVERS=]<connect_string>[, <connect_string > ...]

cluster_info::= CLUSTER <connect_string>[, <connect_string>...]
```

以下の省略形を使用できます。

表 11. TC 情報の省略形

省略形	対応する構文
TF	TF_LEVEL

表 11. TC 情報の省略形 (続き)

省略形	対応する構文
CO	CONNECTION
SES	SESSION
PA	PREFERRED_ACCESS
RM	READ_MOSTLY
WM	WRITE_MOSTLY
LR	LOCAL_READ
S	SERVERS

障害透過性属性

障害透過性は、障害のマスキングを処理します。障害透過性レベルは、TC 情報の TF_LEVEL 属性で設定します。次の 3 つのレベルがあります。

1. NONE - 障害透過性は無効です。これはデフォルト値です。
2. CONNECTION - サーバー接続が保存されます。つまり、フェイルオーバーや切り替えの場合に再接続する必要はありません。
3. SESSION - デフォルト以外の値が設定された特定のセッション属性が保存されます。さらに、準備済みステートメントも保存されます。ただし、オープン・カーソルはクローズされ、進行中のトランザクションは打ち切られます。

優先アクセス属性 - ロード・バランシング

優先アクセス属性 (PREFERRED_ACCESS) は、読み取り専用負荷のロード・バランシングを適用するかどうかを示します。次のレベルがあります。

- WRITE_MOSTLY - ロード・バランシングなし (デフォルト)。すべてのトランザクションは、1 次サーバーで実行されます。
- READ_MOSTLY - **Cluster.ReadMostlyLoadPercentAtPrimary** による定義に従って、1 次サーバーと 2 次サーバー間で読み取り専用トランザクションを分散することによるロード・バランシング。
- LOCAL_READ - 可能な場合は読み取り専用トランザクションをローカルで実行することによるロード・バランシング。読み取り専用トランザクションは、1 次サーバーか 2 次サーバーかに関係なく、常にローカル・サーバーに送信されます。ローカル・サーバーが見つからないと、1 次サーバーの割り当てられたワークロード接続 (接続ストリングで定義された最初のネットワーク・ベースの接続) が使用されます。書き込みトランザクションは、常に 1 次サーバーで実行されます。

  LOCAL_READ は、各ノードに少なくとも 1 つの SMA アプリケーションがある SMA のセットアップで一般的に使用されます。

接続ターゲット・リストとクラスター情報 - サーバー・アドレス

solidDB TC 情報には、*接続ターゲット・リスト* と呼ばれるサーバー・アドレスのリストが含まれています。HotStandby 構成に SMA が含まれている場合は、SMA 固有の接続ターゲット・リストを使用する必要があります。

ネットワーク・ベースのアプリケーションの接続ターゲット・リストの構文は、以下のとおりです。

```
[SERVERS:] <network_connect_string>, <network_connect_string>
```

SMA アプリケーションの接続ターゲット・リストの構文は、以下のとおりです。

```
[SERVERS:] <sma_connect_string>, <network_connect_string>
```

network_connect_string

ネットワーク・ベースの接続の接続ストリングの形式は、以下のとおりです。

protocol_name [*options*] [*host_computer_name*] *server_name*

ここで、

- *options* には、以下のオプションを任意に組み合わせて指定できます。

表 12. 接続ストリングのオプション

オプション	説明	プロトコル
-4	クライアントの接続が IPv4 プロトコルのみを使用することを指定します。	TCP/IP
-6	クライアントの接続が IPv6 プロトコルのみを使用することを指定します。 IPv6 プロトコルが使用される場合、Windows 環境では、このオプションは必須です。	TCP/IP
-i*source_address*	システムのデフォルト・ソース IP アドレス・バインディングがアプリケーションのニーズに合わない場合のために、明示的に接続するソケット・ソース・アドレスを指定します。 *source_address* には IP アドレスまたはホスト名を使用できます。	TCP/IP
-z	この接続でのデータ圧縮を可能にします。	すべて
-c *milliseconds*	ログイン・タイムアウトを指定します (デフォルトは、オペレーティング・システムに固有です)。指定された時間が経過すると、ログイン要求が失敗します。	TCP/IP
-r *milliseconds*	接続 (または読み取り) のタイムアウトを指定します。指定された時間内に応答が受信されないと、ネットワーク要求が失敗します。値 0 (デフォルト) を指定すると、タイムアウトの制限がなくなります (オペレーティング・システムのデフォルト・タイムアウトが適用されます)。	TCP/IP
-o*filename*	ネットワーク・トレース機能をオンにして、トレース出力ファイルの名前を定義します。 詳しくは、「*IBM solidDB 管理者ガイド*」の『*ネットワーク・トレース機能*』を参照してください。	すべて
-p*level*	指定されたレベル (0 - 5) でサーバーに ping します。 クライアントは、レベル 1 の solidDB Ping 機能をいつでも使用できます (0 はノーオペレーション/デフォルト)。レベル 2、3、4、または 5 は、サーバーで少なくとも同じレベルの Ping 機能を使用するように設定されている場合に限り使用できます。 詳しくは、「*IBM solidDB 管理者ガイド*」の『*Ping 機能*』を参照してください。	すべて
-t	ネットワーク・トレース機能をオンにします。 詳しくは、「*IBM solidDB 管理者ガイド*」の『*ネットワーク・トレース機能*』を参照してください。	すべて

- *host_computer_name* は、クライアントとサーバーを別のマシンで実行している場合に、TCP/IP プロトコルおよび名前付きパイプ・プロトコルで必要となります。
- *server_name* は、通信プロトコルによって異なります。

– TCP/IP プロトコルでの server_name は、2315」などのサービス・ポート番号です。
– その他のプロトコルでの server_name は、「soliddb」や「chicago_office」などの名前です。

各種通信プロトコルでの構文について詳しくは、「*IBM solidDB 管理者ガイド*」の『*通信プロトコル*』を参照してください。

注:

- *protocol_name* と server_name は、サーバーがネットワーク listen 名で使用しているものと一致している必要があります。
- 接続ストリングを接続時に指定する場合は、引用符で囲む必要があります。
- 接続ストリングのすべてのコンポーネントでは、大/小文字を区別しません。

sma_connect_string

SMA ベースの接続の接続ストリングの形式は、以下のとおりです。

```
sma protocol_name port_number | pipe_name
```

SMA を使用する場合、各ノードのアプリケーションは、SMA 接続でローカル・サーバーに、ネットワーク・ベースの接続でリモート・サーバーに接続できる必要があります。つまり、サーバー・アドレスのリストは、以下の形式になります。

```
connect_target_list::= [SERVERS=]<sma_connect_string>, <network_connect_string>

cluster_info::= CLUSTER <sma_connect_string>, <network_connect_string>
```

以下に例を示します。

```
sma tcp 2315, tcp 192.168.255.1 1315
```

ドライバーはリストを左から右へスキャンして、1 次サーバーと 2 次サーバーを検出しようとします。したがって、優先したい構成はリストの先頭に配置する必要があります。リストの残りの部分には、システムの存続期間中、いずれかの時点でアクティブにされる可能性があるスペア・アドレスを含めることができます。リストは常に短くしておいてください。エラー状況で、エラーがアプリケーションに返されるまでに長い時間を要する可能性があるからです。アドレスは 1 つずつ試行され、指定されたログイン・タイムアウトが適用されます (ネットワーク・ベースの接続)。リスト内のアドレスの数は無制限です。

属性 TF_LEVEL と PREFERRED_ACCESS のどれも指定されなかった場合 (または TF_LEVEL=NONE)、接続動作は基本接続にフォールバックします。複数の接続ストリングを指定した場合は、接続要求を受け入れるリスト上の最初のサーバーに対して接続が確立されます。

マルチホーム・サーバーでのサーバー・アドレスの構成

セットアップで、マルチホーム・サーバーを使用して、アプリケーションとサーバー間の接続に対するさまざまなネットワークと、サーバー自体をデプロイする場合には、**HotStandby.TCConnect** パラメーターで TC 接続のサーバー・アドレスを定義する必要があります。

アプリケーション接続の観点から見ると、**HotStandby.TCConnect** パラメーターで指定されたアドレスは、**HotStandby.Connect** パラメーターで定義されたアドレスに優先します。したがって、TC 接続は **HotStandby.TCConnect** パラメーターで指定されたサーバー・アドレスを使用するのに対し、サーバー間の HotStandby 接続は **HotStandby.Connect** パラメーターで定義されたサーバー・アドレスを使用します。

マルチホーム・サーバー構成の例については、 93 ページの『例: マルチホーム・サーバーを使用した TC 接続』を参照してください。

CLUSTER

CLUSTER キーワードを指定すると、TF_LEVEL は SESSION に、PREFERRED_ACCESS は READ_MOSTLY に自動的に設定されます。

例えば、以下の TC 情報と CLUSTER ストリングは同義です。

```
TF_LEVEL=SESSION PREFERRED_ACCESS=READ_MOSTLY
SERVERS=tcp srv1.acme.com 1315, tcp srv2.acme.com 1315
```

```
CLUSTER tcp srv1.acme.com 1315, tcp srv2.acme.com 1315
```

ヒント: クラスター構成はクライアント・サイドの solid.ini ファイルでも定義できます。この場合、SQLConnect の接続ストリングは論理名を使用できます。以下に例を示します。

```
rc = SQLConnect(comHandle, "Cluster1", ...
```

```
[Data Sources]
Cluster1=
   TF_LEVEL=SESSION
   PREFERRED_ACCESS=READ_MOSTLY
   SERVERS=
      tcp -c 1000 srv1.dom.acme.com 1315,
      tcp srv2.dom.acme.com 1315,
      tcp srv3.dom.acme.com 1316
```

SQLConnect の例

```
rc = SQLConnect(comHandle, "TF=CONNECTION
      USE_ECRYPTION=YES PA=READ_MOSTLY
      SERVERS=
      tcp -c 1000 srv1.dom.acme.com 1315,
      tcp srv2.dom.acme.com 1315,
      tcp srv3.dom.acme.com 1316", ...
```

```
rc = SQLConnect(comHandle, "CLUSTER=
      tcp -c 1000 srv1.dom.acme.com 1315,
      tcp srv2.dom.acme.com 1315,
      tcp srv3.dom.acme.com 1316", ...
```

SMA のセットアップ: ノード 1 (srv1.dom.acme.com) のアプリケーションの SQLConnect

```
rc = SQLConnect(comHandle, "TF=CONNECTION
      PA=LOCAL_READ
      SERVERS=
      sma tcp 1315,
      tcp srv2.dom.acme.com 2315", ...
```

SMA のセットアップ: ノード 2 (srv2.dom.acme.com) のアプリケーションの SQLConnect

```
rc = SQLConnect(comHandle, "TF=CONNECTION
      PA=LOCAL_READ
      SERVERS=
      sma tcp 2315,
      tcp srv1.dom.acme.com 1315", ...
```

クライアント・サイドの solid.ini の例

注: レイアウトの理由から、以下の例の Com.Connect パラメーター値は複数の行に分かれています。実際の solid.ini ファイルでは、パラメーター入力全体を 1 行に収める必要があります。

```
[Com]
Connect = TF=CONNECTION USE_ECRYPTION=YES
      PA=READ_MOSTLY
      SERVERS=
         tcp -c 1000 srv1.dom.acme.com 1315,
         tcp srv2.dom.acme.com 1315,
         tcp srv3.dom.acme.com 1316
```

SMA のセットアップ: ノード 1 (srv1.dom.acme.com) の solid.ini ファイル

```
[Com]
Connect = TF=CONNECTION USE_ECRYPTION=YES
      PA=LOCAL_READ
      SERVERS=
         sma tcp 1315,
         tcp srv2.dom.acme.com 2315
```

SMA のセットアップ: ノード 2 (srv2.dom.acme.com) の solid.ini ファイル

```
[Com]
Connect = TF=CONNECTION USE_ECRYPTION=YES
      PA=LOCAL_READ
      SERVERS=
         sma tcp 2315,
         tcp srv1.dom.acme.com 1315
```

JDBC による透過接続

JDBC では、透過接続は標準外の接続プロパティーによって使用可能にされます。サーバー・アドレスのリストは、JDBC 接続ストリングの一部として与えられます。

注: JDBC で透過接続を使用する場合は、ステートメント・オブジェクトの廃棄も明示的に処理する必要があります。ガーベッジ・コレクターは、参照されなかったステートメント・オブジェクトを検出しません。

障害透過性レベル (solid_tf_level)

障害透過性は、solid_tf_level 接続プロパティーで使用可能にされます。この値はストリングです。ニーモニック (例えば、NONE) または数値 (NONE の場合は 0) として指定できます。明確にするため、ニーモニックを使用することをお勧めします。

次の 3 つのレベルがあります。

1. NONE | 0 - 障害透過性は無効です。これはデフォルト値です。
2. CONNECTION | 1 - サーバー接続が保存されます。つまり、フェイルオーバーや切り替えの場合に再接続する必要はありません。

3. SESSION | 3 - デフォルト以外の値が設定された特定のセッション属性が保存されます。さらに、準備済みステートメントも保存されます。ただし、オープン・カーソルはクローズされ、進行中のトランザクションは打ち切られます。

優先アクセス属性 (solid_preferred_access)

優先アクセス属性は、読み取り専用負荷を分散するかどうかを示します。優先アクセス属性は、solid_preferred_access 接続プロパティーで使用可能にされます。この値はストリングです。ニーモニックまたは数値として指定できます。明確にするため、ニーモニックを使用することをお勧めします。

次の 3 つのレベルがあります。

- WRITE_MOSTLY | 0 では、ワークロードが 1 次サーバーへ誘導されます。これはデフォルト値です。WRITE_MOSTLY は、接続を WRITE MOSTLY モードにも設定します。数値を指定してこれを行うことはできません。
- READ_MOSTLY | 1 - デフォルトでは、ワークロードは 2 次サーバーに送信されます。書き込みトランザクションは 1 次サーバーへ引き渡されます。
- LOCAL_READ | 2 – ワークロードがローカル・サーバーへ誘導されます。ローカル・サーバーが見つからないと、1 次サーバーの割り当てられたワークロード接続 (接続ストリングで定義された最初のネットワーク・ベースの接続) が使用されます。

再接続タイムアウト (solid_tf1_reconnect_timeout)

solid_tf1_reconnect_timeout プロパティーでは、接続の再接続タイムアウトをミリ秒単位で指定します。デフォルト値は 10,000 ミリ秒 (10 秒) です。

サーバー・アドレス

サーバー・アドレスのリストは、拡張 JDBC 接続ストリングの一部として与えられます。

```
conStr= "jdbc:solid://host_name:port [,host_name:port].../user_name/password";
```

アドレス・リスト内のアドレスの数は、20 までに制限されています。

マルチホーム・サーバーでのサーバー・アドレスの構成

セットアップで、マルチホーム・サーバーを使用して、アプリケーションとサーバー間の接続に対するさまざまなネットワークと、サーバー自体をデプロイする場合には、**HotStandby.TCConnect** パラメーターで TC 接続のサーバー・アドレスを定義する必要があります。

アプリケーション接続の観点から見ると、**HotStandby.TCConnect** パラメーターで指定されたアドレスは、**HotStandby.Connect** パラメーターで定義されたアドレスに優先します。したがって、TC 接続は **HotStandby.TCConnect** パラメーターで指定されたサーバー・アドレスを使用するのに対し、サーバー間の HotStandby 接続は **HotStandby.Connect** パラメーターで定義されたサーバー・アドレスを使用します。

SMA 接続 (solid_shared_memory)

SMA 構成では、標準外のプロパティー solid_shared_memory を yes に設定する必要があります。

例

```
...
String conStr = "jdbc:solid://srv1.acme.com:1323,srv2-acme.com:1423/dba/dba";
Properties prop = new Properties();
prop.setProperty("solid_tf_level", "CONNECTION");
...
Connection c = DriverManager.getConnection(conStr, prop);
...
```

```
...
String conStr = "jdbc:solid://localhost:1323,srv2-acme.com:1423/dba/
dba?solid_shared_memory=yes";
Properties prop = new Properties();
prop.setProperty("solid_tf_level", "CONNECTION", "LOCAL_READ");
...
Connection c = DriverManager.getConnection(conStr, prop);
...
```

TC の属性の組み合わせ

以下の表は、TC の属性の可能な組み合わせを要約したもので、結果としての接続機能も示しています。

表 13. TC 情報の属性の可能な組み合わせ

PREFERRED_ ACCESS:	TF_LEVEL: 指定されないか NONE	TF_LEVEL: CONNECTION	TF_LEVEL: SESSION
指定されない	• フェイルオーバーまたは切り替えはサポートされない • ロード・バランシングなし (基本接続)	• 透過的フェイルオーバー (セッション状態は保存されない) • 透過的な切り替え • 1 次サーバーのみでのワークロード • ロード・バランシングなし	• 透過的フェイルオーバー (セッション状態は保存される) • 透過的な切り替え • 1 次サーバーのみでのワークロード • ロード・バランシングなし
WRITE_MOSTLY (デフォルト)	• 透過的フェイルオーバーのサポートなし • 透過的な切り替え • 1 次サーバーのみでのワークロード • ロード・バランシングなし	• 透過的フェイルオーバー (セッション状態は保存されない) • 透過的な切り替え • 1 次サーバーのみでのワークロード • ロード・バランシングなし	• 透過的フェイルオーバー (セッション状態は保存される) • 透過的な切り替え • 1 次サーバーのみでのワークロード • ロード・バランシングなし

表 13. TC 情報の属性の可能な組み合わせ (続き)

PREFERRED_ ACCESS:	TF_LEVEL: 指定されないか NONE	TF_LEVEL: CONNECTION	TF_LEVEL: SESSION
READ_MOSTLY	• 透過的フェイルオーバーのサポートなし • 透過的な切り替え • 2 次サーバーと 1 次サーバーでのワークロード • ロード・バランシング	• 透過的フェイルオーバー (セッション状態は保存されない) • 透過的な切り替え • 2 次サーバーと 1 次サーバーでのワークロード • ロード・バランシング	• 透過的フェイルオーバー (セッション状態は保存される) • 透過的な切り替え • 2 次サーバーと 1 次サーバーでのワークロード • ロード・バランシング

接続エラー処理

TC 接続の接続要求が発行された場合、少なくとも 1 つの該当するサーバーが検出されて、それに接続した場合、接続要求は成功したと見なされます。

サーバーは、PRIMARY ACTIVE、PRIMARY ALONE、STANDALONE のいずれかの状態になります。そうでない場合、接続の試みは失敗したと考えられます。アドレス・リストは 1 回だけスキャンされます。

接続要求の失敗には、さまざまな理由が存在する場合があります。その大部分は、以下のエラー・ケースによってマスクされます。

表 14. 接続要求エラー

SQLSTATE	ネイティブ・コード	メッセージ・テキストおよび説明
08001	25217	`Client unable to establish a connection` 説明: ドライバーは、該当するサーバーを見つけて接続するために、TC 接続情報を使用しました。その試みは、以下のいずれかの理由で失敗しました。 • アドレス・リストにリストされたホストが見つからなかった • ホストは見つかったが、ログインがタイムアウトになった • ホストは見つかったが、ログインが拒否された • ホストは見つかったが、PRIMARY/STANDALONE 状態ではなかった

表 14. 接続要求エラー (続き)

SQLSTATE	ネイティブ・コード	メッセージ・テキストおよび説明
HY000	21307	`Invalid connect info...` 説明: 基本接続ストリングまたは TC 接続情報 (データ・ソース情報) の中で構文エラーが検出されました。
HY000	21300	`Protocol ... not supported.` 説明: TC 接続情報の先頭にあるストリング「TC」のつづりに誤りがあります (または、基本接続ストリング内で誤ったプロトコル名が指定されています)。

警告とともに接続が受け入れられる場合もあります。

表 15. 警告

SQLSTATE	ネイティブ・コード	メッセージ・テキストおよび説明
0100	25218	`Connected to Standalone or Primary Alone server.` TF_LEVEL または PREFERRED_ACCESS にデフォルト以外の値を設定使用としましたが、使用可能なサーバーは 1 台だけです。この場合、障害透過性もロード・バランシングも使用できません。

例: TC 接続

この例では、アプリケーションとサーバーが同じネットワーク内にある典型的な HotStandby セットアップの透過接続構成を示します。

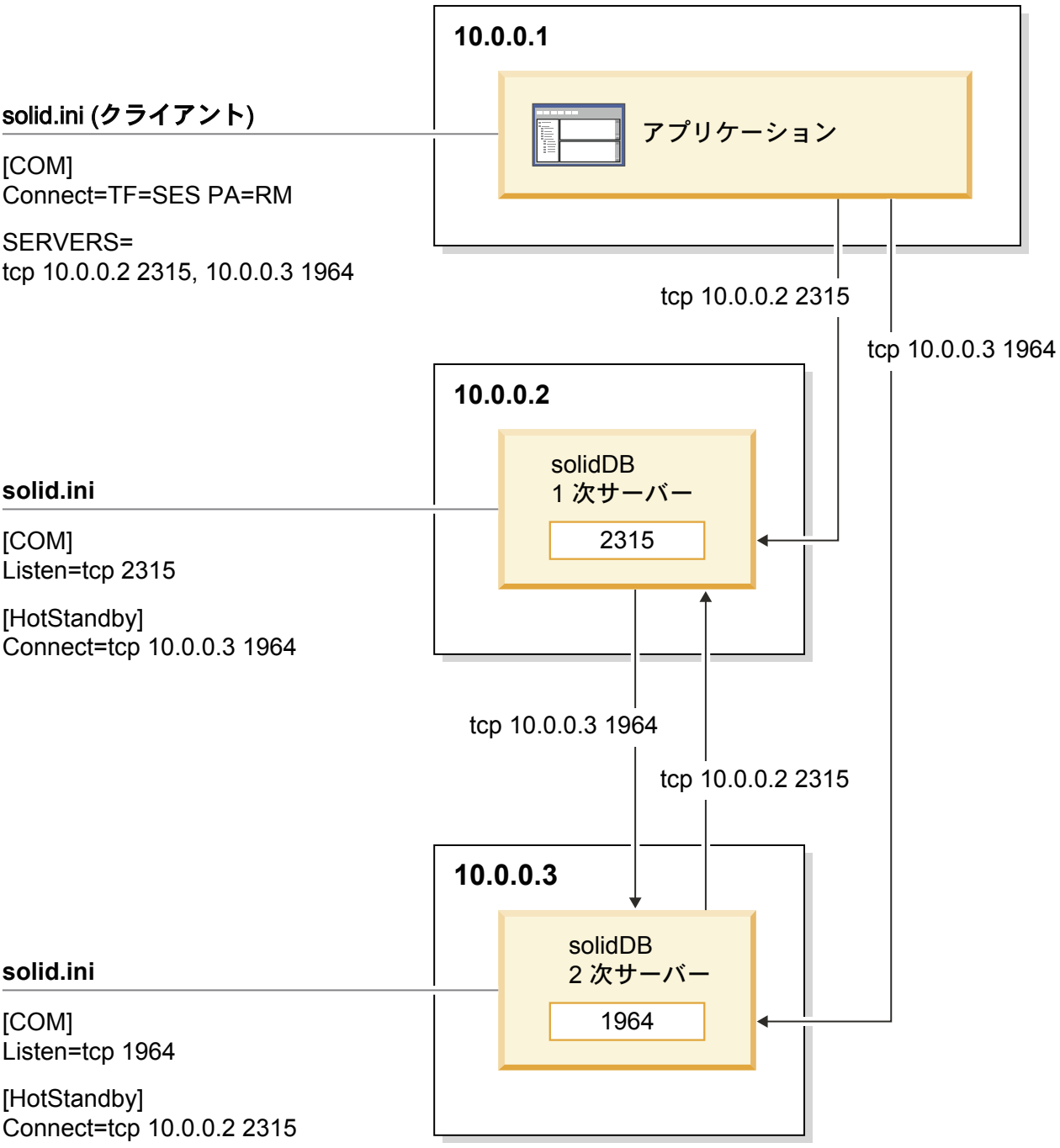


図 14. 例: TC 接続

例: マルチホーム・サーバーを使用した TC 接続

この例では、1 次サーバーおよび 2 次サーバーがマルチホーム・サーバーである場合の HotStandby セットアップの透過接続構成を示します。これは、アプリケーションとサーバーが別々のネットワークにあるセットアップや、サーバー間の HotStandby 接続に使用されているネットワークを使用してアプリケーションがサーバーに接続できないか、接続すべきでないセットアップでは標準的です。

TC 接続と HotStandby 接続を区別するために、サーバー・アドレスを定義する以下の 2 つのパラメーターを使用する必要があります。

- **HotStandby.Connect** パラメーターは、1 次サーバーと 2 次サーバーの間の HotStandby 接続で使用されるサーバー・アドレスを指定します。
- **HotStandby.TCConnect** パラメーターは、TC 接続で使用されるサーバー・アドレスを指定します。

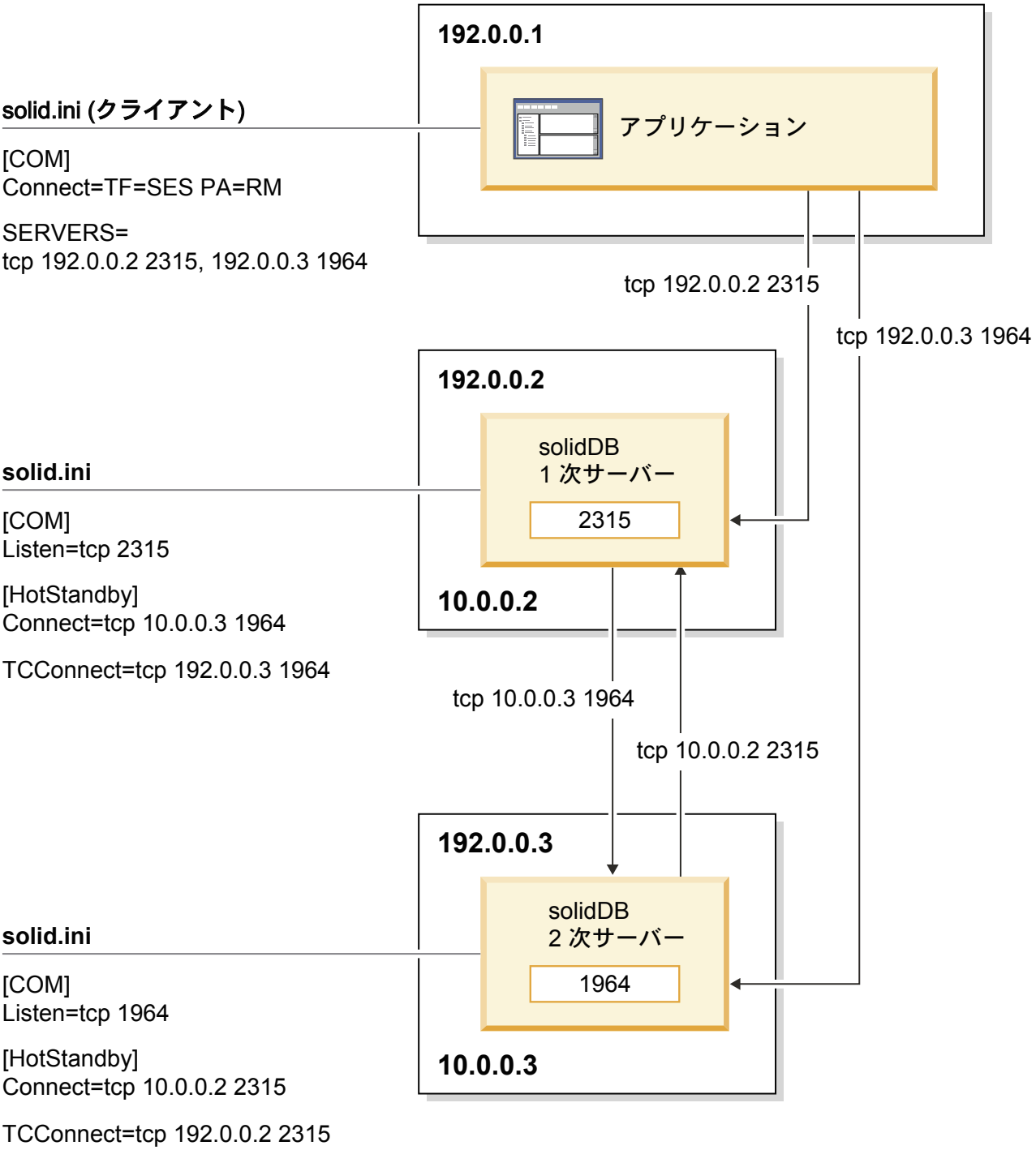


図 15. 例: マルチホーム・サーバーを使用した TC 接続

4.2.2 透過接続での障害透過性

障害透過性は、障害のマスキングを処理します。障害透過性レベルは、TF_LEVEL 属性で設定されます。

次の 3 つの障害透過性レベルがあります。

TF_LEVEL=NONE
障害透過性は無効です (デフォルト)。

TF_LEVEL=CONNECTION
サーバー接続が保存されます。つまり、フェイルオーバーや切り替えの場合に再接続する必要はありません。

TF_LEVEL=SESSION
デフォルト以外のほとんどのセッション・レベルの設定が保存されます。さらに、準備済みステートメントも保存されます。ただし、オープン・カーソルはクローズされ、進行中のトランザクションは打ち切られます。

接続切り替え処理の原則

接続切り替えとは、ドライバーがアクティブ・サーバー接続を変更する状態を指しています。一般に、接続切り替えの理由は 2 次サーバーへのフェイルオーバーか、サーバー間の切り替えです。

具体的には、接続切り替えの必要性は、以下のいずれかのイベントから検出されます。

- PRIMARY ALONE (フェイルオーバー) または PRIMARY ACTIVE (切り替え) への状態変更に関する 2 次サーバーからのイベント。これは、接続切り替えを行う主な (また最高速の) モードです。
- 1 次サーバーでの状態変更の標識。
- アクティブ接続でのリンク障害。
- アクティブ接続での接続タイムアウト。

ドライバーは、以下の 2 つのステップで接続切り替えを実行します。

1. 接続切り替えの必要性が検出されます。ドライバーは、処理中の要求または以下の要求について、以下の接続切り替えエラーを返します。

表 16. 接続切り替え要求

SQLSTATE	ネイティブ・コード	メッセージ・テキストおよび説明
HY000	25216	`Connection switch, some session context may be lost` 説明: ドライバーは接続切り替えの必要性を検出しました。クライアントは、接続切り替えをファイナライズするために、トランザクション・ロールバック呼び出しを発行することを期待されます。このエラー・コードおよびメッセージは、ロールバック呼び出しが発行されるまで、連続するネットワーク要求呼び出しごとに受信されます。

2. クライアント・プログラムはロールバック・コマンドを発行します (ODBC: SQL_ROLLBACK 付きの SQLEndTran()。JDBC: Connection.rollback())。ロールバックが成功すると、新しいアクティブ接続が、使用される可能性がある TC 接続へマップされています。

   注: 少数の連続する ODBC 呼び出しについて、接続切り替えエラーが返される場合があります。したがって、すべての ODBC ネットワーク要求で、このエラーに対して常にロールバックで応答するよう、対策を講じておく必要があります。トランザクションの最中にこれが発生した場合は、トランザクションを再実行する必要があります。

   一方、新しいアクティブ接続を確立できない場合は、以下のエラー・コードが返されます。

表 17. 通信リンク障害

SQLSTATE	ネイティブ・コード	メッセージ・テキストおよび説明
08S01	14503	`Communication link failure` 説明: ドライバーは、新しいアクティブ接続の確立に失敗しました。TF 接続は失われ、クライアントは作業を続行するために、(データ・ソース情報を使用して) 再接続する必要があります。

ロールバック呼び出しを受信後、ドライバーは新しいアクティブ接続を検出する少数の代替手段を使用します。最も単純なケースでは、その目的にスタンバイ接続が使用されます。その接続が正しい状態にない場合、ドライバーは適正なイベントが到着するまで 2 秒間待機します。イベントが到着しない場合、およびその他のケースでは、ドライバーは TC 接続情報内のアドレス・リストにフォールバックし、接続シーケンスを最大 10 秒間繰り返します。すべての試みが失敗した場合、ドライバーは上記のエラーに戻ります。

エラーによる影響は、アプリケーションが検出したように、接続が失われることです。それ以後にその接続上で発行される要求は、同じエラーになります。

セッション状態の保存

ドライバーによって接続切り替えが実行された場合、一部のセッション・コンテキストが失われる可能性があり、アプリケーションはそれを再構成する必要があります。保存される状態の量は、透過的フェイルオーバーのレベルによって指示され、TC 情報の属性 TF_LEVEL で表されます。

TF レベルの CONNECTION では、状態は保存されません。SESSION レベルでは、ほとんどのセッション状態が保存されます。セッション状態の保存は、必要なデータをドライバー内にキャッシングすることによって実施されます。透過性レベルが高いほど、キャッシングを必要とする要求の応答時間は遅くなり、ドライバーでのメモリー使用量は増加します。

フェイルオーバーの場合は、TF レベルに関係なく、以下が適用されます。

- 現行トランザクションの更新情報は (トランザクション・ロールバックのために) 失われます。
- オープン・カーソルおよびその位置は失われます。

表 18. セッション状態の保存

TF_LEVEL	**保存される状態**
CONNECTION	セッション状態は保存されません。
SESSION	準備済みステートメント • 準備済みステートメントは保存されます。 以下のステートメントの効果は保存されます。 • SET CATALOG • SET SQL INFO • SET SQL SORTARRAYSIZE • SET SQL CONVERTORSTOUNIONS • SET SQL JOINPATHSPAN • SET LOCK TIMEOUT <*seconds*> • SET IDLE TIMEOUT • SET OPTIMISTIC LOCK TIMEOUT • SET STATEMENT MAXTIME • SET ISOLATION LEVEL • SET DURABILITY • SET SAFENESS • SET SCHEMA • SET SYNC USER • SET SYNC MODE 以下の標準 ODBC 属性は保存されます。 • SQL_ATTR_ACCESS_MODE (SET READ ONLY、 SET READ WRITE) • SQL_ATTR_CURRENT_CATALOG (上記の SET CATALOG を複製する) • SQL_ATTR_AUTOCOMMIT

4.2.3 透過接続でのロード・バランシング

TC では、ドライバーは 2 つの方式を使用してトランザクションの負荷を方向付けます。1 つは読み取り集中負荷を処理する方式で、もう 1 つは書き込み集中負荷を処理する方式です。ロード・バランシングの場合は、論理 TC 接続は「*ワークロード接続*」と呼ばれる低い方のレベルのサーバー接続にマップされます。ワークロー

ド接続は、時間が経つと変化する場合があり、通常はアプリケーションには関係ありません。しかし、必要な場合は、何が現行のワークロード接続であるかを知るための方法があります。

静的ロード・バランシング構成

ロード・バランシング方式は、以下のとおりです。

PREFERRED_ACCESS=WRITE_MOSTLY - ロード・バランシングなし (デフォルト)。 WRITE_MOSTLY では、すべてのトランザクションは 1 次サーバーで実行されます。これは、一般的な HotStandby 操作に対応しています。WRITE_MOSTLY メソッドは、書き込み集中の負荷で役立ちます。

PREFERRED_ACCESS=READ_MOSTLY - 1 次サーバーと 2 次サーバーの間で読み取り専用トランザクションを分散することによるロード・バランシング。

READ_MOSTLY では、読み取り専用トランザクションは、2 次サーバーと 1 次サーバーの両方で実行できます。書き込みトランザクションは、常に 1 次サーバーで実行されます。

読み取り専用トランザクションでは、**Cluster.ReadMostlyLoadPercentAtPrimary** パラメーターの設定を基に*割り当てられたワークロード・サーバー* が選択されます。このパラメーターでは、1 次サーバーに送信される読み取り主体の負荷全体のパーセンテージを指定します。

Cluster.ReadMostlyLoadPercentAtPrimary パラメーターのデフォルト値は `50` です。つまり、デフォルトでは、接続の半分が 1 次サーバーを使用し、もう半分が 2 次サーバーを使用します。これは、ほとんどの混合負荷に望ましい値です。この値をゼロに設定した場合、すべての負荷が 2 次サーバーへ誘導されます。これは、極度の読み取り集中 (または読み取り専用) アプリケーションに PREFERRED_ACCESS=READ_MOSTLY を使用しており、(それと同時に) 書き込み集中アプリケーションに PREFERRED_ACCESS=WRITE_MOSTLY を使用している場合に適しています。

READ_MOSTLY の場合、1 次サーバーはドライバーに、ワークロード接続用の接続先サーバーを指示します。所定の接続の負荷が 2 次サーバーに送信される場合に、書き込み操作が実行されると、1 次サーバーへの引き渡しが発生し、トランザクションは 1 次サーバーで実行されます。トランザクションのコミット後、負荷は再び 2 次サーバーへ誘導されます。2 次サーバーに障害が起きると、接続は 2 次サーバーから 1 次サーバーへのフェイルオーバーが行われます。

PREFERRED_ACCESS=LOCAL_READ - 可能な場合はトランザクションをローカルで実行することによるロード・バランシング。

LOCAL_READ では、読み取り専用トランザクションは、1 次サーバーか 2 次サーバーかに関係なく、常にローカル・サーバーに送信されます。書き込みトランザクションは、常に 1 次サーバーで実行されます。

一般的に、LOCAL_READ メソッドは、SMA のセットアップで使用されます。1 次サーバーのアプリケーションは、読み取りトランザクションおよび書き込みトランザクションに SMA 接続を使用します。2 次サーバーのアプリケーションは、読み取りに SMA 接続を使用し、1 次サーバーでの書き込みにネットワーク・ベースの接続を使用します。

分離レベルとロード・バランシング

ロード・バランシングは、READ COMMITTED の分離レベルでのみ機能します。サーバーの分離レベルの (始動) デフォルトが別の値に設定されている場合、PREFERRED_ACCESS=READ_MOSTLY と PREFERRED_ACCESS=LOCAL_READ を設定すると、そのセッションの分離レベルが強制的に READ COMMITTED にされます。分離レベルは、より高いレベル (REPEATABLE READ など) に動的に再設定できますが、その場合、ロード・バランシングは使用不可になります。

また、ロード・バランシングは、セッションが自動コミット・モードに設定された場合にも使用不可になります。

ロード・バランシングの動的な制御

ロード・バランシング (READ_MOSTLY または LOCAL_READ) を使用すると、割り当てられたワークロード・サーバーをプログラムで 2 次サーバーから 1 次サーバーに変更できます。

セッション・レベルでは、以下のステートメントでワークロード接続サーバーを 1 次サーバーに変更できます。

- SET WRITE
- SET ISOLATION LEVEL REPEATABLE READ
- SET ISOLATION LEVEL SERIALIZABLE

ステートメントは、トランザクションの最初のステートメントの場合、直ちに有効になり、そうでない場合は次のトランザクションから有効になります。

トランザクション・レベルでは、以下のステートメントでワークロード接続サーバーを 1 つのトランザクションの間、1 次サーバーに変更できます。

- SET TRANSACTION WRITE
- SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
- SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

影響を受けるトランザクションは、このステートメントを使用して開始したトランザクションか、それ以外の場合は次のトランザクションです。トランザクションが 1 次サーバーで実行された後、ワークロード接続サーバーはセッション用に割り当てられたサーバーに戻ります。

SET [TRANSACTION] WRITE ステートメントの効果は、ステートメント SET [TRANSACTION] READ WRITE で元に戻すことができます。また、以下の分離レベル・ステートメントでも同じ効果があります。

- SET ISOLATION LEVEL READ COMMITTED
- SET TRANSACTION ISOLATION LEVEL READ COMMITTED

ロード・バランシングでのフェイルオーバー透過性

障害透過性 (**TF_LEVEL** が NONE 以外) が設定されていると同時に、ロード・バランシングが使用可能 (**PREFERRED_ACCESS**=READ_MOSTLY または **PREFERRED_ACCESS**=LOCAL_READ) に設定されている場合、適用されるフェイルオーバー・ポリシーは以下のようになります。

1. 1 次サーバー障害: すべての負荷は PRIMARY ALONE 状態にある新規 1 次サーバーへ誘導されます。
2. 2 次サーバー障害: すべての負荷は 1 次サーバー (PRIMARY ALONE) へ誘導されます。
3. 2 つのサーバーが PRIMARY ALONE 状態と SECONDARY ALONE 状態の場合のサーバー間の接続切断: 2 次サーバー内で読み取り専用トランザクションの実行が進行中である場合、そのトランザクションも 2 次サーバー内で正常にコミットされます。それ以後のすべてのトランザクションは、(PRIMARY ALONE の) 1 次サーバーへ誘導されます。

通常のホット・スタンバイ操作が (PRIMARY ACTIVE 状態と SECONDARY ACTIVE 状態にあるサーバーを使用して) 再開されると、1 次サーバーと 2 次サーバーの間で再び負荷のバランスがとられます。

注: 障害透過性が使用不可 (TF_LEVEL=NONE) の場合でも、基本的なフェイルオーバー機能は使用可能です。つまり、2 次サーバーで障害が発生すると、2 次サーバーから 1 次サーバーへのフェイルオーバーが行われます。それ以外の障害は、結果として通信リンクの障害になります。したがって、一般的に TF_LEVEL=NONE の状態で障害が発生した場合、アプリケーションは同じ TC 情報に再接続する必要があります。再接続を避けるには、ロード・バランシングが使用されている状態で障害透過性を使用可能に設定してください。

ロード・バランシング下でのストアード・プロシージャーの実行

SQL ストアード・プロシージャーはすべて、プロシージャー宣言の SQL 標準節 *SQL_data_access_indication* で読み取り専用プロシージャーとして指定されていない限り、1 次サーバーで実行されます。

```
<SQL_data_access_indication> ::=
     NO SQL |
     READS SQL DATA |
     CONTAINS SQL |
     MODIFIES SQL DATA
```

トランザクションの引き渡しを課すキーワードは MODIFIES SQL DATA のみです。これがデフォルトの動作です。

読み取り専用プロシージャーおよび機能の不必要な引き渡しを回避するために、以下のいずれかの値を使用します。

- NO SQL
- READS SQL DATA
- CONTAINS SQL

4.2.4 矛盾する TC 情報の処理

TC 情報の属性が、使用可能にされた実際のサービスと矛盾する場合があります。そのような状況では、接続は認可されますが、SUCCESS_WITH_INFO の警告が発行されます。

これは、以下の場合に行われます。

- PREFERRED_ACCESS が指定されたが、HSB が使用可能でない。基本接続は使用可能である。

- TF_LEVEL が指定されたが、HSB が使用可能でない。基本接続は使用可能である。

4.3 基本接続

基本接続では、HotStandby 構成またはクラスター構成の各サーバーへの接続を、アプリケーションが特定のサーバー・アドレスを使用して別々に処理する必要があります。フェイルオーバーが発生した場合、アクティブ接続は失われ、アプリケーションは新規 1 次サーバーに再接続する必要があります。

基本接続の構文

基本接続では、標準の solidDB 接続ストリング構文を使用します。

protocol_name [*options*] [*host_computer_name*] *server_name*

ここで、

- *options* には、以下のオプションを任意に組み合わせて指定できます。

表 19. 接続ストリングのオプション

オプション	説明	プロトコル
-4	クライアントの接続が IPv4 プロトコルのみを使用することを指定します。	TCP/IP
-6	クライアントの接続が IPv6 プロトコルのみを使用することを指定します。 IPv6 プロトコルが使用される場合、Windows 環境では、このオプションは必須です。	TCP/IP
-i*source_address*	システムのデフォルト・ソース IP アドレス・バインディングがアプリケーションのニーズに合わない場合のために、明示的に接続するソケット・ソース・アドレスを指定します。 *source_address* には IP アドレスまたはホスト名を使用できます。	TCP/IP
-z	この接続でのデータ圧縮を可能にします。	すべて
-c *milliseconds*	ログイン・タイムアウトを指定します (デフォルトは、オペレーティング・システムに固有です)。指定された時間が経過すると、ログイン要求が失敗します。	TCP/IP
-r *milliseconds*	接続 (または読み取り) のタイムアウトを指定します。指定された時間内に応答が受信されないと、ネットワーク要求が失敗します。値 0 (デフォルト) を指定すると、タイムアウトの制限がなくなります (オペレーティング・システムのデフォルト・タイムアウトが適用されます)。	TCP/IP
-o*filename*	ネットワーク・トレース機能をオンにして、トレース出力ファイルの名前を定義します。 詳しくは、「*IBM solidDB 管理者ガイド*」の『*ネットワーク・トレース機能*』を参照してください。	すべて
-p*level*	指定されたレベル (0 - 5) でサーバーに ping します。 クライアントは、レベル 1 の solidDB Ping 機能をいつでも使用できます (0 はノーオペレーション/デフォルト)。レベル 2、3、4、または 5 は、サーバーで少なくとも同じレベルの Ping 機能を使用するように設定されている場合に限り使用できます。 詳しくは、「*IBM solidDB 管理者ガイド*」の『*Ping 機能*』を参照してください。	すべて
-t	ネットワーク・トレース機能をオンにします。 詳しくは、「*IBM solidDB 管理者ガイド*」の『*ネットワーク・トレース機能*』を参照してください。	すべて

- *host_computer_name* は、クライアントとサーバーを別のマシンで実行している場合に、TCP/IP プロトコルおよび名前付きパイプ・プロトコルで必要となります。
- *server_name* は、通信プロトコルによって異なります。
  - TCP/IP プロトコルでの server_name は、2315」などのサービス・ポート番号です。
  - その他のプロトコルでの server_name は、「soliddb」や「chicago_office」などの名前です。

  各種通信プロトコルでの構文について詳しくは、「*IBM solidDB 管理者ガイド*」の『*通信プロトコル*』を参照してください。

注:

- *protocol_name* と server_name は、サーバーがネットワーク listen 名で使用しているものと一致している必要があります。
- 接続ストリングを接続時に指定する場合は、引用符で囲む必要があります。
- 接続ストリングのすべてのコンポーネントでは、大/小文字を区別しません。

以下に例を示します。

```
Connect=tcp srv1.dom.acme.com 1315
```

4.3.1 アプリケーションから 1 次サーバーへの再接続

HotStandby 用のクライアント・アプリケーションの準備

1 次サーバーとの接続を失ったクライアント・プログラムは、新しい 1 次サーバー (旧 2 次サーバー) に再接続できなければなりません。クライアント・アプリケーションは、以下のことができるようにコーディングする必要があります。

1. 1 次サーバーを書き込みトランザクションにそれ以上使用できないことを認識する。
2. 他のサーバーに接続するか、前に作成された接続の使用に切り替える。
3. 現行の (中断された) トランザクションが失われるか打ち切られており、新しい 1 次サーバー上で再実行する必要があるかどうかを考慮する。

2 次サーバー・アドレスの取得

2 次データベース・サーバーの接続情報を取得する最も簡単な方法は、**ADMIN COMMAND 'hotstandby cominfo'** コマンドを使用することです。このコマンドは、HSB ペアの相手側サーバーの接続情報を返します。

このタスクについて

手順

1. アプリケーションは、初めて 1 次サーバーに接続したとき、**ADMIN COMMAND 'hotstandby cominfo'** コマンドを実行して、結果を保管できます。 **cominfo** コマンドが値を返した場合、それは 1 次サーバーと 2 次サーバーが現在接続されていることを意味するわけではありません。「cominfo」コマンドは、単に solid.ini 構成ファイルの **Connect** パラメーターで指定された値を返すか、**hsb parameter connect** コマンドで指定された最新の値を返します。1 次サーバーと

2 次サーバーの間の接続状態を確認する必要がある場合は、**ADMIN COMMAND 'hotstandby status connect'** を使用できます。

2. その後、1 次サーバーが障害を起こした場合、アプリケーションは保管されている情報を使用して、2 次サーバー (新しい 1 次サーバー) に接続できます。

クライアント・アプリケーションでの HotStandby サーバーの障害の検出

HotStandby (HSB) コンポーネントを使用するには、アプリケーションが、障害を起こした 1 次サーバーから 2 次 (新規 1 次) サーバーへ切り替える時期を知っている必要があります。これを行うには、いくつか方法があります。最良の方法は、単に呼び出した機能からの戻りコードを検査して、他のサーバーへ切り替える必要があることを示すエラーを受信したかどうかを調べることです。

また、サーバーの状態をモニターしてもかまいません (例えば、1 次サーバーを検査して、その状態が PRIMARY UNCERTAIN に変化したかどうかを調べます)。

別のサーバーへの切り替えを試みる必要があることを示すエラーには、以下のものがあります。

- 10013: トランザクションが読み取り専用である
- 10041: データベースが読み取り専用である
- 10047: レプリケーション・トランザクションが異常終了した
- 11002: ディスクが満杯である
- 11003: ファイル書き込みに失敗し、構成を超過した
- 14501: 操作が失敗した
- 14502: 無効な rpc パラメーター
- 14503: 通信エラー
- 14506: サーバーがクローズされた (例えば、現在 HSB コピーまたはネットコピー操作のターゲットとなっているため)
- 14510: 通信書き込み操作が失敗した
- 14511: 通信読み取り操作が失敗した
- 14518: 接続が切断された
- 14519: ユーザーが拒否された (例えば、何らかの管理操作のため)
- 14529: 操作がタイムアウトになった
- 20009: セッション・エラー、書き込み操作が失敗した
- 21306: サーバーを検出できず、接続が失敗した
- 21308: 接続が切断された (書き込みがコード ... で失敗した)
- 21318: 操作が失敗した (通常でない戻りコード)

ODBC アプリケーション

以下のエラー・メッセージは、(例えば、操作不能なデータベース・サーバーが原因で) 接続を確立できなかった ODBC アプリケーションへ返されます。

- SQLState = 08001 - Client unable to establish connection

さらに、以下の solidDB 通信エラー・メッセージが生成されます。

- 21306 - Server *'server_name'* not found, connection failed.

照会の実行と結果の取得など、操作と操作の間で接続が (例えば、ネットワーク障害などのために) 失敗した場合、以下のエラー・メッセージが返されます。

- SQLState = 08S01 - Communication link failure

JDBC アプリケーション

以下のエラー・メッセージは、(例えば、操作不能なデータベースが原因で) 接続を確立できなかった JDBC アプリケーションへ返されます。

- SQLState = 08001 - Unable to connect to data source.

照会の実行と結果の取得など、操作と操作の間で接続が (例えば、ネットワーク障害などのために) 失敗した場合、以下のエラー・メッセージが返されます。

- SQLState = 08S01 - Communication link failure

注:

ODBC および JDBC は、同じエラー・コード (08001) に異なるエラー・メッセージを使用します。

新規 1 次サーバーへのアプリケーションの切り替え

アプリケーションは、「旧 1 次」サーバーへトランザクションを送信できないことを検出した後、旧 1 次サーバーと旧 2 次サーバーに対して、PRIMARY ACTIVE、PRIMARY ALONE、STANDALONE のいずれかの状態のサーバーを検出するまで、ポーリングを行う必要があります。

ポーリングは、アプリケーションがサーバーへの接続を試み、接続が確立されたときは、サーバーの状況を検査することによって行われます。接続が成功した場合、クライアントは SQL 関数 HOTSTANDBY_STATE を使用してサーバー状態を要求できます。この SQL 関数については、106 ページの『HOTSTANDBY_STATE 関数の使用』のセクションで説明されています。

注意:
切り替え後、オープンされているすべてのデータベース・オブジェクト (準備済みステートメント、オープン・カーソル、トランザクションなど) は、アクティブでなくなります。このため、それらのオブジェクトを再び初期化する必要があります。また、テンポラリー表またはトランジエント表 (solidDB メイン・メモリー・エンジン機能) を使用していた場合、それらの表は新規 1 次サーバーでは空になります。

HOTSTANDBY_CONNECTSTATUS 関数の使用

アプリケーションから 1 次サーバーに再接続したとき、接続状況情報を検証するために、HOTSTANDBY_CONNECTSTATUS 関数を使用できます。この関数は、管理コマンド **`hotstandby status connect`** と等価です。

この関数は引数を持たず、以下のいずれかの状況値を返します。

表 20. HOTSTANDBY_CONNECTSTATUS 状況値

状況	説明
CONNECTED	接続はアクティブです。この状況は、1 次と 2 次の両方のサーバーから返されます。
CONNECTING	1 次サーバーが 2 次サーバーに接続中です。この状況は、1 次と 2 次の両方のサーバーから返されます。
CATCHUP	1 次サーバーが 2 次サーバーに接続しましたが、トランザクション・ログはまだ完全にはコピーされていません。この状況は、1 次と 2 次の両方のサーバーから返されます。
BROKEN	接続は切断されています。この状況は、1 次と 2 次の両方のサーバーから返されます。

HOTSTANDBY_STATE 関数の使用

1 次および 2 次サーバーのアプリケーション・ポーリングを実装するには、HOTSTANDBY_STATE 関数を使用できます。この関数は、管理コマンド **hotstandby state** と等価です。これは、アプリケーションがサーバーへ接続されたとき、アプリケーションが現行の HotStandby 状態を要求できるようにします。

注: この関数に引数はありません。この関数が返す可能性があるそれぞれの状態の説明については、69 ページの『3.4.8, HotStandby サーバー状態の検証』を参照してください。

サンプルの疑似コード

アプリケーションは、HSB 対応であろうとなかろうと、障害/異常終了を起こしたトランザクションをやり直せるエラー処理機能を備えている必要があります。

非 HSB 環境では、トランザクションは並行性競合 (オプティミスティック表) またはデッドロック (ペシミスティック表) のために異常終了する場合があります。アプリケーションは、それらのエラー状態をキャッチし、自動的にトランザクションを再試行するか、対話式ユーザーにトランザクションを再実行するよう依頼する必要があります。

障害を起こすか異常終了したトランザクションを処理するコードが既にアプリケーションに存在する場合、そのコードを拡張して HSB を使用するのは、比較的に簡単です。

非常に単純化した例では、非 HA 対応アプリケーション処理用の適正なエラー処理機能を備えたアプリケーション疑似コードは、以下のようになります。

```
BEGIN TRANSACTION
EXECUTE APPLICATION LOGIC
PREPARE & EXECUTE STATEMENTS
COMMIT TRANSACTION
IF ERROR OCCURRED
      IF ERROR == concurrency conflict or deadlock
            GO TO BEGIN TRANSACTION
      END IF
      other error handling
END IF ;
```

上記のアプリケーションを改良して HA 対応にするのは、非常に簡単です。アプリケーションが以下のことを実行できるよう、コードを追加する必要があります。

- 1 つだけでなく、2 つのサーバーのどちらにも接続でき、しかも、
- エラーが起きた場合は、現在の状態が PRIMARY ACTIVE、PRIMARY ALONE、STANDALONE のいずれかであるサーバーを検出する。

疑似コードは、以下のようになります。

```
BEGIN TRANSACTION
EXECUTE APPLICATION LOGIC
PREPARE & EXECUTE STATEMENTS
COMMIT TRANSACTION
IF ERROR OCCURRED
      IF ERROR == server unavailable for write transactions
            FIND CURRENT PRIMARY SERVER
            GO TO BEGIN TRANSACTION
      END IF
      IF ERROR == concurrency conflict or deadlock
            GO TO BEGIN TRANSACTION
      END IF
      IF ERROR == something else
            other error handling
      END IF
END IF
```

現行 1 次サーバーを検出するロジックも、非常に簡単です。単に両方のサーバーの現行状態を検査し (必要であれば再接続を試み)、どちらかが PRIMARY ACTIVE、PRIMARY ALONE、STANDALONE であれば、そのサーバーを現行 1 次サーバーとして設定します。どちらのサーバーも基準を満たさない場合は、しばらく待ってから現行サーバー状態の検査を再試行します。

4.3.2 2 次サーバーへの再接続

現行 2 次サーバー (稼働している場合) に接続したい場合もあります。アプリケーションは、2 次サーバーに対し、読み取り専用の照会をサブミットできます。これは、サーバー間でワークロードのバランスをとるのに役立つ場合があります。

アプリケーションは、読み取り専用モードでのみ、2 次サーバーのデータベースに接続できます。クライアントが (読み取り専用モードに限って) 2 次サーバーに接続できるようにするには、以下のように、各サーバーの `solid.ini` 構成ファイルの HotStandby セクションで、以下のパラメーター値を使用します。

- 1 次サーバー内で **`Connect`** パラメーター
- 2 次サーバー内で **`Listen`** パラメーター

次のコマンドを使用して、サーバーのパートナーの接続情報を取得することもできます。

```
ADMIN COMMAND 'hotstandby cominfo';
```

このように、現行 1 次サーバーに接続している場合は、cominfo 照会を使用して現行 2 次サーバーのアドレスを取得できます。

4.4 アプリケーションとサーバー間のタイムアウトの定義

このセクションでは、「アプリケーション読み取りタイムアウト」と「接続タイムアウト」の設定を、solid.ini の **Connect** パラメーター、または ODBC の SQLConnect 機能の connect ストリングを使用して構成する方法について説明します。

これらのタイムアウト値は、そのサーバーと、クライアント・アプリケーション (solidDB SQL エディター (**solsql**)、solidDB リモート制御 (**solcon**)、および HA マネージャーなど) の接続に適用されます。

4.4.1 アプリケーション読み取りタイムアウト・オプション

アプリケーション読み取りタイムアウト・オプションは、ロー・レベルのネットワーク RPC 読み取り操作で障害を検出するのに役立ちます。このタイムアウト設定は、物理ネットワーク内での読み取りに適用されます (TCP/IP プロトコルのみ)。

この RPC 読み取りタイムアウト (ODBC および JDBC で接続タイムアウトと呼ばれます) は、以下の方法で (ミリ秒単位で) 構成できます。デフォルトのタイムアウトは 0、つまり無限です (オペレーティング・システムのデフォルト・タイムアウトが適用されます)。

- クライアント・サイド **Com.ClientReadTimeout** パラメーター

  例えば、以下のように指定します。

  ```
  [Com]
  ;Set RPC read timeout to 1000 milliseconds (one second)
  ClientReadTimeout=1000
  ```

- クライアント・サイド **Com.Connect** パラメーター (**-r***milliseconds* オプション付き)

  例えば、以下のように指定します。

  ```
  [Com]
  ;Set RPC read timeout to 1000 milliseconds (one second)
  Connect=TCP -r1000 1313
  ```

  注: Watchdog などのクライアント・アプリケーションでは、RPC 読み取りタイムアウト (接続タイムアウトとも呼ばれます) を、**Com.Connect** パラメーターの中で **-r** オプションを使用して指定すると便利です。そうしなかった場合、特定のネットワーク障害タイプで無期限の待機が起きることがあります。

  注:

  [Com] セクション、[Watchdog] セクション、および [Hotstandby] セクション内の **Connect** パラメーターは、すべて目的が異なります。必ず、正しいものを編集してください。

- SQLConnect 関数の Connect ストリング (**-r** オプション)

  以下に例を示します。

  ```
  SQLConnect (hdbc, "TCP -r1000 1313", SQL_NTS,
  "dba", SQL_NTS, "dba", SQL_NTS);
  ```

上記の例では、定数 SQL_NTS は前のストリング (servername、username、または password) が標準のヌル終了ストリングとして渡されたことを示しています。

4.4.2 接続パラメーターでの -C オプションの指定

接続タイムアウト (ログイン・タイムアウトとも呼ばれます) の値は、solid.ini ファイルの [Com] セクションおよび [Watchdog] セクションで使用される **Connect** パラメーターの中で指定できます。この接続タイムアウトは、TCP/IP プロトコルについてのみ機能します。

構文は以下のとおりです。

Parameter = tcp *-cnumber-of-milliseconds [machine name] port_number*

ここで、*Parameter* は **Connect** または **Listen** です。

接続タイムアウトに値を指定しなかった場合、サーバーはオペレーティング・システム固有のデフォルト値を使用します。

注:

Watchdog などのクライアント・アプリケーションでは、接続タイムアウト値を、Connect パラメーターの中で **-c** オプションを使用して指定すると便利です。そうしなかった場合、特定のネットワーク障害タイプで、障害が検出される前に無期限の待機が起きることがあります。

例えば、以下のように指定します。

アプリケーション・ノード:

```
[Com] ;The server listens to port 1320, and the Connection timeout is 1000 ms.
Listen = tcpip -c1000 1320
```

4.5 HotStandby での SMA の構成

SMA を透過性接続 (TC) で使用している場合、1 次ノードまたは 2 次ノード上のアプリケーションは、SMA 固有の TC 接続情報構文を使用して、データベースに接続しなければなりません。

このタスクについて

SMA TC を使用する場合、各ノード上のアプリケーションは SMA 接続を使用してローカル・サーバーに接続可能になっている必要があります。また、リモート・サーバーへは、ネットワーク・ベースの接続を使用して接続可能になっている必要があります。

ホット・スタンバイを使用する SMA の TC 接続ターゲット・リストの形式は、以下のとおりです。

connect_target_list::=[SERVERS:]*sma_connect_string*, *network_connect_string*

ここで

```
sma_connect_string::= sma protocol_name port_number | pipe_name

network_connect_string::= protocol_name IP_address | host_computer_name
                                        port_number | pipe_name
```

さらに、ロード・バランシング・メソッドを LOCAL_READ (PREFERRED_ACCESS=LOCAL_READ) に設定しておく必要があります。

手順

1. ホット・スタンバイ・サーバーを 2 台セットアップします。
2. 両方のサーバーに SMA をセットアップします。
3. 両方のアプリケーションに、SMA 固有の接続ターゲット・リスト構文およびロード・バランシング属性 PREFERRED_ACCESS=LOCAL_READ を使用して、TC 接続を定義します。
4. アプリケーションをコンパイルして開始します。

例

solidDB がポート 1964 でリスニングしている host1 のアプリケーションの情報を接続します。

```
PREFERRED_ACCESS=LOCAL_READ SERVERS=sma tcp 1964, tcp host2 2315
```

solidDB がポート 2315 でリスニングしている host2 のアプリケーションのストリングを接続します。

```
PREFERRED_ACCESS=LOCAL_READ SERVERS=sma tcp 2315, tcp host1 1964
```

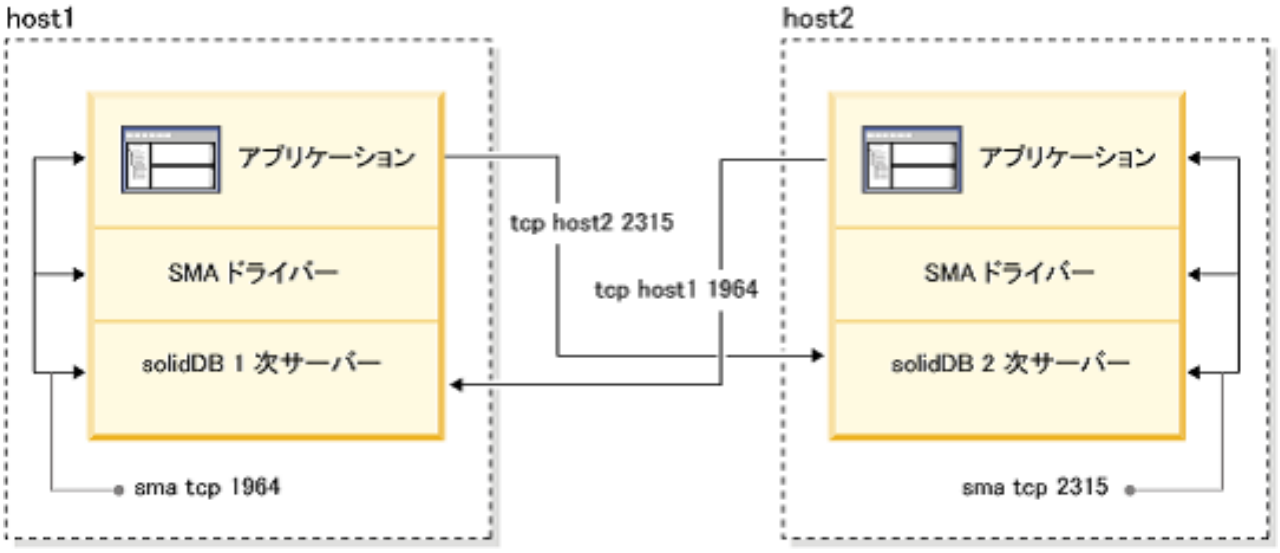


図 16. 例: SMA 構成におけるホット・スタンバイ

4.6 HotStandby を使用する拡張レプリケーションの構成

拡張レプリケーション・システムのすべてのノードは、solidDB HotStandby コンポーネントによって可用性を高めることができます。

拡張レプリケーション・システムのマスター・データベースとレプリカ・データベースが同期データである場合、データベース・サーバー・ペアの 1 次サーバー同士の間で同期化が発生します。言い換えれば、マスターの 1 次サーバーは、レプリカの 1 次サーバーと通信します。 19 ページの図 7 を参照してください。

データベース・サーバーは、どのような時点でもそのサーバーの 2 次サーバーへフェイルオーバーする可能性があり、データベース・サーバーが拡張レプリケーションを使用して別のサーバーとデータの同期をとっているときも、その例外ではありません。同期化の最中にフェイルオーバーが発生した場合、同期化メッセージの実行は停止し、そのプロセスはフェイルオーバー後に処理を再開する必要があります。エラーが起きた後に、同期化を再開する方法の詳細については、「*IBM solidDB 拡張レプリケーション・ユーザー・ガイド*」を参照してください。

マスター・データベースを格納しているサーバーを solidDB HotStandby でフォールト・トレラントにした場合、マスター・データベースのレプリカに両方のマスター・サーバーへの接続ストリングを知らせる必要があります。そのためには、それぞれのレプリカ・データベースで以下のステートメントを実行します。

```
SET SYNC CONNECT 'connect_string_to_server_1, connect_string_to_server_2'
TO MASTER master_nodename
```

下の図で、灰色の矢印は当初の 1 次サーバーへの当初の接続を表し、黒の矢印は新規 1 次サーバー (旧 2 次サーバー) への新規接続を表しています。旧 1 次サーバーとの同期に失敗した場合は、代替接続が使用されます。

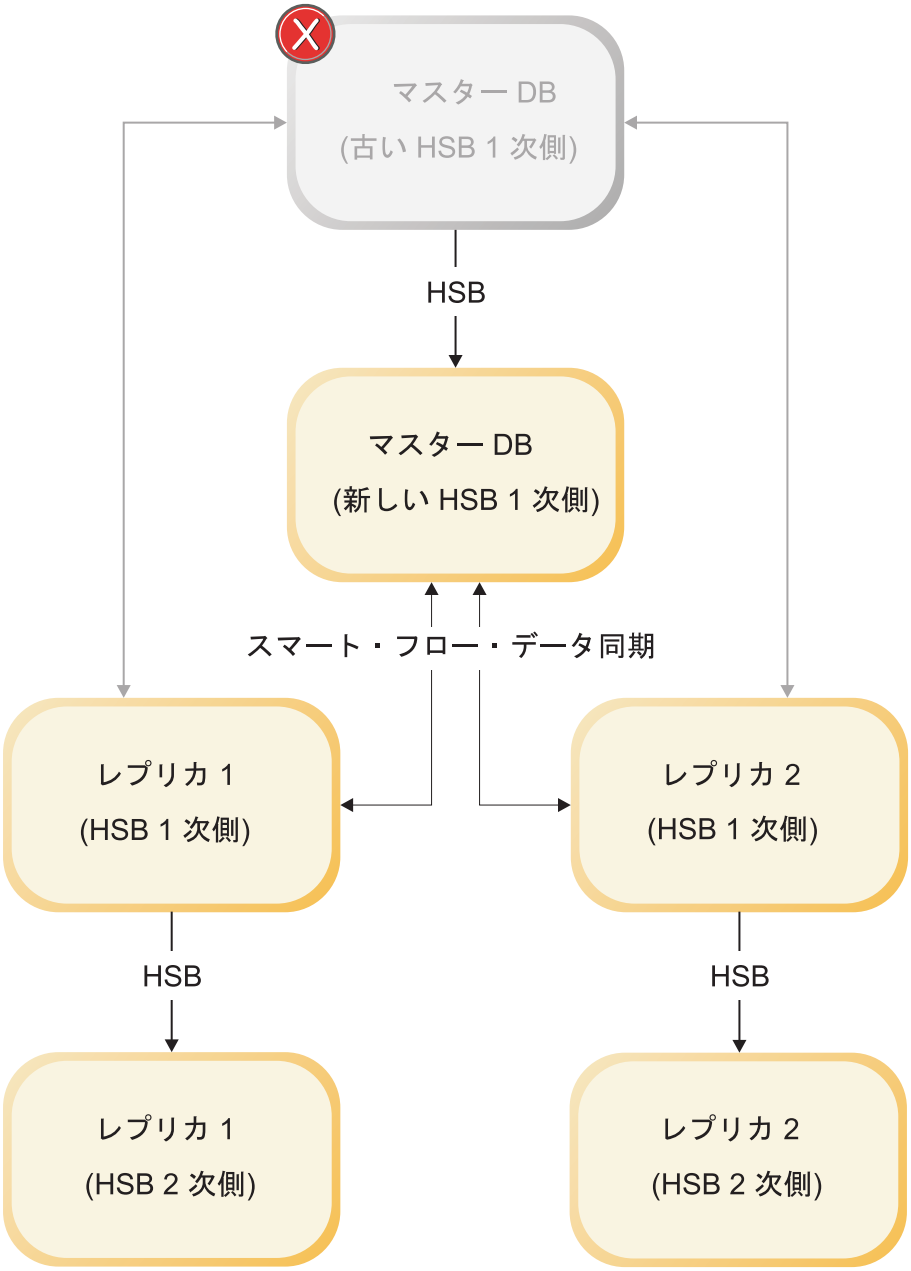


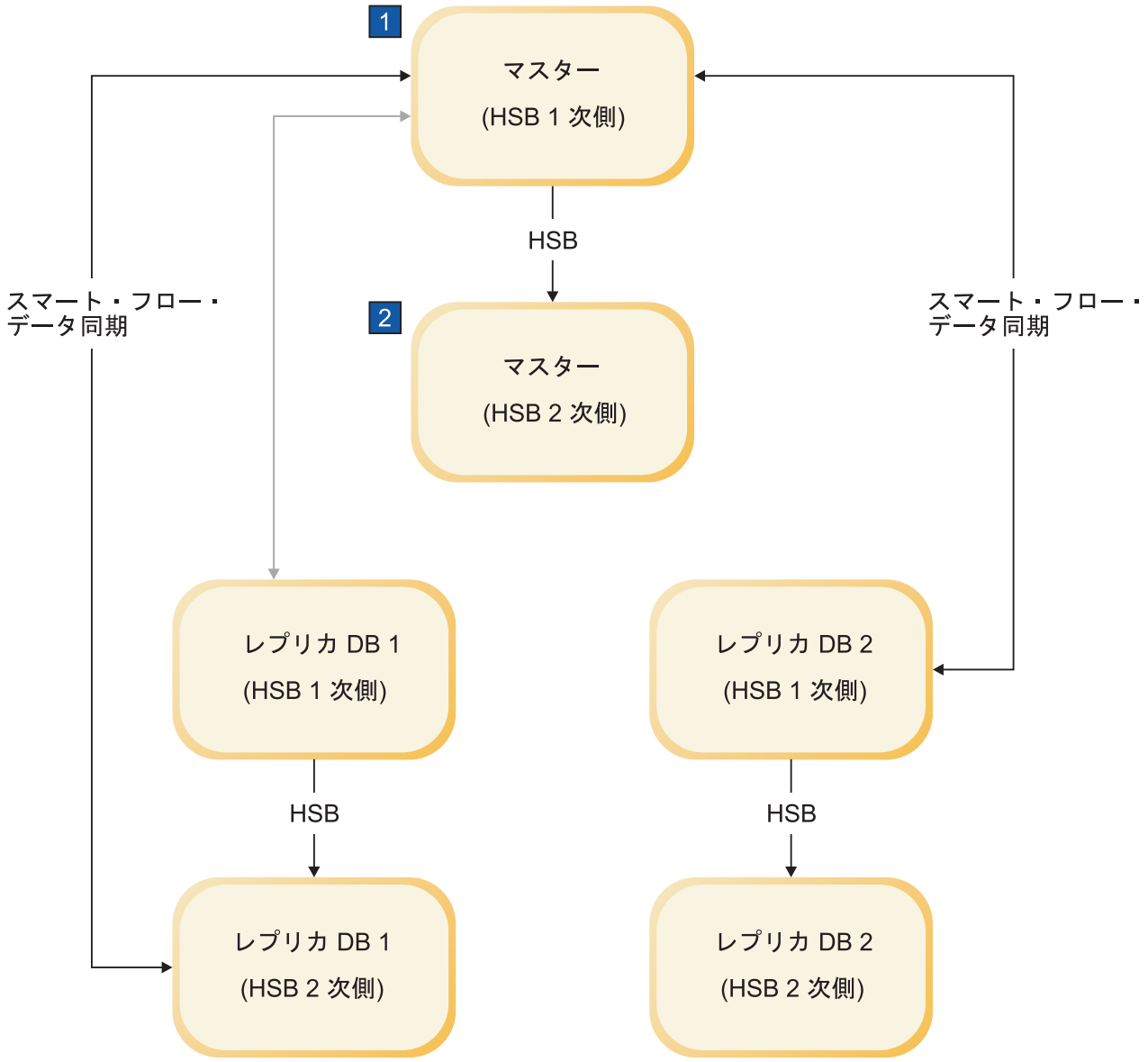
図 17. HotStandby と拡張レプリケーション: マスター・データベースのフェイルオーバー

solidDB HotStandby を使用しているサーバーがレプリカ・データベースを格納しているサーバーであり、マスター・サーバーがリモート・プロシージャー・コール (CALL *procedure_name* AT *node_name*) を使用してレプリカでプロシージャーを実行する (例えば、同期化を開始する) 場合、マスター・サーバーは、両方のレプリカ・サーバーへの接続ストリングを知らされている必要があります。一般に、マスター・サーバーはリモート・プロシージャー・コールを使用して、レプリカ・データベースとの同期化を開始します。マスターにレプリカ・サーバー・ペアへの接続ストリングを知らせるには、マスター・データベースで以下のステートメントを実行します。

```
SET SYNC CONNECT 'connect_string_to_server_1,
connect_string_to_server_2' TO REPLICA replica_nodename
```

別の方法として、ステートメントをレプリカ・サーバー内に保存しておき、それを次回に同期をとるときにマスターへ伝搬させることもできます。その場合は、以下のステートメントを使用します。

```
SAVE SET SYNC CONNECT 'connect_string_to_server_1,
connect_string_to_server_2' TO REPLICA replica_nodename
```



1. レプリカ・データベースでのフェイルオーバー
2. SET SYNC CONNECT 'tcp machine4 1315' REPLICA TO replicaDB1

図 18. HotStandby と拡張レプリケーション: レプリカ・データベースのフェイルオーバー

マスター・サーバーがレプリカ内でリモート・プロシージャー・コールを実行しない場合、上記のステートメントは必要ありません。

5 高可用性コントローラー (HAC) での障害処理

このセクションでは、高可用性コントローラー (HAC) を使用する場合に考えられる障害のシナリオ、およびそこからの標準的なリカバリー手順について説明します。HAC は、さまざまな障害シナリオを暗黙的に処理します。しかし、人間の管理者または *Watchdog* タイプのソフトウェア・プログラムは、さまざまな障害や初期化のシナリオ (要するに管理シナリオ) を処理することができます。

HAC は 1 次サーバーと 2 次サーバーをモニターする Watchdog タイプ・プログラムで、必要な場合には、それらのサーバーの状態を変更するコマンドを発行します。例えば、HAC は 1 次サーバーまたは 2 次サーバー自体が障害を起こした場合、またはそれらのサーバー間の通信リンクがダウンした場合にそれを判別できます。

リカバリーの目的は、障害を起こしたコンポーネントを操作に戻すことです。ときには、リカバリーの最中にさらに障害が発生します。通常、それらの障害が起きると、システムの可用性が制限されたまま (1 つのサーバーしか稼働せず)、人間の介入を待つ状態になります。よくあるリカバリー時の障害で、自動的に処理されないものは、以下のとおりです。

- 障害を起こしたデータベースが、再開が不可能なところまで壊れている
- キャッチアップを実行するのに十分な空きディスク・スペースがない

5.1 1 次サーバーのデータベースの障害

シナリオ

ノード 1 にある (PRIMARY ACTIVE 状態の) 1 次データベースが障害を起こします。

ノード 2 にある (SECONDARY ACTIVE 状態の) 2 次データベースが、ノード 1 の 1 次データベースへの接続障害を検出します。

リカバリー

1 次サーバーのデータベース障害からのリカバリーで、2 次サーバーは、1 次サーバーに置き換わります。リカバリーは、以下のように自動的に進行します。

1. 接続に障害が起きると、ノード 2 にある 2 次サーバーのデータベースは、自動的に SECONDARY ALONE 状態に移行します。
2. ノード 2 にある 2 次サーバーのデータベース上の HAC インスタンスは、ノード 1 にある 1 次サーバーのデータベースが障害を起こしたと結論付け、ノード 2 にある 2 次サーバーのデータベースを PRIMARY ALONE 状態に設定します。
3. 上記のタスクと並行して、ノード 1 にある 1 次サーバーのデータベース上の HAC インスタンスは、1 次サーバーのデータベースを再開し、そのデータベースは SECONDARY ALONE 状態に入ります。

4. ノード 2 にある 2 次サーバーのデータベース上の HAC インスタンスは、1 次サーバーと 2 次サーバーのデータベースの接続プロセスを開始します。
5. キャッチアップが行われます。

   オプションとして、接続プロセスにはノード 2 にある 2 次サーバーのデータベースからノード 1 にある 1 次サーバーのデータベースへの **netcopy** 操作が含まれます。

5.2 2 次サーバーのデータベースの障害

シナリオ

ノード 2 にある (SECONDARY ACTIVE 状態の) 2 次データベースが障害を起こします。

ノード 1 にある (PRIMARY ACTIVE 状態の) 1 次サーバーのデータベースが、ノード 2 にある 2 次サーバーのデータベースへの接続障害を検出します。

リカバリー

2 次サーバーのデータベース障害からのリカバリーで、2 次サーバーが再始動されます。リカバリーは、以下のように自動的に進行します。

1. 接続に障害が起きると、ノード 1 にある 1 次サーバーのデータベースは自動的に PRIMARY UNCERTAIN 状態に移行するか、**AutoPrimaryAlone** パラメーターが使用可能に設定されている場合は PRIMARY ALONE 状態に移行します。
2. ノード 1 にある 1 次サーバーのデータベース上の HAC インスタンスは、ノード 2 にある 2 次サーバーのデータベースが障害を起こしたと結論付けます。
3. 1 次サーバーのデータベースがステップ 1 で PRIMARY UNCERTAIN 状態に設定された場合、HAC はこの時点でそれを PRIMARY ALONE 状態に設定します。
4. 上記のタスクと並行して、ノード 2 にある 2 次サーバーのデータベース上の HAC インスタンスは、2 次サーバーのデータベースを再開し、そのデータベースは SECONDARY ALONE 状態に入ります。
5. ノード 1 にある 1 次サーバーのデータベース上の HAC インスタンスは、1 次サーバーと 2 次サーバーのデータベースの接続プロセスを開始します。
6. キャッチアップが行われます。

   オプションとして、接続プロセスには 1 次サーバーのデータベースから 2 次サーバーのデータベースへの **netcopy** 操作が含まれます。

5.3 1 次ノードの障害

シナリオ

1 次ノード (ノード 1) が障害を起こします。

ノード 2 にある (SECONDARY ACTIVE 状態の) 2 次サーバーのデータベースが、ノード 1 にある (PRIMARY ACTIVE 状態の) 1 次サーバーのデータベースへの接続障害を検出します。

リカバリー

1 次サーバー・ノードの障害からのリカバリーで、1 次サーバーが再始動されます。リカバリーは、以下のように自動的に進行します。

1. 接続に障害が起きると、2 次サーバーのデータベースは、自動的に SECONDARY ALONE 状態に移行します。
2. ノード 2 にある 2 次サーバーのデータベース上の HAC インスタンスは、ノード 1 にある 1 次サーバーのデータベースが障害を起こしたと結論付けます。
3. ノード 2 にある 2 次サーバーのデータベース上の HAC インスタンスは、2 次サーバーのデータベースを PRIMARY ALONE 状態に設定します。
4. 1 次サーバー・ノード (ノード 1) が再始動されます。
5. 1 次サーバーのデータベース (ノード 1) 上の HAC インスタンスが再始動されます。
6. 1 次サーバーのデータベース (ノード 1) 上の HAC インスタンスは、1 次サーバーのデータベースが稼働していないと結論付けます。
7. 1 次サーバーのデータベース (ノード 1) 上の HAC インスタンスは、1 次サーバーのデータベースを再開し、それを SECONDARY ALONE 状態に設定します。
8. ノード 2 にある 2 次サーバーのデータベース上の HAC インスタンスは、1 次サーバーと 2 次サーバーのデータベースの接続プロセスを開始します。
9. キャッチアップが行われます。

   オプションとして、接続プロセスには 1 次サーバーのデータベースから 2 次サーバーのデータベースへの **netcopy** 操作が含まれます。

5.4 2 次ノードの障害

シナリオ

2 次ノード (ノード 2) が障害を起こします。

ノード 1 にある (PRIMARY ACTIVE 状態の) 1 次サーバーのデータベースが、ノード 2 にある (SECONDARY ACTIVE 状態の) 2 次サーバーのデータベースへの接続障害を検出します。

リカバリー

2 次サーバー・ノードの障害からのリカバリーで、2 次サーバーが再始動されます。リカバリーは、以下のように自動的に進行します。

1. 接続に障害が起きると、ノード 1 にある 1 次サーバーのデータベースは、自動的に PRIMARY UNCERTAIN 状態に移行します。
2. ノード 1 にある 1 次サーバーのデータベース上の HAC インスタンスは、ノード 2 にある 2 次サーバーのデータベースが障害を起こしたと結論付けます。

3. ノード 1 にある 1 次サーバーのデータベース上の HAC インスタンスは、1 次サーバーのデータベースを PRIMARY ALONE 状態に設定します。
4. 2 次サーバー・ノード (ノード 2) が再始動されます。
5. 2 次サーバーのデータベース (ノード 2) 上の HAC インスタンスが再始動されます。
6. 2 次サーバーのデータベース (ノード 2) 上の HAC インスタンスは、2 次サーバーのデータベースが稼働していないと結論付けます。
7. 2 次サーバーのデータベース (ノード 2) 上の HAC インスタンスは、2 次サーバーのデータベースを再開し、それを SECONDARY ALONE 状態に設定します。
8. ノード 1 にある 1 次サーバーのデータベース上の HAC インスタンスは、1 次サーバーと 2 次サーバーのデータベースの接続プロセスを開始します。
9. キャッチアップが行われます。

   オプションとして、接続プロセスには 1 次サーバーのデータベースから新しい 2 次サーバーのデータベースへの **netcopy** 操作が含まれます。

5.5 HotStandby リンクの障害

シナリオ

HotStandby リンクが障害を起こします。

ノード 1 にある (PRIMARY ACTIVE 状態の) 1 次サーバーのデータベースと、ノード 2 にある (SECONDARY ACTIVE 状態の) 2 次サーバーのデータベースの両方が、相互の接続の障害を検出します。

リカバリー

HotStandby リンクの障害からのリカバリーで、各 HAC インスタンスは障害を起こしたのがネットワークなのか、それとも相手側サーバーなのかを知るために、外部参照エンティティー (ERE) に対して ping を実行します。リカバリーは、以下のように自動的に進行します。

1. 接続に障害が起きると、両方のデータベースは自動的にそれぞれ PRIMARY UNCERTAIN (ノード 1) と SECONDARY ALONE 状態 (ノード 2) の状態に移行します。
2. 両方の HAC インスタンスからリモート・サーバーへの直接接続は、失敗します。
3. 両方の HAC インスタンスは、オペレーティング・システムの ping ユーティリティーを使用して、ERE に対して ping を実行します。
4. ping が失敗すると、ローカル・サーバーはそのまま保持されるか、SECONDARY ALONE 状態に設定されます。
5. ping が成功すると、成功した HAC はリモート・データベース・サーバーに接続しようとします。

6. リモート・データベース・サーバーへの接続の試みが失敗した場合、HAC はそのネットワーク接続の部分は操作可能であると結論付け、ローカル・サーバーを PRIMARY ALONE 状態に設定します。
7. 1 次サーバーのデータベース上の HAC インスタンスは、2 次サーバーのデータベースへの接続を再確立しようとします。
8. ネットワークが操作可能になり、接続が成功した場合、1 次サーバーのデータベースと 2 次サーバーのデータベースは自動的にそれぞれ、PRIMARY ACTIVE 状態と SECONDARY ACTIVE 状態に移行します。

5.6 サーバーが外部クライアントに応答しない

シナリオ

サーバーへの接続が失敗するか、永遠にハングします。サーバーは PRIMARY ACTIVE 状態と SECONDARY ACTIVE 状態ですが、クライアントはそれらのサーバーに接続できず、サーバーはトランザクションを実行できません。

リカバリー

応答しないサーバーの状態に関係なく、HAC は `solidhac.ini` 構成ファイル内で **`LocalDB.UnresponsiveActionScript`** パラメーターによって構成されたスクリプトを実行します。スクリプトは、応答しない solidDB プロセスのプロセス ID が組み込まれた単一パラメーターを使用して開始されます。

一般的な解決方法は、プロセス ID によって識別されたプロセスを強制終了することです。その場合、リカバリーは以下のように自動的に進行します。

応答しないサーバーが 1 次サーバー・ロールだった場合:

1. 接続に障害が起きると、2 次サーバーのデータベース (ノード 2) は、自動的に SECONDARY ALONE 状態に移行します。
2. 2 次サーバーのデータベース (ノード 2) 上の HAC インスタンスは、1 次サーバーのデータベース・プロセス (ノード 1) が終了したことを感知し、2 次サーバーのデータベース (ノード 2) を PRIMARY ALONE 状態に設定します。
3. 上記のタスクと並行して、1 次サーバーのデータベース (ノード 1) 上の HAC インスタンスは旧 1 次サーバーのデータベースを再始動し、そのデータベースは SECONDARY ALONE 状態に入ります。
4. 2 次サーバーのデータベース (ノード 2) 上の HAC インスタンスは、1 次サーバーと 2 次サーバーのデータベースの接続プロセスを開始します。
5. キャッチアップが行われます。

   オプションとして、接続プロセスには 2 次サーバーのデータベース (ノード 2) から 1 次サーバーのデータベース (ノード 1) への **netcopy** 操作が含まれます。

応答しないサーバーが 2 次サーバー・ロールだった場合:

1. 接続に障害が起きると、1 次サーバーのデータベース (ノード 1) は自動的に PRIMARY UNCERTAIN 状態に移行するか、AutoPrimaryAlone パラメーターが使用可能に設定されている場合は PRIMARY ALONE 状態に移行します。

2. 1 次サーバーのデータベース (ノード 1) 上の HAC インスタンスは、2 次サーバーのデータベース・プロセス (ノード 2) が終了したことを感知します。
3. 1 次サーバーのデータベースがステップ 1 で PRIMARY UNCERTAIN 状態に設定された場合、HAC はこの時点でそれを PRIMARY ALONE 状態に設定します。
4. 上記のタスクと並行して、2 次サーバーのデータベース (ノード 2) 上の HAC インスタンスは旧 2 次サーバーのデータベースを再始動し、そのデータベースは SECONDARY ALONE 状態に入ります。
5. 1 次サーバーのデータベース (ノード 1) 上の HAC インスタンスは、1 次サーバーと 2 次サーバーのデータベースの接続プロセスを開始します。
6. キャッチアップが行われます。

   オプションとして、接続プロセスには 1 次サーバーのデータベースから 2 次サーバーのデータベースへの **netcopy** 操作が含まれます。

6 HotStandby サーバーのアップグレード

6.1 HotStandby サーバーのアップグレード (マイグレーション)

マイグレーション には、ソフトウェアのバージョンの更新が伴います。solidDB HotStandby のような高可用性システムでは、マイグレーションは *コールド* の場合と *hot* の場合があります。

コールド・マイグレーション は、システム全体 (両方のサーバー) をシャットダウンし、システムをアップグレードして、新しいソフトウェアと構成データを使用して再始動することを意味します。

ホット・マイグレーション は、HotStandby サーバーのペアのアップグレード作業の間、システム全体をオフラインにすることなくアップグレードを行うことを意味します。一方のサーバーを更新している間、もう一方のサーバーを作動させておくことができます。

重要: ホット・マイグレーションでは、システム全体はダウンしませんが、基本接続を使用している場合、ユーザーまたはアプリケーションは一方のサーバーから切断し、もう一方のサーバーに接続しなければならない場合があります。透過接続を使用している場合は、ユーザーおよびアプリケーションに意識されることなく、ホット・マイグレーションを行うことができます。

6.2 HSB 互換バージョン間のマイグレーション

マイグレーション前後の solidDB サーバーのバージョンが HSB 互換である場合、HotStandby ペアのサーバーは、バージョンは異なってもホットマイグレーション手順の間相互に通信することができます。例えば、solidDB サーバーの新規バージョンは、(少なくとも) 2 つ前のレベルのバージョンと HSB 互換です。

コールド・マイグレーション

コールド・マイグレーションの基本的な手順は、以下のとおりです。

1. システム全体をシャットダウンします。
2. サーバーをアップグレードします。
3. シャットダウン前と同じロールを使用して、新しいソフトウェア・バージョンのサーバーを再始動します。

ホット・マイグレーション

ホット・マイグレーションの基本的な手順は、以下のとおりです。

1. 2 次サーバーを切断し、シャットダウンしてから 2 次サーバーをアップグレードします。
2. 旧 2 次サーバーを新規 1 次サーバー (PRIMARY ALONE 状態) になるように設定します。次に、旧 1 次サーバーをシャットダウンし、旧 1 次サーバーをアップグレードします。

3. 旧 1 次サーバーを新規 2 次サーバーとして復帰させます。新規 1 次サーバーと新規 2 次サーバーを接続し、新規 2 次サーバーを新規 1 次サーバーに「キャッチアップ」させます。

注: ホット・マイグレーション手順ではサーバーが逆になり、この一連の手順が終わると、当初 1 次サーバーだったサーバーが 2 次サーバーになり、2 次サーバーだったサーバーが 1 次サーバーになります。

6.3 HSB 非互換バージョン間のマイグレーション

マイグレーション前後の solidDB サーバーのバージョンが HSB 非互換である場合、HotStandby ペアのサーバーは、ホット・マイグレーション手順の間通信できません。

ヒント: HSB 非互換のバージョン間でのマイグレーション手順は、例えば、ハードウェアやオペレーティング・システムをアップグレードする場合にも使用できます。

6.3.1 HSB 非互換バージョン間のマイグレーションの準備手順

このセクションでは、HSB 非互換のバージョン間でのマイグレーションの準備ステップについて、典型的なアップグレード・シナリオの例を使用して説明します。この例では、アプリケーションがそれぞれの接続の状態を検知して、新規 1 次サーバーに自動的にフェイルオーバーすることを想定しています。このため、制御されたサーバー切り替えにより、アプリケーションは中断されません。ただし、オープン・トランザクションが切り替え中に打ち切られる場合があります。

1. アプリケーションが自動的にフェイルオーバーするように設計されていない場合は、接続が失われること、および新規 1 次サーバーに再接続する必要があることを、ユーザーに通知します。
2. システムとソフトウェアをアップグレードのために準備します。特に、以下のような作業を行う必要があります。
   a. アップグレード操作の一部で、それぞれの solidDB サーバーが単独で (特に PRIMARY ALONE 状態で) 作動することになるため、両方のコンピューターが健全な状態であることを確認する必要があります。例えば、これらには、十分な空きディスク・スペース、信頼できるネットワーク接続、および電源障害に備えた UPS 装置がある必要があります。
   b. それぞれのサーバーは、もう一方のサーバーがアップグレードされている間、少なくともある短い時間、PRIMARY ALONE 状態で作動します。サーバーは、PRIMARY ALONE 状態にある間、トランザクションをトランザクション・ログに保管します。もう一方のサーバーがアップグレードされている間 (そのサーバーが再始動後にキャッチアップに要する時間も含む) に発生するすべてのトランザクションを保管するために、ログ・ファイル用の十分なディスク・スペースが使用可能でなければなりません。
   c. アップグレード・ソフトウェアのコピーが各コンピューター上にあるか、すぐに入手可能であることを確認します。
3. Watchdog プログラムがある場合は、一時的にオフにしてください。これは、アップグレード・プロセス中に発行するコマンドと競合するコマンドを発行しないようにするためです。この例として、1 次サーバーを 2 次サーバーから切断し

た後、2 次サーバーをアップグレードする前に Watchdog がサーバー同士の再接続を試みないようにしたい場合があります。

6.3.2 コールド・マイグレーション手順

このタスクについて

マイグレーション・ステップは以下のとおりです。

手順

1. サーバー同士を切断し、両方ともシャットダウンします。
2. 新バージョンのソフトウェアをインストールします。
3. `solid.ini` ファイルを更新します。
4. コマンド行パラメーター **`-x autoconvert`** を指定して 1 次サーバーを始動します。このパラメーターは、サーバーに既存のデータベースを新しいフォーマットに変換するよう指示します。
5. 1 次サーバーを PRIMARY ALONE 状態に設定します。
6. 1 次サーバーから 2 次サーバーへの 'hsb copy' または 'hsb netcopy' を実行します。
7. サーバー同士を接続します。

6.3.3 ホット・マイグレーション手順

このタスクについて

以下の手順で、S1 (「OP」) および S2 (「OS」) は、*当初の* 1 次サーバーと 2 次サーバーを表しています。各サーバーの状態は、このプロセスを進めるにつれて変化します。

手順

1. S1: 1 次サーバーを 2 次サーバーから切断します。

   ```
   ADMIN COMMAND 'hsb set broken';
   ADMIN COMMAND 'hsb status connect ping';
   ```

2. S2 OS: サーバー S2 (2 次サーバー) をシャットダウンします。

   ```
   ADMIN COMMAND 'shutdown force';
   ```

3. S1 OP: サーバー S1 (1 次サーバー) に、PRIMARY ALONE 状態で作動するよう指示します (まだ自動的にその状態に切り替わっていない場合)。サーバー S1 (1 次サーバー) が PRIMARY ALONE 状態にあることを検証します。

   ```
   ADMIN COMMAND 'hsb set primary alone';
   ADMIN COMMAND 'hsb state';
   ```

4. S2 OS: サーバー S2 (当初の 2 次サーバー) をアップグレードします。

   ソフトウェアを更新するだけでなく、`solid.ini` ファイル内の構成パラメーターも更新してください。

5. S2 OS: **`-x migratehsbg2`** コマンド行スイッチを使用して、サーバー S2 (当初の 2 次サーバー) を起動します。

   ```
   solid -x migratehsbg2
   ```

このコマンド行スイッチには 2 つの効果があります。まず、サーバーに、既存のデータベースを受け入れて変換するよう指示します (**-x autoconvert** パラメーターと同じ効果)。また、新規 2 次サーバーが旧 1 次サーバーと古いレプリケーション・プロトコルで通信できるようにします。 サーバーは SECONDARYALONE 状態で起動します。

6. S1 OP: サーバー S1 (1 次サーバー) を検査し、まだ PRIMARY ALONE 状態であることを確認します。

   ```
   ADMIN COMMAND 'hsb state';
   ```

7. S1 OP: **hsb connect** コマンドを実行して、1 次サーバーを 2 次サーバーに接続します。

   ```
   ADMIN COMMAND 'hsb connect';
   ```

   2 次サーバーが新しい方のバージョンのサーバーを実行している場合は、2 次サーバーから接続できないことに注意してください。

   このステップにより、2 次サーバーがダウンしている間に発生したデータ変更に 2 次サーバーが「キャッチアップ」するプロセスが開始されます。

8. S1 OP: 続行する前に、「キャッチアップ」が完了するのを待ちます。キャッチアップが失敗した場合は、以下を行ってください。
   a. サーバー S2 (2 次サーバー) をシャットダウンします。
   b. S1 (1 次サーバー) から **hsb copy** を実行して、データベース全体をサーバー S2 にコピーします。

      注: **hsb netcopy** でなく **hsb copy** を使用する必要があります。**hsb netcopy** は異なるサーバー・バージョン間では機能しないからです。

   c. 旧バージョンのサーバー (S2) でコピーをリカバリーします。
   d. S2 (2 次サーバー) をシャットダウンします。
   e. 前のステップに戻ります。

9. S1 OP: サーバー同士が接続され、キャッチアップされた後、以下を実行します。

   ```
   ADMIN COMMAND 'shutdown force';
   ```

   注意:
   force オプションは、オープン・トランザクションをすべて打ち切ります。
   サーバー S2 (2 次サーバー) がキャッチアップした後、それを新規 1 次サーバーにします。サーバー S1 (前の 1 次サーバー) を、アップグレードのためにシャットダウンします。

10. S2 OS: 新規 1 次サーバー S2 (旧 2 次サーバー) を PRIMARY ALONE 状態で作動するように設定します (まだ自動的にその状態に切り替わっていない場合)。サーバー S2 が PRIMARY ALONE 状態にあるかどうかを検証します。

    ```
    ADMIN COMMAND 'hsb set primary alone';
    ADMIN COMMAND 'hsb state';
    ```

11. S1 OP: サーバー S1 (当初の 1 次サーバー) をアップグレードします。

    ソフトウェアを更新するだけでなく、solid.ini ファイル内の構成パラメーターも更新してください。

12. S1 OP: サーバー S1 上で、solidDB サーバーを OFFLINE 状態で再始動します。

    ```
    solid -x backupserver
    ```

13. S2 OS: 新規 1 次サーバー S2 (旧 2 次サーバー) を検査し、まだ PRIMARY ALONE 状態であることを確認します。

    ```
    ADMIN COMMAND 'hsb state';
    ```

14. S2 OS: データベースを新規 1 次サーバー (S2) から新規 2 次サーバー (S1) にネットコピーします。

    ```
    ADMIN COMMAND 'hsb netcopy';
    ```

15. S2 OS: ネットコピーが成功したかどうかを検証します。

    ```
    ADMIN COMMAND 'hsb status copy';
    ADMIN COMMAND 'hsb connect';
    ```

    hsb connect コマンドは新規 1 次サーバーを新規 2 次サーバーに接続し、新規 2 次サーバーがダウンしている間に発生したデータ変更に、新規 2 次サーバーが「キャッチアップ」するプロセスが開始されます。

    このステップが失敗した場合は、データベース全体を (**hsb copy** を使用して) 2 次サーバーにコピーした後、ステップ 11 から再開してください。

6.3.4 アップグレード後のタスク

新規 2 次サーバーが新規 1 次サーバーにキャッチアップした後、システムは完全に通常の状態に戻ります。新規 1 次サーバーと新規 2 次サーバーは両方ともアップグレードされ、最新の現行データを持ちます。いくつかテストの照会を行い、すべてが正しく作動していることを確認するとよいでしょう。

手順

1. 1 次サーバーと 2 次サーバーが正しく機能していることをテストします。 例えば、以下の一連の操作を選択してもかまいません。

   1 次サーバー上で:

   ```
   ADMIN COMMAND 'hsb state';
   ADMIN COMMAND 'hsb status catchup';
   ```

   何らかの読み取り専用照会を発行します。

   2 次サーバー上で:

   ```
   ADMIN COMMAND 'hsb state';
   ADMIN COMMAND 'hsb status catchup';
   ```

   何らかの読み取り専用照会を発行します。

2. Watchdog プログラムを使用していた場合は、それを再開します。

付録 A. HotStandby 構成パラメーター

このセクションでは、HotStandby で使用される構成パラメーターを説明します。

パラメーターには、以下の 4 つのタイプがあります。

- サーバー・サイド solid.ini 構成ファイル内のサーバー・サイド・パラメーター
  - サーバー・サイドの solid.ini 構成ファイルには、[Cluster] および [HotStandby] という 2 つの HotStandby 固有セクションがあります。
  - その他のさまざまなパラメーターも HotStandby 機能に影響を与える可能性があります。
- クライアント・サイド solid.ini 構成ファイル内のクライアント・サイド・パラメーター
- solidhac.ini 構成ファイル内の高可用性コントローラー (HAC) の構成パラメーター
- HAManager.ini 構成ファイル内の高可用性マネージャー (HAM) 構成パラメーター

ヒント: solidDB パッケージに用意された Watchdog サンプルを使用する場合は、solid.ini の [Watchdog] セクションに Watchdog 固有のパラメーターも含まれています。詳しくは、187 ページの『付録 F. Watchdog サンプル』のセクションを参照してください。

アクセス・モード

パラメーターのアクセス・モードは、そのパラメーターの動的変更が可能であるかどうか、およびその変更がいつ有効になるかを定義します。パラメーターの値を ADMIN COMMAND で変更する場合、その変更は、即時にも、次にサーバーを始動する際にも適用されません。パラメーター値が solid.ini ファイルに書き込まれたとき、それは次回のサーバー始動時に有効になります。

アクセス・モード値

可能なアクセス・モードは以下のとおりです。

- *RO* (読み取り専用): 値の変更は不可です。現行値は必ず始動時の値と同一です。
- *RW*: ADMIN COMMAND で変更可能です。変更内容はすぐに有効になります。
- *RW/Startup*: ADMIN COMMAND で変更可能です。変更内容は次回のサーバー始動時に有効になります。
- *RW/Create*: ADMIN COMMAND で変更可能です。変更内容は新規データベースの作成時に有効になります。

関連概念

41 ページの『3.2, HotStandby の構成』
HotStandby は、1 次ノードと 2 次ノードの両方で solid.ini ファイルを使用して構成されます。 [HotStandby] セクションには、HotStandby 固有の構成パラメーターが含まれています。**Com.Listen** パラメーターなど、その他のセクションとパラメーターも設定する必要があります。

A.1 サーバー・サイド・パラメーター

A.1.1 Cluster セクション

表 21. Cluster パラメーター

[Cluster]	説明	ファクトリー値	アクセス・モード
ReadMostlyLoadPercentAtPrimary	1 次サーバーに向けられる読み取り負荷のパーセント	50	RW/Startup

A.1.2 HotStandby セクション

表 22. HotStandby パラメーター

HotStandby	説明	ファクトリー値	アクセス・モード
1SafeMaxDelay	1-safe レプリケーションの場合に、コミット済みのトランザクションが 2 次サーバーに送信されるまでの最大遅延 (ミリ秒)。	5000	RW
2SafeAckPolicy	これは、2 次サーバーが 1 次サーバーからのトランザクションを受信するときの確認応答のタイミングを指定します。 有効な値は以下のとおりです。 • 1 = 2-safe received。2 次サーバーはデータを受信すると応答します。 • 2 = 2-safe visible。2 次サーバーは、データが「可視」状態になったとき、つまり 2 次サーバーがトランザクションを実行したときに応答します。 • 3 = 2-safe durable。2 次サーバーは、データが永続的になったとき、つまりデータをコミットしてディスクに書き込んだときに応答します。 2-safe durable の安全性が最も高く、2-safe received の応答時間が最も速くなります。しかし、実際にはほとんどの場合、2-safe received モードでも十分なデータの安全性を確保できるため、安全性と速度のバランスが最も優れたモードと言えます。 このパラメーターは、サーバーが 2-safe レプリケーションを使用している場合にのみ適用されます。 **注:** このパラメーターは 2 次サーバーの動作を制御しますが、パラメーターの設定場所は 1 次サーバーになります。 2 次サーバーの solid.ini の値は無視されます。	1	RW

表 22. *HotStandby* パラメーター *(続き)*

HotStandby	説明	ファクトリー値	アクセス・モード
AutoPrimaryAlone	このパラメーターを Yes に設定すると、 2 次サーバーとの接続が切断されたときに、サーバーが自動的に PRIMARY ALONE 状態 (PRIMARY UNCERTAIN 状態ではない) になります。	no	RW
CatchupSpeedRate	サーバーはキャッチアップ実行中も、クライアントからのデータベース要求の処理も継続します。**CatchupSpeedRate** パラメーターは、アプリケーション要求への応答により重点を置き、キャッチアップの優先度をより低くするため、またはその逆にするために使用できます。 速度レートは、リンクおよび 2 次サーバーのスループットで決まる最大速度の何パーセントであるかで表します。数値が大きいほど、キャッチアップにより重点を置き、クライアント要求の処理の優先度を低くします。有効な値は 1 から 99 です。	50	RW
Connect	**Connect** パラメーターは、ペアになっているもう一方の HotStandby サーバーのアドレスを示します。 このパラメーターの値は、標準的な solidDB 接続ストリング (基本接続) または TC 固有接続ストリング (透過接続) です。 このパラメーターで定義された接続ストリングは、(**Com.Listen** パラメーターで定義された) 他の HotStandby サーバーのサーバー listen 名と一致している必要があります。 このパラメーターを HotStandby 用のサーバーで省略した場合は、ADMIN COMMAND を使用して、このパラメーターを動的に設定できます。サーバーが接続ストリングを持つまで、サーバーに可能な状態は、HotStandby 接続に関係しない状態、つまり PRIMARY ALONE、SECONDARY ALONE、および STANDALONE のみです。 **HSBEnabled** が No に設定されている場合、このパラメーターは無視されます。 マルチホーム・サーバーを持つ透過接続 (TC) の場合、**Connect** パラメーターは **HotStandby.TCConnect** パラメーターでオーバーライドできます。	ファクトリー値なし	RW

表 22. *HotStandby* パラメーター (続き)

HotStandby	説明	ファクトリー値	アクセス・モード

ConnectTimeout	接続タイムアウト値を指定することにより、HotStandby 接続操作でリモート・マシンに接続するまでの最大待ち時間を、秒単位で設定できます。
	
	ConnectTimeout パラメーター (一部のプラットフォームでのみ有効) は、特定の管理コマンドとの組み合わせでのみ使用します。これに該当するものは以下のとおりです。
	• **hotstandby connect**
	• **hotstandby switch primary**
	• **hotstandby switch secondary**
	
	例えば、タイムアウトを 30 秒 (30000 ミリ秒) に設定する場合には、以下のように指定します。
	`[HotStandby]`
	`ConnectTimeout=30000`
	
	PingTimeout も参照してください。
		`0` (タイムアウトなし)
		
		単位: ミリ秒
			RW

CopyDirectory	`[HotStandby]` セクションの **CopyDirectory** パラメーターは、ユーザーが以下のコマンドを実行したときに実行される HotStandby コピー操作用の名前と場所を定義します。
	`ADMIN COMMAND 'hotstandby copy';`
	
	例えば、パラメーターは以下のようになります。
	`[HotStandby]`
	`CopyDirectory=C:¥solidDB¥secondary¥dbfiles`
	
	CopyDirectory パラメーターに相対パスを指定する場合、そのパスは 1 次サーバーの `solid.ini` ファイルが存在するディレクトリーからの相対パスになります。
	
	このパラメーターにはファクトリー値はないため、ディレクトリーを `solid.ini` ファイルに指定しない場合は、コピー・コマンドに指定する必要があります。
	
	ADMIN COMMAND 'hotstandby netcopy' は、より柔軟性の高いソリューションなので、データベースをコピーする方法として推奨されます。
		ファクトリー値なし	RW

表 22. *HotStandby* パラメーター (続き)

HotStandby	説明	ファクトリー値	アクセス・モード
HSBEnabled	このパラメーターを yes に設定すると、サーバーを HotStandby 1 次サーバーまたは 2 次サーバーとして動作させることができます。このパラメーターを no に設定すると、サーバーは HotStandby サーバーとしては動作しません。 このパラメーターを yes に設定すると、サーバーを初めて始動するときに、サーバーのデフォルトの初期状態が SECONDARY ALONE になるよう暗黙的に定義することになります。有効な値は yes と no です。 HotStandby を使用するには、 **Connect** パラメーターも指定する必要があります。このパラメーターは、solid.ini ファイルに設定するか、または ADMIN COMMAND を使用して設定します。	no	RO (読み取り専用)
MaxLogSize	ディスク・ベース HSB ログの最大サイズ。ファクトリー値は無制限です。	0 単位: バイト、k = KB、m = MB	
MaxMemLogSize	ファイル・ベースのロギングが無効な場合 (**Logging.LogEnabled**=no)、 2 次サーバーに送信される前のトランザクションを保持しているインメモリー・ログのサイズ。値は、インメモリー・ログに空きがなくなるまで、サーバーが PRIMARY ALONE 状態を維持する時間に影響を与えます。	8M 単位: バイト、k = KB、m = MB	RO (読み取り専用)
NetcopyRpcTimeout	ネットコピー操作のデータ転送の確認応答タイムアウト (ミリ秒)	30000 単位: ミリ秒	RW
PingInterval	1 次サーバーおよび 2 次サーバーは、一定間隔で相互に「ping」メッセージを送信して、接続が維持されているかどうかを確認します。(これらの ping は、1 次サーバーが 2 次サーバーに送信するトランザクション情報とは無関係です。) 値は、サーバーから 2 回連続して送信される ping の間隔 (ミリ秒) に相当します。	1000 (1 秒) 単位: ミリ秒	RW
PingTimeout	このパラメーターは、他のサーバーがダウンまたはアクセス不能になっていると判断するまで、サーバーが待機する時間を指定します。 指定した時間 (ミリ秒) が経過すると、サーバーは接続に失敗したと判断し、それに応じて状態を変更します。 **ConnectTimeout** も参照してください。	4000 (4 秒) 単位: ミリ秒	RW
PrimaryAlone	このパラメーターは推奨されません。 **AutoPrimaryAlone** パラメーターを使用してください。	no	RW

表 22. *HotStandby* パラメーター (続き)

HotStandby	説明	ファクトリー値	アクセス・モード
SafenessLevel	このパラメーターは、レプリケーション・プロトコルの安全性レベルを設定します。 設定可能な値は 1-safe、2-safe および auto auto 値を使用することで、持続性レベルとの関連性に基づいて、安全性レベルを動的に変更することができます。 **SafenessLevel** を auto に設定し、SET DURABILITY コマンドまたは **DurabilityLevel** パラメーターを使用して持続性をリラックスに設定した場合、安全性レベルは 1-safe に設定されますが、持続性レベルをストリクトに設定したときには、安全性レベルは 2-safe に設定されます。しかし、**DurabilityLevel** を 2 (アダプティブ持続性) に設定した場合、 auto 設定は無効になり、安全性レベルは常に 2-safe になります。	2-safe	RW
SecondaryThreads	このパラメーターは、2 次サーバーが書き込み操作を処理するために使用するスレッドの数を定義します。 スレッドの最適な数は、環境によって異なります。原則としては以下のとおりです。 有効な値は 1 から 256 です。	4	RW/Startup
TCConnect	このパラメーターは、アプリケーションとサーバーが別々のネットワークを使用して相互に接続する必要がある場合 (例えば、マルチホーム・サーバーを使用する場合) に、透過接続 (TC) のペアのもう一方の HotStandby サーバーのアドレスを定義します。 アプリケーション接続の観点からは、**HotStandby.Connect** パラメーターで定義されたアドレスよりも、このパラメーターで指定されたアドレスが優先されます。したがって、TC 接続ではこのパラメーターで指定されたサーバー・アドレスが使用されますが、サーバー間の HotStandby 接続では **HotStandby.Connect** パラメーターで定義されたサーバー・アドレスが使用されます。	ファクトリー値なし	RW

A.2 クライアント・サイド・パラメーター

A.2.1 Com セクション

クライアント・サイドの solid.ini ファイル内の [Com] セクションには、HotStandby で使用できる **Com.Connect** パラメーターが含まれています。

表 23. クライアント・サイドの通信パラメーター

[Com]	説明	ファクトリー値
ClientReadTimeout	このパラメーターは、接続 (読み取り) のタイムアウトをミリ秒単位で定義します。指定した時間の間に応答を受け取らない場合、ネットワーク要求は失敗します。0 の値を指定すると、タイムアウトは無限に設定されます。この値は、接続ストリング・オプション `-r`、さらに ODBC 属性 SQL_ATTR_CONNECTION_TIMEOUT でオーバーライドすることができます。 **注:** このパラメーターは、TCP プロトコルにのみ適用されます。	0 (無限)
Connect	接続パラメーターで接続ストリングが明示的に指定されていない場合、**Connect** パラメーターは、クライアントが solidDB サーバーに接続する際のデフォルト・ネットワーク名 (接続ストリング) を定義します。この値は、SQLConnect() 呼び出しが空のデータ・ソース名で発行された場合も使用されます。 標準的な solidDB の接続ストリングのフォーマットは以下のとおりです。 `protocol_name [options] [host_computer_name] server_name` ただし、`options` と `server_name` は、通信プロトコルによって異なります。 **重要:** HotStandby と SMA のセットアップでは、追加の接続ストリング属性を使用して、透過接続 (TC) などの追加的な機能の指定を行います。 詳しくは、ネットワーク名と接続ストリングの構文を参照してください。	tcp localhost 1964
ConnectTimeout	**ConnectTimeout** パラメーターは、ログイン・タイムアウトをミリ秒単位で定義します。 この値は、接続ストリング・オプション `-c`、さらに ODBC 属性 SQL_ATTR_LOGIN_TIMEOUT でオーバーライドすることができます。 **注:** このパラメーターは、TCP プロトコルにのみ適用されます。	OS 固有
ODBCHandleValidation	このパラメーターは、ODBC ハンドル妥当性検査のオンとオフを切り替えます。 SQL_ATTR_HANDLE_VALIDATION ODBC 属性について詳しくは、「*IBM solidDB プログラマー・ガイド*」の『*ODBC ハンドル妥当性検査*』のセクションも参照してください。	no
Trace	このパラメーターを `yes` に設定すると、確立済みのネットワーク接続のネットワーク・メッセージに関するトレース情報が、**TraceFile** パラメーターで指定したファイルに書き込まれます。	no
TraceFile	**Trace** パラメーターを `yes` に設定すると、ネットワーク・メッセージに関するトレース情報はこのパラメーターで指定されたファイルに書き込まれます。 このトレース・ファイルは、トレースを開始した側がどちらかによって、サーバーまたはクライアントの現行作業ディレクトリーに出力されます。	`soltrace.out`

A.2.2 TransparentFailover セクション

TransparentFailover パラメーターは、クライアント・サイド・パラメーターです。

表 24. TransparentFailover パラメーター

[TransparentFailover]	説明	ファクトリー値
ReconnectTimeout	このパラメーターは、TF の切り替えまたはフェイルオーバーが生じた場合に、ドライバーが 1 次サーバーへの再接続を試行するまでの待ち時間 (単位はミリ秒) を指定します。ドライバーが新しい 1 次サーバー (再接続) を発見できなかった場合、エラーが返され、TF 接続は切断されます。	10000
WaitTimeout	このパラメーターは、ドライバーがサーバーの状態切り替えを待つ時間の長さ (単位はミリ秒) を指定します。ドライバーがサーバーへの再接続を試行するとき、接続するサーバーが中間的 (切り替え中あるいは不確定) な状態になっている可能性があります。	10000

A.3 高可用性コントローラー (HAC) パラメーター

このセクションでは、solidhac.ini 構成ファイルにある、高可用性コントローラー (HAC) の構成パラメーターを説明します。

HAC 構成ファイル solidhac.ini は、異なる 3 つのセクションに分かれています。これらのセクションを以下のセクションで説明します。パラメーターは構成ファイルにあるとおりの順序で並んでいます。

solidhac.ini の例については、セクション、 142 ページの『A.5.2, solidhac.ini 構成ファイル』を参照してください。

solidhac.ini 構成ファイルのパラメーターの名前、値、およびセクション見出しのフォーマットは、solid.ini 構成ファイルの場合と同じ規則に従っています。

[HAController] セクション

表 25. HAC 構成パラメーター: [HAController] セクション

パラメーター名	説明	必須	ファクトリー値
Listen	**Listen** パラメーターの値は、HAC が、solsql、solcon、および高可用性マネージャーとの通信に使用するプロトコルとポートを指定します。 例えば、Listen=tcp 3135 です。 • サポートされているプロトコルは、TCP/IP ('tcp') だけです。 • listen を開始できない場合、例えば、ポートが別のプロセスに使用されている場合は、HAC の終了直後に、情報がログ・ファイル (hacmsg.out) に書き込まれます。	X	

表 25. HAC 構成パラメーター: [HAController] セクション (続き)

パラメーター名	説明	必須	ファクトリー値
StartInAutomaticMode	このパラメーターは、HAC の初期モードを指定します。つまり、HAC が自動モードで実行を開始するかどうかを定義します。 HAC は、起動した後、AUTOMATIC と ADMINISTRATIVE のどちらかのモードになります。 • AUTOMATIC モード (yes) では、HAC は自動的に可用性の最大化を試み、サーバーの HSB 状態を変更し、必要な場合はサーバー・プロセスを再開します。 • ADMINISTRATIVE モード (no) では、HAC はサーバーの正常性のモニターのみを行います。 このパラメーターは動的に変更可能です。 可能な値は、yes および no です。 **注:** LocalDB セクションの **PreferredPrimary** パラメーターも参照してください。**PreferredPrimary** パラメーターは、**StartInAutomaticMode** パラメーターが yes の値を持つ場合にのみ有効です。		yes
EnableDBProcessControl	**EnableDBProcessControl**=yes を設定すると HAC でローカル・サーバー・プロセスを管理できます。これにより、サーバーを自動的に始動でき、ユーザーにはデータベース・プロセスのシャットダウンと再開を行うコマンドが提供されます。 この値は、HAC が AUTOMATIC モードの場合にのみ有効です。 可能な値は、yes および no です。 **注: EnableDBProcessControl**=yes を設定した場合、[LocalDB] セクションの **StartScript** パラメーターは必須になります。		no
EnableAutoNetcopy	**EnableAutoNetcopy**=yes を設定すると、**hsb connect** コマンドで HSB リンクを確立できないときに、HAC でネットコピーを開始することができます。 可能な値は、yes および no です。		yes
RequiredConnectFailures	このパラメーターは、接続試行が何回連続して失敗した後に、サーバーで障害が起きたと見なすかその回数を定義します。 サーバー状態が不明の場合、または HAC が何らかの理由でサーバー状態を確認する必要がある場合は、非ブロッキング **SQLConnect** (**check**) コマンドが使用されます。そのようなケースで非ブロッキング **SQLConnects** の実行が失敗した場合、問題のサーバーが無応答と見なされるまで、実行が複数回繰り返されます。 可能な値は 1 から無制限の数値です。		1
CheckTimeout	このパラメーターは、連続する CHECK モードの非ブロッキング **SQLConnect** コマンド間のタイムアウトをミリ秒単位で定義します。 値を非常に小さくすると、「誤判定」が起きることがあります。つまり、サーバーは稼働しているのに、タイムアウト期間内に応答できなかったために、障害を起こしているように見えます。 可能な値は 1 ミリ秒から無制限のミリ秒単位です。		150

表 25. HAC 構成パラメーター: [HAController] セクション (続き)

パラメーター名	説明	必須	ファクトリー値
CheckInterval	このパラメーターは、連続する非ブロッキング **SQLConnect** コマンドの間隔を定義します。この値は、フェイルオーバー時間に影響しません。一般に、検査 (ポーリング) は障害の後に行われるか、システムの始動中に行われます。 可能な値は 1 ミリ秒から無制限のミリ秒単位です。		1000
Username	HAC のユーザー名。	X	
Password	**Username** パラメーターで識別されるユーザーの HAC 用パスワード。	X	
DBUsername	HAC が接続するローカル HotStandby サーバーのユーザー名。 データベース・ユーザーは、SYS_ADMIN_ROLE または SYS_CONSOLE_ROLE を持っている必要があります。	X	
DBPassword	**DBUsername** パラメーターで識別されるユーザーのローカル HotStandby サーバー用パスワード。	X	
ApplicationConnTestUsername	アプリケーション接続テストで使用される接続のユーザー名を定義します (**EnableApplicationConnCheck** が yes に設定されています)。 **EnableApplicationConnCheck** が yes に設定されていて、このパラメーターの値が設定されなかった場合は、**DBUsername** の値が使用されます。		
ApplicationConnTestPassword	アプリケーション接続テストで使用される接続のパスワードを定義します (**EnableApplicationConnCheck** が yes に設定されています)。 **EnableApplicationConnCheck** が yes に設定されていて、このパラメーターの値が設定されなかった場合は、**DBPassword** の値が使用されます。		

[LocalDB] セクション

表 26. HAC 構成パラメーター: [LocalDB] セクション

パラメーター名	説明	必須	ファクトリー値
Connect	このパラメーターは、ローカル・データベース・サーバーの接続情報を定義します。 HAC はローカル・サーバーに接続するときに、この情報を使用します。 接続情報は、通信プロトコル (tcp) とサーバー・ポートで構成されます。例: tcp 2125。	X	

表 26. HAC 構成パラメーター: [LocalDB] セクション (続き)

パラメーター名	説明	必須	ファクトリー値
EnableApplicationConnTest	「yes」に設定した場合、アプリケーション接続の定期的なテストが使用可能になります。アプリケーション接続テストは、アプリケーションでサーバーへの接続に使用される接続が機能しているかどうかを検査します。 接続をテストするために、HAC はアプリケーションと同じ接続情報を使用してサーバーに接続し、単純なコマンドを実行してサーバーが応答することを確認します。 アプリケーション接続テストにより、HAC は solidDB サーバーの外部可用性をモニターできます。サーバーへの接続を確立できない場合、またはサーバーが単純な照会に応答しない場合、HAC はサーバーが提供するサービスは使用不可であると判断します。システム内にサーバー・プロセスが存在する場合、HAC はパラメーター **UnresponsiveActionScript** で指定されたスクリプトを呼び出します。 **重要:** アプリケーションの接続テストを有効にする前に、そのテストで使用するユーザー・アカウントを必ず作成しておいてください。アプリケーションの接続テストでユーザー名またはパスワードが無効であるために solidDB に接続できない場合、テストは、**ApplicationConnTestInterval** で指定した時間が経過するまで待機します。エラーは hacmsg.out にも出力されます。 また、**EnableApplicationConnTest** が yes に設定されていて、かつ、Application Connection Tester が使用するユーザー・アカウントがどちらかのサーバーから欠落している場合は、solidDB が大幅にスローダウンする可能性があります。これをリカバリーするには、以下の手順を実行します。 1. ADMIN COMMAND 'hac suspend' を実行して HAC 操作を中断します。 2. 両方のサーバーにユーザー・アカウントを作成します。 3. ADMIN COMMAND 'hac resume' を実行して HAC 操作を再開します。		no
ApplicationConnTestConnect	このパラメーターは、アプリケーション接続テスト接続の接続情報を定義します (**EnableApplicationConnCheck** は yes に設定されています)。 このパラメーターの値は、アプリケーションがサーバーへの接続に使用する接続情報と同じものであることが必要です。値は、通信プロトコル (tcp)、サーバー・アドレス、およびポート番号で構成されます。例: tcp -i10.0.0.101 2125。 **ApplicationConnTestConnect** が指定されなかった場合は、パラメーター **Connect** で定義された値が使用されます。		
EnableUnresponsiveActions	yes に設定した場合、アプリケーション接続テストが失敗した場合は、ユーザー提供のスクリプトが実行されます。 このスクリプトは、パラメーター **UnresponsiveActionScript** で定義されます。		no
RequiredAppConnTestFailures	このパラメーターは、サーバーを応答不能と見なすまでにアプリケーション接続テスト・コマンドを何回実行するか、その回数を定義します。		3

表 26. HAC 構成パラメーター: [LocalDB] セクション (続き)

パラメーター名	説明	必須	ファクトリー値
ApplicationConnTestTimeout	このパラメーターは、アプリケーション接続テストで使用される連続したコマンドのタイムアウトをミリ秒単位で定義します。		5000
ApplicationConnTestInterval	このパラメーターは、アプリケーション接続テストで使用される連続した非ブロッキング・コマンド間の間隔をミリ秒単位で定義します。		3000
StartScript	このパラメーターは、データベース・プロセスの開始に使用されるスクリプトの名前 (例えば、/home/soliddb/start_solid.sh) を宣言します。 このパラメーターは、HAC が、**EnableDBProcessControl** パラメーターを使用して、データベースを制御するように構成されている場合、必須パラメーターになります。つまり、以下のようになります。 • **EnableDBProcessControl** が yes に設定されている場合、このパラメーターは必須パラメーターです。 • **EnableDBProcessControl** が no に設定されている場合、このパラメーターは無効です。 **ヒント:** Linux および UNIX 環境では、sh に -x オプションを使用して、コマンドの実行時にコマンドおよびそれらの引数を出力することができます。 以下に例を示します。 `#!/bin/sh -x` 始動スクリプト (start_solid.sh) のログは、デフォルトでは /tmp/hac_start_solid.err に出力されます。 solidDB プロセスの開始ログは、デフォルトでは /tmp/hac_start_solid.out に出力されます。	説明を参照	

表 26. HAC 構成パラメーター: [LocalDB] セクション (続き)

パラメーター名	説明	必須	ファクトリー値

UnresponsiveActionScript	このパラメーターは、アプリケーション接続テストが失敗した場合に実行するアクションが入っているスクリプトの、名前と場所を定義します。例: /home/solid/terminate_solid.sh。
	
	スクリプトを呼び出すとき、HAC は solidDB プロセス ID をパラメーターとして指定する必要があります。HAC に solidDB プロセス ID が不明な場合、スクリプトを実行することはできません。
	
	このパラメーターは、**EnableApplicationConnTest** と **EnableUnresponsiveActions** の両方が yes に設定されている場合は必須です。
	
	例:
	
	次の terminate_solid.sh スクリプトは、Linux および UNIX オペレーティング・システムで、ID が 1 の solidDB プロセスを終了します。
	`#!/bin/sh`
	`#terminate_solid.sh`
	
	`ulimit -c unlimited`
	`kill -6 $1`
	`sleep 30`
	`kill -9 $1`
	
	ヒント: Linux および UNIX 環境では、sh に -x オプションを使用して、コマンドの実行時にコマンドおよびそれらの引数を出力することができます。
	
	以下に例を示します。
	`#!/bin/sh -x`
	
	アクション・スクリプト (terminate_solid.sh) のログは、/tmp/hac_terminate_solid.err に出力されます。 solidDB プロセスの終了ログは、/tmp/hac_terminate_solid.out に出力されます。
		説明を参照	

PreferredPrimary	このパラメーターは、両方のサーバーの *logpos* 値が等しい場合に、ローカル・サーバーが 1 次サーバーになるかどうかを定義します。両方のサーバーが **PreferredPrimary** に同じ値を持つ場合、最初のサーバーが新規 1 次サーバーになります。 このパラメーターは、**StartInAutomaticMode** パラメーターが yes の値を持つ場合にのみ有効です。 可能な値は、yes および no です。		no

[RemoteDB] セクション

表 27. HAC 構成パラメーター: [RemoteDB] セクション

パラメーター名	説明	必須	ファクトリー値
Connect	このパラメーターは、リモート・データベース・サーバーの接続情報を定義します。 HAC はリモート・サーバーに接続するときに、この情報を使用します。 リモート・データベースは、通信プロトコル、リモート・サーバー IP アドレス、およびそのポート (例えば、tcp 192.168.3.123 2125) を指定することによって、定義されます。	X	

[ERE] セクション

表 28. HAC 構成パラメーター: [ERE] セクション

パラメーター名	説明	必須	ファクトリー値
EREIP	このパラメーターは、外部参照エンティティーの IP アドレス (例えば、192.168.3.1) を識別します。 **RequiredPingFailures** パラメーターも参照してください。		
RequiredPingFailures	このパラメーターは、Ping 呼び出しが何回か連続して失敗したため、サーバーは ERE から切断され、クライアント・ネットワークから分離されていると HAC が結論付ける Ping 呼び出しの連続失敗の最大数を定義します。 このパラメーターは ERE でのみ使用可能です。 可能な値は 1 から無制限の数値です。		3

A.4 高可用性マネージャー (HAM) の構成パラメーター

このセクションでは、HAManager.ini 構成ファイルにある、高可用性マネージャーの構成パラメーターを説明します。

HAManager.ini 構成ファイルのパラメーターの名前、値、およびセクション見出しのフォーマットは、solid.ini 構成ファイルの場合と同じ規則に従っています。

表 29. 高可用性マネージャーの構成パラメーター

パラメーター名	説明
Header_text	このパラメーターは、HA マネージャーのヘッダー・テキストを定義します。この値は、ユーザー・インターフェースに表示されます。 これは必須パラメーターです。
Server1_host Server2_host	これらの 2 つのパラメーターは、HAC インスタンスのホスト名を定義します。

表 29. 高可用性マネージャーの構成パラメーター (続き)

パラメーター名	説明
Server1_name Server2_name	これらの 2 つのパラメーターは、HAC インスタンスの名前を定義します。
Server1_pass Server2_pass	これらの 2 つのパラメーターは、HAC インスタンスのパスワードを定義します。
Server1_port Server2_port	これらの 2 つのパラメーターは、HAC インスタンスのポートを定義します。
Server1_user Server2_user	これらの 2 つのパラメーターは、HAC インスタンスのユーザー名を定義します。
Window_title	このパラメーターは、HA マネージャーのウィンドウ・タイトルを定義します。この値は、ユーザー・インターフェースに表示されます。 これは必須パラメーターです。

A.5 構成ファイルの例

以下のセクションでは、HotStandby に関連するさまざまな構成ファイルの例を示します。

A.5.1 solid.ini 構成ファイル

以下は、最初の HotStandby サーバー用の solidDB 構成ファイル (solid.ini) の例 (抜粋) です。

```
[Com]
; The first server listens to the network with this
; name
Listen = tcp 1320
[HotStandby]
HSBEnabled=yes
; The first server connects to the second server
; using the following connect string.
Connect = tcp 188.177.166.12 1321
AutoPrimaryAlone=No
[Logging]
LogEnabled=yes
```

以下は、2 番目の HotStandby サーバー用の solidDB 構成ファイル (solid.ini) の例 (抜粋) です。

```
[Com]
; The second server listens to the network using the following
; connect string.
Listen = tcp 1321
[HotStandby]
HSBEnabled=yes
; The second server connects to the first server
; using the following connect string.
```

```
Connect = tcp  188.177.166.11 1320
AutoPrimaryAlone=No
[Logging]
LogEnabled=yes
```

A.5.2 solidhac.ini 構成ファイル

以下は高可用性コントローラー (HAC) 構成ファイル (solidhac.ini) の例 (抜粋) です。

```
;============================================
; NOTE : Copy this file as solidhac.ini
;        to solidhac working directory
;
; solidDB High Availability Controller inifile
;============================================

[HAController]
;** HAC connect info
;** HAC clients, HA Manager, for example, use this information.
;** Mandatory
;** Listen=tcp 3135
Listen=

;** Setting StartInAutomaticMode=Yes starts HAC in AUTOMATIC mode.
;** In AUTOMATIC mode, solidhac automatically tries
;** to maximize the availability by changing the HSB states of the
;** server, and restarting the server processes when necessary.
;** In contrast, it can be in ADMINISTRATIVE mode
;** in which HAC only monitors the health of the servers.
;**
;** This is dynamically changeable parameter.
;** Optional
;** Values : Yes/No, default = Yes
StartInAutomaticMode=

;** Setting EnableDBProcessControl=Yes allows solidhac
;** manage local db process by automatically starting
;** the db, and by providing the user with commands to
;** shutdown and restart db process.
;**
;** Optional
;** Effective only when HAC is in AUTOMATIC mode.
;** Values : Yes/No, default = No
EnableDBProcessControl=

;** Setting EnableAutoNetcopy=Yes allows solidhac to initiate
;** netcopy when HSB link cannot be established with 'hsb connect'.
;**
;** Optional
;** Effective only when HAC is in AUTOMATIC mode.
;** Values : Yes/No, default = Yes
EnableAutoNetcopy=

;** When server state is unknown, or HAC needs, for some other reason, to
;** ensure the state of server, non-blocking SQLConnect command is used.
;** If the execution of non-blocking SQLConnect in such a case fails,
;** it is repeated multiple (RequiredConnectFailures) times before
;** the server in question is considered as non-responsive.
;**
;** Optional
;** Values : 1..n, default=2
RequiredConnectFailures=

;** Timeout in milliseconds for non-blocking SQLConnect commands.
;** Too short interval can cause 'false positives', server seems
```

```
;** to be failed because it wasn't able to respond within the timeout period.
;**
;** Optional
;** Values : 1..n, default=150 (milliseconds)
CheckTimeout=

;** Interval between consecutive non-blocking SQLConnect commands.
;** The value doesn't affect on failover time. Checking (polling)
;** takes place typically after failure, or during system startup.
;**
;** Optional
;** default = 1000 (milliseconds)
CheckInterval=

;** HAC user identification
;** Mandatory
Username=
Password=

;** HSB server user identification
;** Mandatory
DBUsername=
DBPassword=

;** Identification for application connection test
;** These values are used when ApplicationConnectionTest
;** thread monitors the connection, and availability of
;** the server.
;** If values are not set, and
;** LocalDB.EnableApplicationConnCheck=Yes, then DBUsername, and
;** DBPassword are used.
;**
;** Optional
ApplicationConnTestUsername=
ApplicationConnTestPassword=


[LocalDB]
;** soliddb connect info
;** Mandatory
;** Connect=tcp 2125
Connect=

;** Enable periodical connection testing in the server.
;** In practice, HAC connects to the server, and executes
;** simple command(s) to ensure the responsiveness.
;**
;** Optional
;** default = No
EnableApplicationConnTest=

;** Connect info for applications, used in application connection test,
;** if it is enabled.
;**
;** Optional, if not specified, LocalDB.Connect is used.
;** ApplicationConnect=tcp 10.0.0.101 2125
ApplicationConnTestConnect=

;** Enables execution of the user-provided script when application
;** connection test fails. The script is defined with
;** UnresponsiveActionScript.
;**
;** Optional
;** default = No
EnableUnresponsiveActions=

;** Number of times the application connection test commands
```

```
;** are executed before the server is considered unresponsive.
;**
;** Optional
;** default = 3
RequiredAppConnTestFailures=

;** Timeout in milliseconds for consecutive application connection
;** test commands.
;**
;** Optional
;** default = 5000 (milliseconds)
ApplicationConnTestTimeout=

;** Interval between consecutive non-blocking application connection
;** test commands.
;**
;** Optional
;** default = 30000 (milliseconds)
ApplicationConnTestInterval=

;** The name of the script, which is used to initiate the db process.
;**
;** Optional, except if HAC controls db process (EnableDBProcessControl=Yes).
;** Value is not effective if EnableHACActions=No or EnableDBProcessControl=No
;** StartScript=/home/solid/start_solid.sh
StartScript=

;** The name and location of the script that contains the intended actions that
;** take place if application connection test fails.
;** When calling the script, HAC needs to specify the solidDB® process
;** identifier as a parameter. If HAC does not know the solidDB process id,
;** the script cannot be executed.
;**
;** Optional, except if ApplicationConnectionTest=Yes, and
;** EnableUnresponsiveAction=Yes
;**
;** UnresponsiveActionScript=/home/solid/terminate_solid.sh
UnresponsiveActionScript=

;** Setting PreferredPrimary=Yes moves local HSB server to as Primary
;** in the case where either of the servers could start as Primary.
;** If both servers have PreferredPrimary=No, or no value, first
;** (new Primary) server wins.
;**
;** Optional
;** Value is not effective if EnableHACActions=No.
;** Values : Yes/No, default no
PreferredPrimary=


[RemoteDB]
;** soliddb connect info
;** Mandatory
;** Connect=tcp 192.168.3.123 2125
Connect=


[ERE]
;** IP address of an ERE
;** Optional
;** Connect=192.168.3.1
EREIP=

;** The number of consecutive ping calls that must
;** fail before HAC concludes that the server is
;** disconnected from ERE.
;**
```

```
;** Optional
;** Values : 1..n, default=3
;** RequiredPingFailures=10
RequiredPingFailures=
```

A.5.3 HAManager.ini 構成ファイル

以下は、高可用性マネージャー構成ファイル (HAManager.ini) の例 (抜粋) です。

```
;================================================
 ;solidDB High-Availability Manager
; Configuration file HAManager.ini
; V. 0.3
; 2008-21-01
;================================================
;** HA Controller connect info, for example
;Server1_name = Server 1
;Server1_host = node1.acme.com
;Server1_port = 2220
;Server2_name = Server 2
;Server2_host = node2.acme.com
;Server2_port = 2220

; All the following lines are mandatory.
Window_title = HA Manager
Header_text  = SolidDB HA Manager

; Display names, host addresses and port numbers
; of the SolidHAC (HA Controllers) instances
;Server 1 HA Controller
;----------------------
Server1_name = Server 1
Server1_host = localhost
Server1_port = 1234
Server1_user = foo
Server1_pass = bar
;
;Server 2 HA Controller
;----------------------
Server2_name = Server 2
Server2_host = 192.168.0.1
Server2_port = 1234
Server2_user = foo
Server2_pass = bar
```

付録 B. HotStandby のエラー・コード

このセクションでは、HotStandby に関連するエラー・コードについて説明します。

solidDBのエラー・コードの完全なリストは、「*IBM solidDB 管理者ガイド* 」の付録、エラー・コード に記載されています。

このセクションで説明するエラーには、ADMIN COMMAND 結果セットの RC (戻りコード) 列の値と、ODBC または JDBC ドライバーのエラー・コードとして返されるエラーが含まれています。例えば、『B.1, HotStandby のエラーおよび状況コード』、および 157 ページの『B.2, 高可用性コントローラーのエラー・コードと状況コード』のセクションにあるエラーの大部分は ADMIN COMMAND 結果セットの値であり、すべての 162 ページの『B.5, solidDB 通信エラー』は、ドライバーが返すエラーです。

B.1 HotStandby のエラーおよび状況コード

solidDB HotStandby エラー (14009 - 147xx, 307xx) は HotStandby コマンドを使用したときに発生します。

HotStandby の solidDBサーバー・エラー

このセクションでは、HotStandby に関連する solidDB サーバー・エラーのリストを示します。サーバー・クラスでのエラーの完全なリストは、「*IBM solidDB 管理者ガイド*」の『*solidDB サーバー・エラー*』のセクションに記載されています。

表 30. HotStandby の solidDBサーバー・エラー

コード	クラス	タイプ	説明
14003	サーバー	戻りコード	ACTIVE コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby status switch'** • **ADMIN COMMAND 'hotstandby status catchup'** • **ADMIN COMMAND 'hotstandby status copy'** 意味: 切り替えプロセス、キャッチアップ・プロセス、コピーまたはネットコピー・プロセスが、まだアクティブです。
14007	サーバー	戻りコード	CONNECTING コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby status connect'** 意味: 1 次サーバーおよび 2 次サーバーは接続プロセスを実行中です。

表 30. HotStandby の solidDBサーバー・エラー (続き)

コード	クラス	タイプ	説明
14008	サーバー	戻りコード	CATCHUP コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby status connect'** 意味: 1 次サーバーが 2 次サーバーに接続しましたが、トランザクション・ログはまだ完全にはコピーされていません。このメッセージを返すのは、1 次サーバーだけです。
14009	サーバー	戻りコード	以前にサーバー切り替えは発生しませんでした。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby status switch'** 意味: サーバー間に切り替えプロセスが発生したことはありません。
14501	サーバー	エラー	操作が失敗しました。 意味: 操作が失敗し、サーバーはシャットダウンしようとしています。失敗の原因としては、非 HotStandby サーバーにコマンドを発行したか、コマンドを適用できない 1 次サーバーまたは 2 次サーバーにコマンドを発行したことが考えられます。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby switch primary'** • **ADMIN COMMAND 'hotstandby switch secondary'** • **ADMIN COMMAND 'hotstandby cominfo'** • **ADMIN COMMAND 'hotstandby connect'** • **ADMIN COMMAND 'hotstandby status switch'** • **ADMIN COMMAND 'hotstandby set standalone'** • **ADMIN COMMAND 'hotstandby copy'** • **ADMIN COMMAND 'hotstandby netcopy'**
14502	サーバー	エラー	RPC パラメーターが無効です。 意味: HSB 接続ストリング内で提供された接続情報の一部にエラーがあり、別のサーバーへの接続が失敗しました。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby status connect'**
14503	サーバー	エラー	通信エラー、接続の切断。 意味: 通信エラーがあったため、もう一方のサーバーを検出できませんでした。もう一方のサーバーへの接続に失敗しました。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby status connect'**
14520	サーバー	エラー	サーバーは HotStandby 2 次サーバーであり、接続は許可されません。

表 30. *HotStandby の solidDB*サーバー・エラー *(続き)*

コード	クラス	タイプ	説明
14522	サーバー	エラー	HotStandby コピー・ディレクトリーが指定されませんでした。 意味: コピー・ディレクトリーが指定されていません。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby copy'** この問題を解決するには、ディレクトリーをコマンドの一部として指定します。以下に例を示します。 `ADMIN COMMAND 'hotstandby copy ¥Secondary¥dbfiles¥'` あるいは、`solid.ini` 構成ファイル内で **CopyDirectory** パラメーターを設定します。
14523	サーバー	エラー	切り替えプロセスは、既にアクティブです。 意味: 切り替えプロセスは、HotStandby サーバー内で既にアクティブです。現行の切り替えを完了する必要があるだけなら、待ってください。2 回目の切り替え (つまり、元の構成に戻すこと) を実行しようとしている場合は、最初の切り替えが完了するまで待ってから、2 回目の切り替えを開始できます。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby switch primary'** • **ADMIN COMMAND 'hotstandby switch secondary'** • **ADMIN COMMAND 'hotstandby status switch'**
14524	サーバー	エラー	HotStandby データベースの基本データベースの 1 つが異なっており、データベースのタイム・スタンプが互いに異なっています。 意味: データベースが、異なるシード・データベースに由来しています。データベースの同期をとる必要があります。1 次サーバーのデータベースを 2 次サーバーにネットコピーしなければならない可能性があります。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby connect'** • **ADMIN COMMAND 'hotstandby status switch'**
14525	サーバー	エラー	HotStandby のデータベース同士が正しく同期していません。 意味: データベース同士が正しく同期していません。データベースの同期をとる必要があります。データベース・サーバーの 1 つ (2 次サーバーにしようとしている方) を、コマンド行パラメーター **-x backupserver** を指定して始動し、1 次サーバーのデータベースを 2 次サーバーにネットコピーする必要があります。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby connect'** • **ADMIN COMMAND 'hotstandby status switch'**

表 30. HotStandby の solidDBサーバー・エラー (続き)

コード	クラス	タイプ	説明
14526	サーバー	エラー	引数が無効です。 意味: HotStandby ADMIN COMMAND で使用した引数が、不明または無効です。 すべての HotStandby コマンドが、ADMIN COMMAND の結果セット内にこのエラーを返す可能性があります。 注: 以下の HotStandby コマンドでは、指定された 1 次サーバーまたは 2 次サーバーを切り替えに適用できない場合、この引数無効エラーは構文エラーです。 • **ADMIN COMMAND 'hotstandby switch primary'** • **ADMIN COMMAND 'hotstandby switch secondary'**
14527	サーバー	エラー	これは HotStandby サーバーではありません。 意味: HotStandby 用に構成されていないサーバーに対してコマンドが実行されました。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby connect'** • **ADMIN COMMAND 'hotstandby status switch'** • **ADMIN COMMAND 'hotstandby switch primary'** • **ADMIN COMMAND 'hotstandby switch secondary'** • **ADMIN COMMAND 'hotstandby state'**
14528	サーバー	エラー	両方の HotStandby データベースが 1 次データベースです。 意味: 両方のデータベースが 1 次です。これは、変更内容が競合する可能性があるので致命的エラーです。両方のデータベースは、システムによって自動的に 2 次状態へ落とされます。どのデータベースが真の 1 次データベースであるかを決定し、データベース同士の同期をとる必要があります。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby connect'** • **ADMIN COMMAND 'hotstandby status switch'**
14535	サーバー	エラー	サーバーは、既に 1 次サーバーです。 意味: 1 次サーバーに切り替えようとしているサーバーは、既にいずれかの PRIMARY 状態にあります。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby switch primary'**
14536	サーバー	エラー	サーバーは、既に 2 次サーバーです。 意味: 2 次サーバーに切り替えようとしているサーバーは、既にいずれかの SECONDARY 状態にあります。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby switch secondary'**

表 30. *HotStandby の solidDB*サーバー・エラー *(続き)*

コード	クラス	タイプ	説明
14537	サーバー	エラー	HotStandby 接続が切断されています。 意味: このコマンドは、1 次と 2 次の両方のサーバーから返されます。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby status connect'** • **ADMIN COMMAND 'hotstandby connect'** この問題の原因として考えられることの 1 つは、2 次サーバーの `solid.ini` ファイル内で Connect ストリングが正しくないことです。ネットコピー操作が成功するのに、接続コマンドが失敗する場合は、Connect ストリングを調べてください。ネットコピーでは、2 次サーバーが 1 次サーバーへの別個の接続を開く必要がないので、2 次サーバー上の Connect ストリングに誤りがあっても、成功する場合があります。
14538	サーバー	エラー	サーバーが HotStandby 1 次サーバーではありません。 意味: このコマンドを発行するには、サーバーが HotStandby 1 次サーバーであることが必要です。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hotstandby copy copy_directory'** • **ADMIN COMMAND 'hotstandby netcopy'** • **ADMIN COMMAND 'hotstandby connect'** • **ADMIN COMMAND 'hotstandby set primary alone'** • **ADMIN COMMAND 'hotstandby set standalone'**

表 30. HotStandby の solidDBサーバー・エラー (続き)

コード	クラス	タイプ	説明
14539	サーバー	エラー	操作が拒否されました。 このエラー・コードは、以下のいずれかの状態が発生したときに発生します。 • ユーザーが 1 次サーバーに対して **netcopy** コマンドを発行しましたが、2 次サーバーであるはずのサーバーが実際には 2 次サーバー状態でないか、「ネットコピー listen モード」になっていません。(1 次サーバーと「2 次」サーバーの両方が PRIMARY ALONE 状態の可能性があります。) この問題を解決するには、**-x backupserver** コマンド行オプションを指定して「2 次サーバー」を再始動し、再度 1 次サーバーに対して **netcopy** コマンドを発行してみます。 **重要:** 両方のサーバーが PRIMARY ALONE 状態だった場合、しかも、両方のサーバーが PRIMARY ALONE 状態だったときにトランザクションを実行していた場合は、それぞれに相手側が持っていないデータが存在する可能性があります。これは深刻なエラーであり、**netcopy** を実行して両方を同期状態に戻すと、既に「2 次」サーバーにコミットされているトランザクションが上書きされる結果になります。 • このメッセージは、コールバック関数を使用し、コールバック関数がシャットダウンを拒否するか、バックアップ・コマンドまたはネットコピー・コマンドの受け入れを拒否したときに生成される場合があります。 リンク・ライブラリー・アクセスを使用している場合は、SSCSetNotifier 関数を使用して「コールバック」関数を指定できます。コールバック関数は、サーバーがシャットダウンまたはネットコピー操作の実行を命令したときに、通知を受け取ります。何らかの理由でアプリケーションがコマンドに従わない場合、コールバックはコマンドをキャンセルする値を返すことがあります。その状態では、エラー 14539 が表示されます。 問題を解決するには、クライアント・コードが中断を望まない操作を終了するまで待ってから、コマンド (例えば、シャットダウンまたはネットコピー) を再試行します。
14540	サーバー	エラー	サーバーは、既に HotStandby サーバーではありません。
14541	サーバー	エラー	`solid.ini` 内の HotStandby 構成が **ADMIN COMMAND 'HSB SET STANDALONE'** と競合しています。
14542	サーバー	エラー	サーバーが backupserver モードです。操作が拒否されました。
14543	サーバー	エラー	コマンドが無効です。データベースは HotStandby データベースですが、`solid.ini` 構成ファイル内に HotStandby セクションが見つかりませんでした。
14544	サーバー	エラー	操作が失敗しました。このコマンドは、ディスクレス・サーバーではサポートされていません。
14545	サーバー	エラー	1 次サーバーは、そのロールが primary broken の場合、PRIMARY ALONE にのみ設定できます。

表 30. HotStandby の solidDBサーバー・エラー (続き)

コード	クラス	タイプ	説明
14546	サーバー	エラー	切り替えが失敗しました。サーバーまたはリモート・サーバーを PRIMARY ALONE から 2 次サーバーに切り替えることができません。切り替えの前に、まずキャッチアップを実行してください。 意味: このコマンドは、PRIMARY ALONE 状態のローカルまたはリモート 1 次サーバーから SECONDARY への状態切り替えが実行され、1 次サーバーと 2 次サーバーが同期していないことが検出された場合に返されます。1 次サーバーを 2 次サーバーに接続し、キャッチアップ・プロセスが完了するのを待ってから、2 次サーバーを 1 次サーバーに切り替える必要があります。 このエラーを返す HotStandby コマンド: • **ADMIN COMMAND 'hotstandby switch secondary'**
14547	サーバー	エラー	**-R** オプション (読み取りタイムアウト) の値がなかったか、無効でした。
14548	サーバー	エラー	切り替えが失敗しました。STANDALONE のサーバーを SECONDARY に切り替えることができませんでした。 意味: このコマンドは、STANDALONE 状態のローカルまたはリモート 1 次サーバーから SECONDARY への状態切り替えが実行され、1 次サーバーと 2 次サーバーが同期していないことが検出された場合に返されます。1 次サーバーを 2 次サーバーに接続し、キャッチアップが完了するのを待ってから、2 次サーバーを 1 次サーバーに切り替える必要があります。 このエラーを返す HotStandby コマンド: • **ADMIN COMMAND 'hotstandby switch secondary'**
14549	サーバー	エラー	HotStandby トランザクションがアクティブです。 意味: HotStandby 接続が切断されている場合、1 次サーバーはシャットダウンの前に、ALONE モードに設定されるか、SECONDARY モードに切り替えられている必要があります。
14550	サーバー	エラー	Hotstandby 接続パラメーターを変更できるのは、1 次サーバーが 2 次サーバーに接続されていないときだけです。
14551	サーバー	エラー	START AFTER COMMIT ステートメントの最大数に到達しました。
14552	サーバー	エラー	サーバーはバックアップ・サーバー・モードにあり、接続は許されません。 エラー 14552 は、クライアントが、バックアップ・サーバー・モード (ネットコピー *listen* モード とも呼ばれます) にある solidDB サーバーへの接続を確立しようとしたときに返されます。バックアップ・サーバー・モードは、solidDB インスタンスがコマンド行オプション **-xbackupserver** を使用して始動された場合の特殊なサーバー・モードです。このモードは、solidDB インスタンスが 2 次サーバーで、そのサーバーは、1 次サーバーで発行された **netcopy** コマンドのため、1 次サーバーからのデータベース・ファイルを待機中であるか、その受信処理の途中であることを示します。

solidDB HotStandby エラー

表 31. solidDB HotStandby エラー

コード	クラス	タイプ	説明
14700	HotStandby	エラー	接続が拒否されました。両方のサーバーが PRIMARY ロールです。 意味: コマンド **'hsb connect'** は、両方のノードのロールが同じ場合に、このエラーを返します。

表 31. solidDB HotStandby エラー (続き)

コード	クラス	タイプ	説明
14701	HotStandby	エラー	接続が拒否されました。両方のサーバーが SECONDARY ロールです。 意味: コマンド **'hsb connect'** は、両方のノードのロールが同じ場合に、このエラーを返します。
14702	HotStandby	エラー	操作が失敗しました。キャッチアップがアクティブです。 意味: サーバーがキャッチアップを実行中に、1 次サーバー上で以下のいずれかのコマンドを発行すると、このエラーが発生します。**'hsb switch secondary'**、**'hsb set secondary alone'**、**'hsb set standalone'**、**'hsb connect'**、**'hsb copy'**、または **'hsb netcopy'**。 サーバーがキャッチアップを実行中に、2 次サーバー上で以下のいずれかのコマンドを発行すると、このエラーが発生します。**'hsb switch primary'**、**'hsb set secondary alone'**、**'hsb set primary alone'**、**'hsb set standalone'**、または **'hsb connect'**。
14703	HotStandby	エラー	操作が失敗しました。コピーがアクティブです。 意味: 1 次サーバーがコピーまたはネットコピーを実行中に、以下のコマンドがこのエラーを返します。**'hsb switch secondary'**、**'hsb set secondary alone'**、**'hsb set standalone'**、**'hsb connect'**、**'hsb disconnect'**、**'hsb copy'**、または **'hsb netcopy'**。
14704	HotStandby	エラー	HotStandby のコピーまたはネットコピーは、1 次サーバーが PRIMARY ALONE 状態であるときにのみ許可されます。 意味: このエラーは、サーバーが PRIMARY ACTIVE 状態にあり、コマンド **'hsb copy'** または **'hsb netcopy'** が発行された場合に返されます。
14705	HotStandby	エラー	この状態で STANDALONE に設定することは許されません。 意味: サーバーが PRIMARY ACTIVE 状態にあるときに **'hsb set standalone'** コマンドを発行すると、このメッセージを受け取ります。
14706	HotStandby	エラー	HotStandby に無効な読み取りスレッド・モードです。モード 2 のみがサポートされています。
14707	HotStandby	エラー	STANDALONE 状態で許可されない操作。
14708	HotStandby	エラー	キャッチアップが失敗しました。キャッチアップ位置をログ・ファイルから検出できませんでした。
14709	HotStandby	エラー	HotStandby は使用可能に設定されていますが、接続ストリングが定義されていません。
14710	HotStandby	エラー	HotStandby 管理コマンドが着信管理コマンドと競合しています。
14711	HotStandby	エラー	サーバーがシャットダウン操作の途中であるために失敗しました。
14712	HotStandby	エラー	サーバーが 2 次サーバーです。この操作には 1 次サーバーを使用してください。

solidDB HSB エラーおよびメッセージ

表 32. solidDB HSB エラーおよびメッセージ

コード	クラス	タイプ	説明
14007	HSB	メッセージ	CONNECTING
14008	HSB	メッセージ	CATCHUP
14009	HSB	メッセージ	サーバーの始動以来、ロールの切り替えがありません。
14010	HSB	メッセージ	DISCONNECTING
14522	HSB	メッセージ	HotStandby コピー・ディレクトリーが指定されませんでした。

表 32. *solidDB HSB* エラーおよびメッセージ *(続き)*

コード	クラス	タイプ	説明
14537	HSB	メッセージ	BROKEN
14704	HSB	エラー	HotStandby のコピーまたはネットコピーは、1 次サーバーが PRIMARY ALONE 状態であるときにのみ許可されます。
14712	HSB	エラー	サーバーが 2 次サーバーです。この操作には 1 次サーバーを使用してください。
30500	HSB	メッセージ	HotStandby 1 次サーバーとして始動しました。
30501	HSB	メッセージ	HotStandby 2 次サーバーとして始動しました。
30502	HSB	メッセージ	データベースは、前回 HotStandby 2 次サーバーとして始動して使用されたとき、正しくシャットダウンされませんでした。
30503	HSB	メッセージ	HotStandby 1 次サーバーを 2 次サーバーとして強制的に始動します。
30504	HSB	メッセージ	HotStandby ロールが 2 次サーバーに切り替えられました。
30505	HSB	メッセージ	HotStandby ロールが 1 次サーバーに切り替えられました。
30506	HSB	メッセージ	1 次サーバーを、PRIMARY ALONE に設定するか 2 次サーバー・ロールに切り替える必要があります。
30507	HSB	メッセージ	HotStandby サーバーが PRIMARY ALONE に設定されました。
30508	HSB	メッセージ	HotStandby サーバーが SECONDARY ALONE に設定されました。
30509	HSB	メッセージ	1 次サーバーへの HotStandby 切り替えが失敗しました。エラー *error_code*
30510	HSB	メッセージ	2 次サーバーへの HotStandby 切り替えが失敗しました。エラー *error_code*
30511	HSB	メッセージ	HotStandby を *server_name* に対して始動できませんでした。エラー *error_code*
30512	HSB	メッセージ	HotStandby ロールの 1 次サーバーへの切り替えが失敗しました。エラー *error_code*
30513	HSB	メッセージ	HotStandby ロールの 2 次サーバーへの切り替えが失敗しました。エラー *error_code*
30514	HSB	メッセージ	両方のデータベースが、2 次サーバーとして始動する 1 次サーバーです。
30515	HSB	メッセージ	両方の HotStandby データベースが 1 次データベースです。
30516	HSB	メッセージ	HotStandby を *server_name* に対して始動できませんでした。もう一方のサーバーはエラー *error_code* で拒否しました。
30517	HSB	メッセージ	2 次サーバー内の HotStandby ロールが切り替えられました。
30518	HSB	メッセージ	HotStandby ロールがスタンドアロンに切り替えられました。
30530	HSB	メッセージ	2 次サーバーへの HotStandby キャッチアップ・データの送信を開始中です。
30531	HSB	メッセージ	HotStandby キャッチアップが正常に完了しました。
30532	HSB	メッセージ	HotStandby キャッチアップが異常終了しました。
30533	HSB	メッセージ	HotStandby キャッチアップを開始できません。2 次サーバーは 1 次サーバーと正しく同期しておらず、完全同期が必要です。
30534	HSB	メッセージ	HotStandby キャッチアップが異常終了しました。状況 *error_code*
30535	HSB	メッセージ	HotStandby キャッチアップが異常終了しました。エラー *error_code*
30536	HSB	メッセージ	HotStandby キャッチアップが通信エラーのために異常終了しました。
30537	HSB	メッセージ	HotStandby キャッチアップが異常終了し、2 次サーバーがエラー *error_code* を返しました。
30538	HSB	メッセージ	HotStandby キャッチアップ・サイズ <value> が、構成された最大サイズ *value* を超えています。HotStandby を停止中です。
30539	HSB	メッセージ	HotStandby キャッチアップでのファイル・エラー。HotStandby を停止中です。
30540	HSB	メッセージ	1 次サーバーから HotStandby キャッチアップ・データの受信を開始中です。
30541	HSB	メッセージ	2 次サーバーは、ログ・ファイルの破損のため、1 次サーバーと正しく同期していません。2 次サーバーを再始動し、HSB netcopy を実行してください。
30550	HSB	メッセージ	HotStandby 2 次サーバーへの接続の破損
30551	HSB	メッセージ	HotStandby に接続しました。
30552	HSB	メッセージ	HotStandby 2 次サーバーに接続しました。
30553	HSB	メッセージ	HotStandby 1 次サーバーに接続しました。

表 32. *solidDB HSB* エラーおよびメッセージ *(続き)*

コード	クラス	タイプ	説明
30554	HSB	メッセージ	2 次サーバーへの HotStandby 接続が破損し、オープン・トランザクションがオペレーターによるトランザクション状況の解決を待っています。1 次サーバーを ALONE モードに設定するか、SECONDARY モードに切り替える必要があります。
30555	HSB	メッセージ	HotStandby ping タイムアウト
30556	HSB	メッセージ	HotStandby 2 次サーバーへの接続の破損
30557	HSB	メッセージ	HotStandby のデータベース同士が正しく同期していません。
30558	HSB	メッセージ	2 次サーバーへの HotStandby 接続がタイムアウトになりました。
30559	HSB	メッセージ	HotStandby 接続が切断されています。
30560	HSB	メッセージ	HotStandby: *HotStandby_error_message*
30561	HSB	メッセージ	HotStandby への接続が開始されました。
30562	HSB	メッセージ	HotStandby 1 次サーバーへの接続の破損
30570	HSB	メッセージ	ネットワーク・バックアップが完了しました。
30571	HSB	メッセージ	ネットワーク・バックアップの受信が開始されました。
30572	HSB	メッセージ	HotStandby コピー/ネットコピーを使用してデータベースを開始しました。
30573	HSB	メッセージ	ネットワーク・バックアップが失敗しました。
30574	HSB	メッセージ	HotStandby は強制的にスレッドを 1 にします。
30575	HSB	メッセージ	HotStandby レプリケーションが構成されましたが、アクティブ・ライセンスがないため、レプリケーションは開始されませんでした。
30577	HSB	メッセージ	HotStandby 接続操作が失敗しました。
30579	HSB	メッセージ	HotStandby 接続は、既にアクティブです。
30581	HSB	メッセージ	無効なイベント *event*
30582	HSB	メッセージ	HotStandby がサーバーを PRIMARY ALONE に設定できません。
30583	HSB	メッセージ	HotStandby コピーが失敗しました。
30585	HSB	メッセージ	データベースはネットコピー用の listen を開始します。
30586	HSB	メッセージ	HotStandby キャッチアップ
30750	HSB	メッセージ	HotStandby 接続は、既にアクティブです。
30752	HSB	メッセージ	操作が失敗しました。切断がアクティブです。
30757	HSB	メッセージ	CONNECTED
30758	HSB	メッセージ	正しくない HotStandby コマンドです。
30759	HSB	メッセージ	HotStandby サーバーが STANDALONE に設定されました。
30760	HSB	メッセージ	サーバー同士の切断プロセスが開始されました。
30761	HSB	メッセージ	ロールを 1 次サーバーに切り替えるプロセスが開始されました。
30762	HSB	メッセージ	ロールを 2 次サーバーに切り替えるプロセスが開始されました。
30763	HSB	メッセージ	サーバー同士の接続プロセスが開始されました。
30764	HSB	メッセージ	コピーが開始されました。
30765	HSB	メッセージ	パラメーター **AutoPrimaryAlone** は Yes に設定されます。
30766	HSB	メッセージ	パラメーター **AutoPrimaryAlone** は No に設定されます。
30767	HSB	メッセージ	パラメーター **Connect** は *value* に設定されます。
30768	HSB	メッセージ	HotStandby 接続は、既に切断されています。
30769	HSB	メッセージ	サーバー間の接続がアクティブなので、操作が失敗しました。
30772	HSB	メッセージ	HotStandby ノード ID を ini ファイル内で定義する必要があります。
30774	HSB	メッセージ	サーバーは、既に STANDALONE です。
30775	HSB	メッセージ	パラメーター **CopyDirectory** は *value* に設定されます。
30776	HSB	メッセージ	パラメーター **ConnectTimeout** は *value* に設定されます。
30777	HSB	メッセージ	パラメーター **PingTimeout** は *value* ミリ秒に設定されます。

表 32. solidDB HSB エラーおよびメッセージ (続き)

コード	クラス	タイプ	説明
30779	HSB	メッセージ	HotStandby マイグレーションがアクティブです。
30782	HSB	メッセージ	サーバーは、既に PRIMARY ALONE に設定されています。
30783	HSB	メッセージ	サーバーは、既に SECONDARY ALONE に設定されています。
30784	HSB	メッセージ	パラメーター *parameter_name* は *value* に設定されます。
30785	HSB	メッセージ	パラメーター *parameter_name* は *value* に設定されます。
30786	HSB	メッセージ	パラメーター *parameter_name* は *value* に設定されます。
30787	HSB	致命的エラー	pri_dologskip:bad type, log pos, log size このエラーは、HSB 1 次サーバーで失敗した操作を指しています。このエラーは、失敗した操作とそのログ内の位置、およびログ・サイズを返します。レプリケーション・ログ内の操作はスキップされます。
30788	HSB	致命的エラー	pri_hsblogcopy_write:bad type, log pos, log size このエラーは、HSB 1 次サーバーで失敗した操作を指しています。レプリケーション・ログ・ファイルへの書き込みは失敗します。このエラーは、失敗した操作とそのログ内の位置、およびログ・サイズを返します。
30789	HSB	致命的エラー	ホット・スタンバイ・レプリケーション・ログ・ファイルを開くことができませんでした。
30790	HSB	致命的エラー	HotStandby ログ用のメモリーを割り振ることができませんでした。最大ログ・サイズは *logsize* です。 このエラーは、HotStandby を使用するディスクレス・データベースに関するものです。それらのシステムでは、HotStandby ログはメモリーに書き込まれます。このエラーは、ログ・ファイルに、より多くのメモリーを割り振ろうとして失敗した場合に発行されます。
30791	HSB	致命的エラー	HotStandby:solhsby:不良タイプ *type*、ログ位置 *log_pos*、ログ・サイズ *log_size*
30792	HSB	メッセージ	両方のサーバーが 2 次サーバーです。

B.2 高可用性コントローラーのエラー・コードと状況コード

solidDB 高可用性コントローラー・エラー (17xxx) は、特定の高可用性コントローラー・コマンドを使用しているときに発生します。

表 33. 高可用性コントローラーのエラー・コードと状況コード

エラーまたは状況コード	説明
17501	HAC はシャットダウン操作の途中です。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac shutdown'** 意味: 高可用性コントローラーがシャットダウン操作の途中です。

表 33. 高可用性コントローラーのエラー・コードと状況コード (続き)

エラーまたは状況コード	説明
17502	コマンドが失敗しました。HAC は中断状態です。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac suspend'** • **ADMIN COMMAND 'hac gethsbdstate'** • **ADMIN COMMAND 'hac getdbstate'** • **ADMIN COMMAND 'hac shutdowndb'** • **ADMIN COMMAND 'hac restartdb'** • **ADMIN COMMAND 'hac switchdb'** • **ADMIN COMMAND 'hac statemachinestate'** • **ADMIN COMMAND 'hac getereip'** • **ADMIN COMMAND 'hac pingere'** 意味: コマンドの実行に失敗し、高可用性コントローラーは中断状態です。
17503	無効なコマンド。'hac commands' を入力してヘルプを表示してください。 意味: 正しくない admin command 構文です。
17504	HAC は、既に稼働しています。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac resume'**
17506	この HSB 状態では、切り替えは許可されません。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac switchdb'**
17507	コマンドを実行できません。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac shutdowndb'**
17509	データベース・サーバーの再始動が失敗しました。詳細については、`solmsg.out` を参照してください。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac restartdb'**

表 33. 高可用性コントローラーのエラー・コードと状況コード (続き)

エラーまたは状況コード	説明
17510	データベース・サーバーに接続できません。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac switchdb'**
17511	データベース・サーバーがシャットダウンされませんでした。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac restartdb'**
17513	切り替えが失敗しました。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac switchdb'**
17514	ERE IP が構成ファイル内で指定されていません。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac getereip'** • **ADMIN COMMAND 'hac pingere'**
17516	HAC は、既にアクティブです。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac setactive'** • **ADMIN COMMAND 'hac pingere'**
17517	HAC は、既にパッシブです。 コマンドの結果セット内にこの状況を返すことがある ADMIN COMMAND: • **ADMIN COMMAND 'hac setpassive'**

B.3 HotStandby の solidDB データベース・エラー

solidDB データベース・エラー (10002 - 10050) は solidDB によって検出され、クライアント・アプリケーションへ送信されます。これらのエラーには、管理アクションを必要とするものもあります。

このセクションでは、HotStandby に関連する solidDB データベース・エラーのリストを示します。データベース・クラスでのエラーの完全なリストは、「*IBM solidDB 管理者ガイド*」の『*solidDB データベース・エラー*』のセクションに記載されています。

表 34. solidDB データベース・エラー

コード	クラス	タイプ	説明
10002	データベース	エラー	操作が失敗しました。 意味: 予期しなかったエラーで接続操作が失敗しました。最も考えられることは、サーバー同士が正しく同期していないことです。
10013	データベース	エラー	トランザクションが読み取り専用です。 意味: 読み取り専用に設定されたトランザクションの内部で書き込みを試みたか、サーバーが (例えば状態切り替え中で) 一時的に読み取り専用モードに設定されています。更新可能なトランザクションは許可されません。
10019	データベース	エラー	バックアップは、既にアクティブです。 意味: 既にバックアップまたはコピーが進行中であるときに、それを開始しようとしました。
10024	データベース	エラー	バックアップ・ディレクトリー「*directory_name*」が正しくありません。 意味: バックアップまたはコピー・ディレクトリーが空ストリングであるか、ドット (現行ディレクトリーにバックアップまたはコピーが作成されることを示します) です。
10030	データベース	エラー	バックアップまたはコピー・ディレクトリー「*directory_name*」は存在しません。 意味: バックアップまたはコピー・ディレクトリーが見つかりません。バックアップまたはコピー・ディレクトリーの名前を確認してください。
10045	データベース	エラー	この操作を HotStandby 2 次サーバーに対して実行することはできません。 意味: この操作を HotStandby 2 次サーバーに対して実行することはできません。 要求した操作を成功させるには、サーバーが 1 次サーバーでなければなりません。
10046	データベース	エラー	操作が失敗しました。データ・ディクショナリー操作がアクティブです。 意味: データ・ディクショナリー操作が現在進行中です。
10047	データベース	エラー	複製されたトランザクションが異常終了しました。 意味: トランザクションが、例えば状態切り替えなどで、異常終了しました。サーバー状態が 1 次サーバーから 2 次サーバーへ切り替えられるとき、すべてのアクティブ・トランザクションは異常終了します。

表 34. *solidDB* データベース・エラー *(続き)*

コード	クラス	タイプ	説明
10048	データベース	エラー	複製されたトランザクションにデータ・ディクショナリー変更が含まれており、通常の更新操作は許可されません。 意味: HotStandby モードでは、データ・ディクショナリー操作は制限されています。例えば、CREATE TABLE を通常の更新操作と混在させることはできません。 このメッセージは、バージョン 4.1 以降では廃止されました。バージョン 4.1 以降では、HSB を使用している間、トランザクション内に DML 操作と DDL 操作を混在させることができます。
10049	データベース	エラー	リモート・サーバーが 2 次サーバーではありません。 意味: コマンド内で指定したサーバーは、SECONDARY 状態ではありません。
10050	データベース	エラー	複製された操作で BLOB 列を更新しました。 意味: BLOB 列を 2 次サーバーへ複製することはできません。
10078	データベース	エラー	ユーザーがトランザクションをロールバックしました。
10079	データベース	エラー	filespec を除去できません。ファイルは、既に使用中です。
10080	データベース	エラー	HotStandby 2 次サーバーが、1 次サーバーから受け取った操作を実行できません。 意味: このエラーの原因として考えられることは、データベースが HotStandby の **copy** または **netcopy** コマンドで 1 次サーバーから作成されなかったことです。
10081	データベース	エラー	データベース・ファイルが不完全であるか破損しています。 意味: ファイルがホット・スタンバイの 2 次サーバー上にある場合は、**hotstandby copy** または **hotstandby netcopy** コマンドを使用してファイルを 1 次サーバーから再度送信してください。
10082	データベース	エラー	バックアップが異常終了しました。
10083	データベース	エラー	既に 2 次サーバーへコミットが送信されているため、HSB トランザクションを中止することができませんでした。
10084	データベース	エラー	表がロックされていません。
10085	データベース	エラー	チェックポイントは使用不可です。
10087	データベース	エラー	HotStandby はメイン・メモリー表に対しては許可されません。
10088	データベース	エラー	指定したロック・タイムアウト値が大きすぎます。
10089	データベース	エラー	操作が失敗しました。サーバーは HSB PRIMARY UNCERTAIN モードです。

B.4 solidDB 表エラー

solidDB データベース表エラーは、エラーのある SQL ステートメントが原因で発生し、solidDB によって検出されます。管理アクションは必要ありません。

このセクションでは、HotStandby に関連する solidDB 表エラーのリストを示します。表クラスでのエラーの完全なリストは、「*IBM solidDB 管理者ガイド*」の『*solidDB表エラー*』のセクションに記載されています。

表 35. solidDB 表エラー

コード	クラス	タイプ	説明
13068	表	エラー	サーバーのシャットダウンが進行中です。 意味: サーバーのシャットダウンが進行中のため、この操作を完了することはできません。
13123	表	エラー	表「*table_name*」が空ではありません。 意味: この操作は、表が空の場合にのみ実行できます。例えば、ディスク・ベースからインメモリー (またはその逆) への表の変更は、表が空のときにのみ行うことができます。
13167	表	エラー	インメモリー表のみをトランジエント表とすることができます。 意味: ディスク・ベースのトランジエント表を作成することはできません。例えば、以下の SQL ステートメントは、このエラー・メッセージを生成します。 `CREATE TRANSIENT TABLE t1 (i INT) STORE DISK;`
13170	表	エラー	インメモリー表のみをテンポラリー表とすることができます。 意味: ディスク・ベースのテンポラリー表を作成することはできません。例えば、以下の SQL ステートメントは、このエラー・メッセージを生成します。 `CREATE TEMPORARY TABLE t1 (i INT) STORE DISK;`

B.5 solidDB 通信エラー

solidDB 通信エラー (21306、21308) は、ネットワーク・エラーが原因で発生します。これらのエラーは、管理アクションを必要とします。

このセクションでは、HotStandby に関連する solidDB 通信エラーのリストを示します。サーバー・クラスでのエラーの完全なリストは、「*IBM solidDB 管理者ガイド*」の『*solidDB 通信エラー*』のセクションに記載されています。

表 36. solidDB 通信エラー

コード	クラス	タイプ	説明
21306	通信	エラー	サーバー「*server_name*」が見つからず、接続が失敗しました。 意味: 2 次サーバーが見つかりませんでした。 • サーバーが稼働しているかどうかを確認してください。 • ネットワーク名が有効かどうかを確認してください。 • サーバーが指定されたネットワーク名を listen しているかどうかを確認してください。

表 36. solidDB 通信エラー (続き)

コード	クラス	タイプ	説明
21308	通信	エラー	接続が切断されています (<operation> 操作が、コード <code> で失敗しました)。 例: 「Connection is broken (TCP/IP 'Write' operation failed with code 7).」 推奨されるアクション: • サーバーが稼働しているかどうかを確認してください。 • ネットワーク名が有効かどうかを確認してください。 • サーバーが指定されたネットワーク名を listen しているかどうかを確認してください。

付録 C. HotStandby と HAC ADMIN COMMAND

このセクションでは、HotStandby で使用可能な ADMIN COMMAND を説明します。

使用しているツールによって、ADMIN COMMAND 構文は以下のように異なります。

- **solidDB SQL エディター (solsql)**

  solsql で使用する場合、コマンド名は引用符とともに指定する必要があります。以下に例を示します。

  ```
  ADMIN COMMAND 'hotstandby switch primary';
  ADMIN COMMAND 'hacontroller shutdown';
  ```

- **solidDB リモート制御 (solcon)**

  solcon で使用する場合は、コマンド名は ADMIN COMMAND の接頭部と引用符を使用せずに指定する必要があります。以下に例を示します。

  ```
  hotstandby switch primary
  hacontroller shutdown
  ```

ヒント: 「hotstandby」は「hsb」に、「hacontroller」は「hac」に省略することができます。以下に例を示します。

```
ADMIN COMMAND hsb switch primary
ADMIN COMMAND hac shutdown
```

構文記述では、考えられる最も短い形式が使用されています。つまり、「ADMIN COMMAND」と引用符が省略され、省略形の「hsb」および「hac」が使用されています。

注: ADMIN COMMAND は、コマンドに構文エラーがなければ必ず、成功を示す戻りコード (0) を返します。コマンドの実際の結果コードは、結果セットの「RC」フィールドに含まれます。

ADMIN COMMAND について詳しくは、「*IBM solidDB SQL ガイド*」のセクション『*ADMIN COMMAND*』を参照してください。

C.1 HotStandby コマンド (ADMIN COMMAND)

表 37. HotStandby コマンド (ADMIN COMMAND)

コマンド	説明
hsb cominfo	もう一方のサーバーとの接続に使用される接続ストリングを返します。これは通常は **HotStandby.Connect** パラメーターの値ですが、以下のコマンドで設定された可能性もあります。 `ADMIN COMMAND 'hsb parameter connect connect_string';` この情報は、他のサーバーに接続するためにアプリケーションで使用することができます。
hsb connect	1 次サーバーおよび 2 次サーバー間の接続が切れるか、またはまだ確立していない場合、このコマンドは 1 次サーバーを 2 次サーバーに接続して、HotStandby レプリケーションを開始します。サーバー間には自動接続メカニズムがないため、このコマンドは、サーバー間の接続に常に必要です。正常に接続した後、1 次サーバーの状態は自動的に PRIMARY ALONE から PRIMARY ACTIVE に設定されます。正常に接続できなかった場合、状態は PRIMARY ALONE のままです。 このコマンドは、1 次サーバーまたは 2 次サーバーのいずれでも実行できます。 **注:** このコマンドを実行したとき、1 次サーバーおよび 2 次サーバーが接続済みであるが、トランザクション・ログがまだ完全には 2 次側へコピーされていなければ、キャッチアップがアクティブであることを示すメッセージ「`Catchup is active`」が表示されます。

表 37. HotStandby コマンド (ADMIN COMMAND) (続き)

コマンド	説明
hsb copy [*directory_name*]	**重要:** このコマンドは推奨されません。代わりに **hsb netcopy** コマンドを使用してください。 **hsb copy** コマンドを使用して、1 次側から最初に 2 次データベースを作成できます。このコマンドは、データベースを 1 次ノードのローカル・ディレクトリーに (および 2 次ノードのローカル・ディレクトリーにも) コピーします。コピーの完了後、2 次サーバーを始動できます。1 次サーバーを 2 次サーバーに接続した後、1 次サーバーは、トランザクション・ログを 2 次サーバーにコピーすることによって、自動的に 2 次サーバーを最新の状態にします。 このコマンドを使用して、1 次データベースを、1 次ノードのローカル・ディレクトリーにある 2 次データベースと (それが相当の期間オフラインになっていたときに) 同期化することもできます。 56 ページの『3.4.5, 1 次サーバーと 2 次サーバーの同期』を参照してください。 オプションの *directory_name* を指定した場合、データベース・ファイルはそのディレクトリーにコピーされます。指定しなければ、solid.ini 構成ファイルの [Hotstandby] セクションにある **copydirectory** パラメーターで指定したディレクトリーにコピーされます。 **hsb copy** コマンドは solid.ini 構成ファイルもログ・ファイルもコピーしないため、このディレクトリーは通常のバックアップ・ディレクトリーとは異なるディレクトリーにすることをお勧めします。 1 次側は、PRIMARY ALONE 状態の場合にのみ **hsb copy** コマンドを実行できます。コマンドの実行中およびその後で、サーバーは PRIMARY ALONE 状態のままです。コマンドが完了した後で、2 次サーバーを始動して、2 つのサーバーを接続できます。
hsb disconnect	これは、HSB ペアのもう一方のメンバーから安全に切断するようサーバーに通知します。このコマンドは、1 次サーバーまたは 2 次サーバーのいずれでも有効です。このコマンドを使用する典型的な理由は、両方のサーバーのいずれかをアップグレードする前にそれらを切断することです。 (もう一方サーバーは、それが引き続きクライアント要求へ応答できるように PRIMARY ALONE 状態に設定できます。) このコマンドによって通常は、両方のサーバーが「単体 (Alone)」モードになります。つまり、1 次サーバーは PRIMARY ACTIVE から PRIMARY ALONE に切り替わり、2 次サーバーは SECONDARY ACTIVE から SECONDARY ALONE に切り替わります。 このコマンドは 1 次側および 2 次側の両方で有効です。 **注:** シャットダウン・コマンド ADMIN COMMAND 'shutdown' を使用すると、サーバーはシャットダウンする前に、制御された方法で切断を実行します。 2 次側がシャットダウン (および切断) した場合、1 次側は PRIMARY ALONE 状態になっても安全なことを認識してそれを実行します。

表 37. HotStandby コマンド (ADMIN COMMAND) (続き)

コマンド	説明

hsb logpos	このコマンドは、前回の操作時のログ操作 ID およびサーバーのロール (PRIMARY、SECONDARY、STANDALONE) を返します。
	
	標準的な出力を下に示してあります。
	
	`ADMIN COMMAND 'hsb logpos';`
	`RC TEXT`
	`-- ----`
	`0 00000000000000000871:PRIMARY`
	
	このコマンドを使用すると、例えば、両方のデータベースで障害が発生した後、どちらのサーバーを 1 次サーバーにすべきか分からない場合などに、2 つのサーバーのどちらを 1 次サーバーにするかを判別できます。 (必ずしも、接続が切断される前に 1 次サーバーであったサーバーが現在の 1 次サーバーにすべきサーバーであるというわけではありません。)
	
	原則として、ログ操作 ID の値が大きいサーバーがより多くのトランザクションを受け入れてきているため、これを 1 次サーバーにすべきです。しかし、HSB 接続で障害が発生した後に両方のサーバーを更新した場合には、ログ操作 ID の値を比較しても信頼性はありません。
	
	このコマンドの使用方法の詳細については、 71 ページの『3.4.9, どちらのサーバーを 1 次サーバーにするかの選択』を参照してください。

hsb netcopy	このコマンドは、1 次データベースまたはディスクレス・インメモリー・データを 2 次サーバーへコピーするために使用できます。データベース・ファイルは、1 次サーバーの `solid.ini` ファイルの **HotStandby.Connect** パラメーターで定義された接続を使用して、ネットワーク・リンクを介してコピーされます。 このコマンドは、例えば、1 次データベースを長時間オフラインになっていた 2 次データベースと同期する場合に使用することができます。 **hsb netcopy** を使用して新しい 2 次データベースを作成することもできます。例えば、破損した 2 次データベースを置き換えたり、新しい HotStandby 構成用の 2 次データベースをセットアップしたり、既存の構成に新しい 2 次データベースを追加したりする場合などです。 このコマンドを発行するには、1 次サーバーが PRIMARY ALONE 状態でなければなりません。 コマンドの完了後 (正常でも、そうでなくても)、1 次サーバーは PRIMARY ALONE 状態のままです。コピーが正常に完了した場合、2 次サーバーは自動的に SECONDARY ALONE 状態に切り替わります。 **ヒント: hsb netcopy** コマンドの後には通常、1 次サーバーおよび 2 次サーバーを接続するために **hsb connect** コマンドが続きます。2 次サーバーに接続した後、1 次サーバーは、トランザクション・ログをコピーすることによって、自動的に 2 次サーバーを最新の状態にします。 **hsb netcopy** の使用についての詳細は、 56 ページの『3.4.5, 1 次サーバーと 2 次サーバーの同期』および 60 ページの『ネットワークを介した 1 次データベースの 2 次側へのコピー』を参照してください。

表 37. *HotStandby* コマンド *(ADMIN COMMAND) (続き)*

コマンド	説明

hsb parameter	**重要:** このコマンドは推奨されません。
	
	このコマンドを使用すると、HSB 固有のパラメーター、例えば、**AutoPrimaryAlone**、**Connect**、および **PingTimeout** などを設定できます。それぞれのパラメーターの詳細については、127 ページの『付録 A. HotStandby 構成パラメーター』を参照してください。
	
	いくつのパラメーターの 1 つの値を設定すると、そのコマンドはすぐに有効になりますが、solid.ini 構成ファイルには、シャットダウンが実行されるまで書き込まれません。
	
	このコマンドの構文は以下のとおりです。
	
	`ADMIN COMMAND 'hsb parameter`
	`*param_name param_value*';`
	
	このコマンドでは等号記号を使用しません。したがって、以下のような他の類似の (推奨される) コマンドとは異なります。
	
	`ADMIN COMMAND 'parameter`
	`hotstandby.*param_name* = *param_value*';`

表 37. HotStandby コマンド (ADMIN COMMAND) (続き)

コマンド	説明
hsb role	**重要:** このコマンドは推奨されません。代わりに **hsb state** コマンドを使用してください。 以下のロールのいずれかを結果セットで返します。 • PRIMARY (接続済みサーバーが通常の 1 次サーバーの場合)。このロールでは、1 次サーバーのトランザクションが 2 次サーバーに送信されます。 • PRIMARY NOHSBLOG (1 次サーバーがトランザクションを受け入れ、データベースに格納することを示す)。ただし、それらのトランザクションが後で 2 次側へ送信できるようにログに格納されることはありません。2 次側と 1 次側の再同期を行うには、1 次側にあるデータベース全体を 2 次サーバーにコピーする必要があります。 • PRIMARY BROKEN (1 次サーバーで 2 次サーバーとの間の接続が切れた場合)。1 次サーバーでは読み取り専用トランザクションのみを実行できます。 • PRIMARY ALONE (1 次サーバーが単独で動作している場合)。2 次側との接続は切れていますが、後で 2 次側に送信できるように 1 次側でトランザクションを受け入れ、トランザクション・ログに追加します。 • PRIMARY CATCHUP (キャッチアップが進行中の場合)。キャッチアップ中に、'hsb connect' コマンドが 1 次側で発行されるとその後で、1 次側は自動的にトランザクション・ログの変更を 2 次サーバーへ送信します。キャッチアップ・プロセスの完了後、サーバーのロールは自動的に PRIMARY へ切り替わります。1 次側は、そのロールが接続前に PRIMARY ALONE だった場合には、引き続きトランザクションを受け入れます。 • SECONDARY (接続済みサーバーが通常の 2 次サーバーの場合)。つまり、サーバーは 1 次側からトランザクションを受信して適用します。 • SECONDARY BROKEN (2 次サーバーで 1 次サーバーとの間の接続が切れた場合)。 • SECONDARY CATCHUP ('**hsb connect**' コマンドが 1 次サーバーで発行された後で、2 次サーバーが、1 次サーバーからの変更内容のキャッチアップ中の場合)。キャッチアップ・プロセスの完了後、2 次側のロールは自動的に SECONDARY へ切り替わります。 **hsb role** が、HotStandby 用に構成されていないサーバーで発行された場合は、エラー・メッセージ「14527: This is a non-HotStandby Server」が返されます。 このコマンドは、SQL 関数 HOTSTANDBY_ROLE と同じ情報を返します。

表 37. HotStandby コマンド (ADMIN COMMAND) (続き)

コマンド	説明
hsb set primary alone	このコマンドは、1 次サーバーを無条件に PRIMARY ALONE 状態に設定します。このコマンドが使用可能なのは、サーバーが現在、PRIMARY ACTIVE、SECONDARY ACTIVE、SECONDARY ALONE、および STANDALONE のいずれかの状態になっている場合です。 このコマンドを使用して、高速フェイルオーバーを実装できます。 2 次側が SECONDARY ACTIVE 状態のとき、サーバーは、このコマンドを受信しても、1 次側との通信をまったく試みません。代わりに、すぐに PRIMARY ALONE 状態に切り替えます。この動作は、1 次側の障害に関する情報が、2 次側でその障害を検出する前に HAC に達する場合に利用できます (遅延は **PingTimeout** および **PingInterval** パラメーターに従います)。 ただし、例えば、正しくない障害検出などのため、このコマンドが 2 次側で実行されたときに、1 次側が「活動中」で、PRIMARY ACTIVE 状態であった場合、1 次側は自動的に PRIMARY UNCERTAIN 状態に強制されます。その後で、トランザクションをまったく失うことなく、SECONDARY ALONE 状態に移して再接続することができます。 **注:** フェイルオーバーを実行する代替方法として、**hsb switch primary** コマンドを使用することがあります。 PRIMARY ALONE 状態では、2 次サーバーとの接続は切れますが、この状態では、1 次サーバーがトランザクション・ログの更新を継続して実行できます。PRIMARY ALONE 状態は、1 次サーバーがシャットダウンするまで、2 次サーバーとの接続が正常に行われるまで、あるいはサーバーでトランザクション・ログ用のスペースが使い尽くされるまで持続します。 PRIMARY ALONE 状態に設定した場合、サーバーは、他のサーバーとの接続の再確立を自動的に試みることはありません。 **重要:** このコマンドをサーバー上で実行する前に、ペアのもう一方のサーバーがまだ PRIMARY ALONE 状態 (または STANDALONE 状態) になっていないことを確認してください。これは、*二重 1 次サーバー* を避けるためです。詳しくは、 74 ページの『3.6.2, ネットワーク分割と二重 1 次サーバー』を参照してください。 コマンド **'hsb switch primary'** も参照してください。
hsb set secondary alone	このコマンドは、サーバーの状態を SECONDARY ALONE に設定します。このコマンドが使用可能なのは、サーバーが現在、PRIMARY ALONE、PRIMARY UNCERTAIN、STANDALONE のいずれかの状態になっている場合です。
hsb set standalone	このコマンドが発行された場合、1 次サーバーの状態は STANDALONE になります。サーバーは 2 次サーバーのためのトランザクションの格納を停止します。1 次側 (STANDALONE) は引き続き、読み取り/書き込みトランザクションを受け入れることができます。このオプションは、2 次サーバーが相当の期間オフラインで、トランザクション・ログが大きくなりすぎている可能性がある場合に 1 次サーバーで役立ちます。このコマンドが使用可能なのは、サーバーが現在、PRIMARY ALONE または SECONDARY ALONE の状態になっている場合です。

表 37. HotStandby コマンド (ADMIN COMMAND) (続き)

コマンド	説明
hsb state	サーバーの状態を返します。 • PRIMARY ACTIVE (接続済みサーバーが通常の 1 次サーバーの場合)。この状態では、1 次サーバーのトランザクションが 2 次サーバーに送信されます。 • STANDALONE (1 次サーバーがトランザクションを受け入れ、データベースに格納するが、それらのトランザクションを 2 次側へ転送するための格納ではないことを示す)。 • PRIMARY UNCERTAIN (1 次サーバーで 2 次サーバーとの間の接続が切れ、PRIMARY ALONE などの別の状態にまだ切り替わっていない場合)。1 次サーバーでは読み取り専用トランザクションのみを実行できます。 • PRIMARY ALONE (1 次サーバーが単独で動作している場合)。2 次側との接続は切れていますが、2 次側に転送できるようにトランザクションを受け入れ、1 次側のトランザクション・ログに格納します。 • SECONDARY ACTIVE (接続済みサーバーが通常の 2 次サーバーの場合)。つまり、サーバーは 1 次側からトランザクションを受信して適用します。 • SECONDARY ALONE (2 次サーバーで 1 次サーバーとの間の接続が切れた場合)。 **ADMIN COMMAND 'hsb state'** が、HotStandby 用に構成されていないサーバーで発行された場合は、エラー・メッセージ「`14527: This is a non-HotStandby Server`」が返されます。 このコマンドは、SQL 関数 HOTSTANDBY_STATE と同じ情報を返します。この関数の詳細については、105 ページの『新規 1 次サーバーへのアプリケーションの切り替え』の『*HOTSTANDBY_STATE 関数の使用*』セクションを参照してください。 管理操作およびトラブルシューティング操作の実行中に起きる HotStandby 状態遷移の概要については、177 ページの『付録 D. サーバー状態遷移』を参照してください。
hsb status *option*	このコマンドは、最後に正常に開始した操作の HotStandby 状況情報を返します。オプションには以下のいずれかを使用できます。 • `catchup` • `connect` • `copy` • `switch` 詳細については、下記の個々のコマンド/オプション (例えば、**hsb status catchup**) の説明を参照してください。 status コマンドは、正常に開始されてから長時間かかる操作の結果に関する情報を提供します。例えば、正しくない状態などの原因により、操作の開始が失敗した場合、status コマンドは、その操作ではなく、その前に実行された操作の状況を返します。

表 37. HotStandby コマンド (ADMIN COMMAND) (続き)

コマンド	説明
hsb status catchup	このコマンドは、サーバーがキャッチアップを行っているかどうか、つまり 2 次側が 1 次側のトランザクション・ログを読み取ったときに、その変更内容を適用するかどうかを示します。 可能な戻り値は以下のとおりです。 • ACTIVE • NOT ACTIVE
hsb status connect	このコマンドは、以下の状況情報を返します。 • CONNECTED - 接続がアクティブ。この情報は、1 次サーバーおよび 2 次サーバーの両方から返されます。 • CONNECTING - 1 次サーバーおよび 2 次サーバーが相互に接続しています。この情報は、1 次サーバーおよび 2 次サーバーの両方から返されます。 • CATCHUP - 1 次サーバーが 2 次サーバーに接続していますが、1 次 HotStandby のデータベース・ログが 2 次サーバーに完全にはコピーされていません。この情報は、1 次サーバーおよび 2 次サーバーの両方から返されます。 • BROKEN - 1 次および 2 次サーバー間の接続が切れました。この情報は、1 次サーバーおよび 2 次サーバーの両方から返されます。 **注:** このコマンドは、SQL 関数 HOTSTANDBY_CONNECTSTATUS と同じ情報を返します。この関数の詳細については、 105 ページの『新規 1 次サーバーへのアプリケーションの切り替え』の『*HOTSTANDBY_CONNECTSTATUS 関数の使用*』セクションを参照してください。
hsb status copy	このコマンドを使用すると、最後の **hsb copy** または **hsb netcopy** コマンドの結果を確認できます。この status コマンドは、**hsb copy** ではなく **hsb netcopy** の結果を確認する場合でも、必ずキーワード copy を使用します。 返される状況情報は以下のとおりです。 • SUCCESS - コピーが正常に完了しました。 • ACTIVE - コピー・プロセスがまだアクティブです。 • ERROR *number* - コピーがエラー・コード *number* で失敗しました。
hsb status switch	このコマンドは、HotStandby の切り替え状況情報を返します。 • ACTIVE - コピー・プロセスがまだアクティブです。 • SUCCESS - コピーが正常に完了しました。 • ERROR *number* - コピーがエラー・コード *number* で失敗しました。 • NO SERVER SWITCH OCCURRED BEFORE - 前に切り替えが起きていません。

表 37. HotStandby コマンド (ADMIN COMMAND) (続き)

コマンド	説明
hsb switch primary	このコマンドは、データベース・サーバーを PRIMARY に切り替えます。このコマンドは切り替えプロセスを開始します。開始したプロセスは、**hsb status switch** コマンドを使用してモニターできます。 このコマンドの実行時にサーバー間が接続している場合には、単に状態を逆にするだけです。つまり、古い 1 次側が PRIMARY ACTIVE から SECONDARY ACTIVE に変更され、2 次サーバーが SECONDARY ACTIVE から PRIMARY ACTIVE に切り替わります。 サーバー間が接続していない状態で、サーバーが SECONDARY ALONE 状態の場合には、サーバーを 1 次に切り替えると、結局それは PRIMARY ALONE 状態になります。新しい 1 次サーバーは、もう一方のサーバーに接続して PRIMARY ACTIVE 状態への切り替えを自動的に試行することはありません。 このコマンドは SECONDARY ACTIVE および SECONDARY ALONE 状態の両方で使用可能なため、フェイルオーバーの実行に使用できます。ただし、サーバーは 1 次側との通信を必ず試みるため、ネットワーク・タイムアウトになる可能性があります。したがって、この方式は **hsb set primary alone** コマンドを使用する場合よりも低速です。一方、この方式は *二重の 1 次* に対するより優れた安全対策になります。 **hsb set primary alone** コマンドも参照してください。
hsb switch secondary	このコマンドは、データベースを SECONDARY 状態に切り替えます。すべてのアクティブな書き込みトランザクションが終了します。 **注:** 接続済みデータベース・サーバーが 1 次サーバーである場合は、それが 2 次サーバーになります。古い 2 次サーバーが使用可能な場合は、その古い 2 次サーバーが 1 次に切り替わります (**hsb switch primary** コマンドを参照)。 **注: hsb switch secondary** コマンドがオープン・トランザクション内部で発行された場合には (トランザクションの開始後で COMMIT ステートメントの実行前の Windows 環境内)、COMMIT ステートメントを発行すると、エラー `replicated transaction is aborted` で失敗します。切り替え中に、switch ステートメントが実行されたトランザクションを含む、すべてのトランザクションがロールバックされます。ADMIN COMMAND はトランザクション・コマンドではないため、切り替え自体は正常に行われます (つまり、ロールバックされません)。ただし、管理コマンドは、新規トランザクションがまだ開かれていない場合、強制的に新規トランザクションを開始します。

C.2 高可用性コントローラーのコマンド (ADMIN COMMAND)

HAC コマンドを発行するには、まず、`solidhac.ini` の **HAController.Listen** パラメーターで定義されたポートを使用して HAC に接続する必要があります。例えば、**solsql** や ODBC インターフェースを使用して HAC に接続することができます。

表 38. 高可用性コントローラーのコマンド (ADMIN COMMAND)

コマンド	説明
hac resume 省略形: **hac rs**	このコマンドは、HAC が中断モード (**hac suspend** で設定) になっている場合に HAC 操作を再開します。 solidhac.ini は再読み取りされません。つまり、solidhac.ini の変更内容は有効になりません。
hac setadministrative 省略形: **hac sad**	このコマンドは HAC を ADMINISTRATIVE モードに設定します。
hac setautomatic 省略形: **hac sam**	このコマンドは HAC を AUTOMATIC モードに設定します。
hac shutdown 省略形: **hac sd**	このコマンドは HAC プロセスを終了します。
hac suspend 省略形: **hac sp**	このコマンドは、HAC が ADMIN COMMAND を listen するために使用するスレッド以外のすべての HAC 操作 (スレッド) をシャットダウンします。 HAC が中断されていた場合、ADMIN COMMAND **hac resume** で操作を再開できます。
trace { on | off } hac 省略形: **tra { on | off } hac**	このコマンドは、HAC 操作のトレースを制御します。トレース情報は、HAC の作業ディレクトリーにある hactrace.out に出力されます。 一般に、IBM ソフトウェア・サポートおよび開発チームは、HAC トレース機能を使用してトラブルシューティングを行います。また、調査している問題に関する情報を取得するためにトレースを生成することもできますが、solidDB ソース・コードの知識がないと、その用途はかなり制限されます。 HAC トレース機能を使用しても、パフォーマンスには最小限の影響しか与えません。

付録 D. サーバー状態遷移

この章では、可能な状態遷移 (例えば、OFFLINE から SECONDARY ALONE への遷移) について説明します。

それぞれのサーバー状態の説明が 9 ページの『サーバー状態の説明』にあります。

D.1 HotStandby 状態遷移図

このセクションの図は、起きる可能性のある状態遷移、およびそれらが起きることのある環境を示しています。

例えば、次のようにコマンド **'hsb set primary alone'** を実行することによって、サーバーの状態を PRIMARY UNCERTAIN から PRIMARY ALONE に変更できます。

```
ADMIN COMMAND 'hsb Set Primary Alone';
```

この遷移図を使う際には、以下の点に注意してください。

1. コマンドの完全な構文は示してありません。例えば、以下のように示してあります。

   ```
   'hsb set primary alone'
   ```

   これの完全な構文は以下のとおりです。

   ```
   ADMIN COMMAND 'hsb Set Primary Alone';
   ```

2. **'hsb copy'** に示されている状態遷移パスは **'hsb netcopy'** にも適用されます。
3. コマンドによっては、実行すると失敗することがあります。コマンドが成功または失敗するときは、両方の可能性が示されます。コマンドが失敗したときにどうなるかを分岐によって説明しようとする場合は、'Disconnect' 失敗 のように、「失敗」という単語が付きます。
4. 一部の状態では、動作は **AutoPrimaryAlone** という名前の solid.ini 構成パラメーターの設定によって決まります。多くの場合、このパラメーターを表すのに省略形「APA」を使用します。
5. 図で「イベント」を指している場合、それは内部生成の通知を指しています。これらは、例えば、SQL コマンドの CREATE EVENT に説明されている、ユーザーが通知して待機可能な「イベント」と同じではありません。
6. 図の左上近くに、「'-x backupserver' で始動」とあります。新規 2 次サーバーを始動して、それに「**netcopy**」コマンド経由で 1 次サーバーからデータベースのコピーを取得したい場合には、そのサーバーを (オペレーティング・システムのコマンド行から) コマンド行オプション **-x backupserver** で始動します。これはサーバーに対して、1 次側からの **netcopy** を待機するよう指示しています。サーバーは **netcopy** の受信を待機する一方で、その状態 (またはロール) に関する照会に応答しないことに注意してください。例えば、以下のコマンドを発行した場合、

   ```
   ADMIN COMMAND 'hsb state';
   ```

サーバーは応答しないので、実際には、それが状態「OFFLINE」を返すことを確認するわけではありません。

7. 「rpc」は「リモート・プロシージャー・コール」を表します。「rpc broken」は、1 次および 2 次側が、明示的な切断 (Disconnect) なしで、相互接続が失われたことを意味します。接続は、ネットワークで障害が起きた場合、または一方のサーバーがクラッシュした場合などに失われる可能性があります。
8. 矢印がその開始元の同じ状態にループバックする場合、それはその状態が変化しないことを意味します。例えば、PRIMARY ALONE 状態のサーバーが、もう一方のサーバーに接続を試みたが失敗した場合に、状態は PRIMARY ALONE のままです。

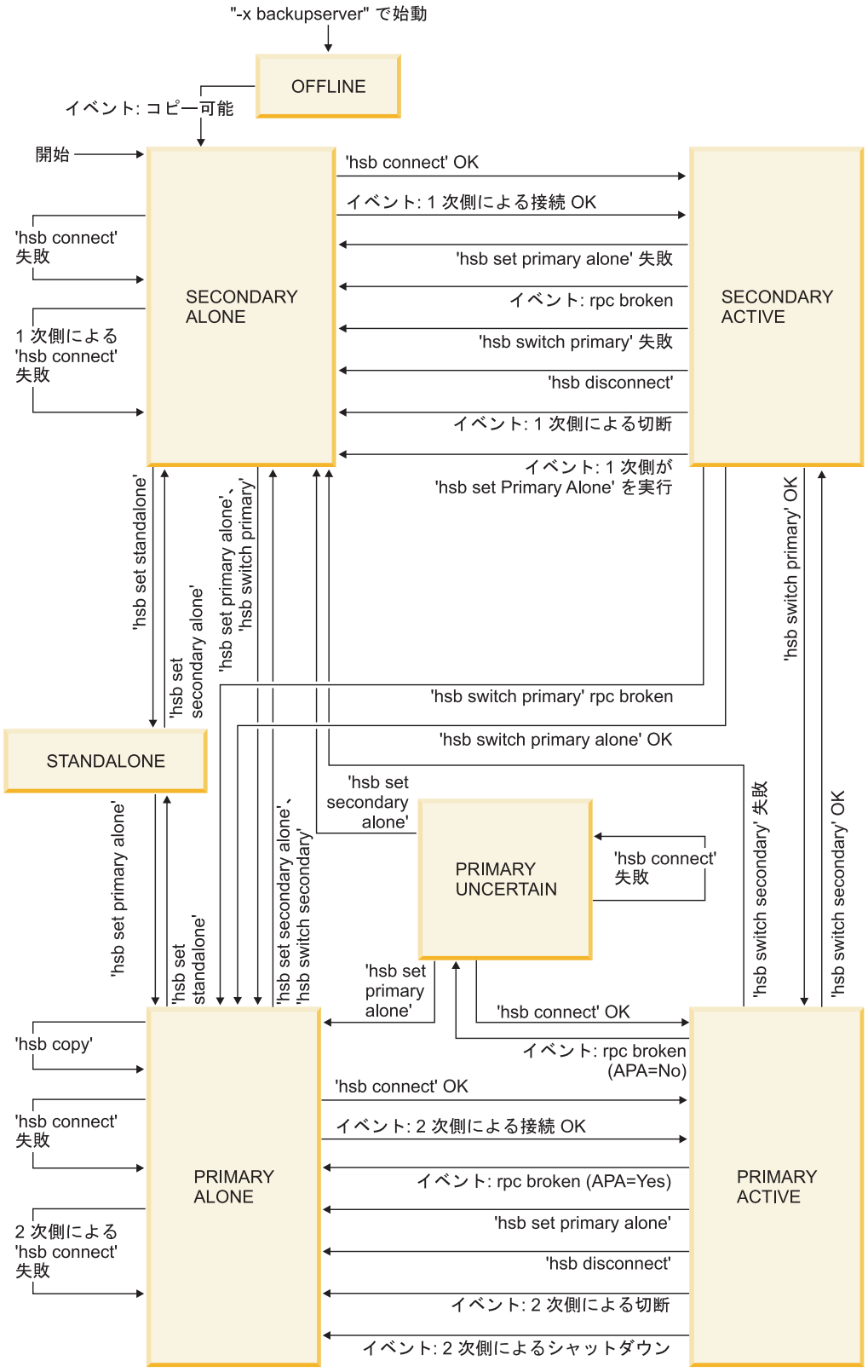


図 19. HotStandby サーバー状態遷移

以下の表は、サーバー状態と、HotStandby コマンドでサーバー状態を変更できる方法を示しています。

表 39. サーバー状態遷移表

サーバー状態	この条件が起きた場合、またはこの HSB コマンドが発行された場合...	サーバーは以下の状態になる...	コマンドが正常でない場合は、以下の状態である...
OFFLINE	1 次サーバーが **ADMIN COMMAND 'hotstandby netcopy'** を実行した場合、2 次サーバーの状態は、データベースのコピー後に SECONDARY ALONE に変更されます。	SECONDARY ALONE	未変更
PRIMARY ACTIVE	**AutoPrimaryAlone** = Yes の場合、HotStandby タイムアウト (自動)。 注: HSB タイムアウトは、2 次サーバーがダウンしたか、または 1 次および 2 次の間の接続が切れた場合に自動的に起きます。	PRIMARY ALONE	(適用外)
PRIMARY ACTIVE	**AutoPrimaryAlone** = No の場合、HotStandby タイムアウト (自動)。 注: HSB タイムアウトは、2 次サーバーがダウンしたか、または 1 次および 2 次の間の接続が切れた場合に自動的に起きます。	PRIMARY UNCERTAIN	(適用外)
PRIMARY ACTIVE	1 次サーバーで **ADMIN COMMAND 'hotstandby set standalone'**	STANDALONE	未変更
PRIMARY ACTIVE	1 次サーバーで **ADMIN COMMAND 'hotstandby switch secondary'** または 2 次サーバーで **ADMIN COMMAND 'hotstandby switch primary'**	SECONDARY ACTIVE	SECONDARY ALONE

表 39. サーバー状態遷移表 (続き)

サーバー状態	この条件が起きた場合、またはこの HSB コマンドが発行された場合...	サーバーは以下の状態になる...	コマンドが正常でない場合は、以下の状態である...
PRIMARY ACTIVE	1 次サーバーで **ADMIN COMMAND 'hotstandby disconnect'**	PRIMARY ALONE	PRIMARY ALONE
PRIMARY ALONE	1 次サーバーで **ADMIN COMMAND 'hotstandby copy'** または **ADMIN COMMAND 'hotstandby netcopy'** 1 次サーバーの状態は変更されないことに注意してください。サーバーは PRIMARY ALONE 状態のままです。状態を PRIMARY ACTIVE に変更するには、「connect」コマンドを **ADMIN COMMAND 'hotstandby connect';** のように発行する必要があります。 注: ディスクレス・サーバーを使用して、2 次サーバーへのファイル・アクセスを行わない場合は、copy ではなく、netcopy を使用してください。	PRIMARY ALONE	PRIMARY ALONE
PRIMARY ALONE	1 次サーバーで **ADMIN COMMAND 'hotstandby connect'** 注: 上記のコマンドは、2 次サーバー (今現在修正されている)、または障害が起きた 2 次側以外のサーバーに接続するために使用します。	PRIMARY ACTIVE (キャッチアップの完了後)	未変更

表 39. サーバー状態遷移表 (続き)

サーバー状態	この条件が起きた場合、またはこの HSB コマンドが発行された場合...	サーバーは以下の状態になる...	コマンドが正常でない場合は、以下の状態である...
PRIMARY ALONE	1 次サーバーで **ADMIN COMMAND 'hotstandby set standalone'**、またはトランザクション・ログがフル	STANDALONE	未変更
PRIMARY ALONE	1 次サーバーで **ADMIN COMMAND 'hotstandby set secondary alone'** または **ADMIN COMMAND 'hotstandby switch secondary'**	SECONDARY ALONE	SECONDARY ALONE
PRIMARY UNCERTAIN	1 次サーバーで **ADMIN COMMAND 'hotstandby set primary alone'** at the Primary server	PRIMARY ALONE	未変更
PRIMARY UNCERTAIN	1 次サーバーで **ADMIN COMMAND 'hotstandby connect'** **注:** 上記のコマンドは、2 次サーバー (今現在修正されている) に接続するため、または障害が起きた 2 次側以外のサーバーに接続するために使用します。	PRIMARY ACTIVE	未変更
PRIMARY UNCERTAIN (2 次側との接続で HSB タイムアウトが起きた)	1 次サーバーで **ADMIN COMMAND 'hotstandby set standalone'**	STANDALONE	未変更
PRIMARY UNCERTAIN	1 次サーバーで **ADMIN COMMAND 'hotstandby set secondary alone'** または **ADMIN COMMAND 'hotstandby switch secondary'**	SECONDARY ALONE	未変更

表 39. サーバー状態遷移表 (続き)

サーバー状態	この条件が起きた場合、またはこの HSB コマンドが発行された場合...	サーバーは以下の状態になる...	コマンドが正常でない場合は、以下の状態である...
SECONDARY ACTIVE	HotStandby タイムアウト (自動) **注:** HSB タイムアウトは、2 次サーバーがダウンしたか、または 1 次および 2 次の間の接続が切れた場合に自動的に起きます。	SECONDARY ALONE	(適用外)
SECONDARY ACTIVE	1 次サーバーで **ADMIN COMMAND 'hotstandby switch secondary'** または 2 次サーバーで **ADMIN COMMAND 'hotstandby switch primary'**	PRIMARY ACTIVE	未変更
SECONDARY ACTIVE	2 次サーバーで **ADMIN COMMAND 'hotstandby set primary alone'**	PRIMARY ALONE	未変更
SECONDARY ACTIVE	2 次または 1 次サーバーで **ADMIN COMMAND 'hotstandby disconnect'**	SECONDARY ALONE	SECONDARY ALONE
SECONDARY ALONE	2 次または 1 次サーバーで **ADMIN COMMAND 'hotstandby connect'**	SECONDARY ACTIVE	未変更
SECONDARY ALONE	2 次サーバーで **ADMIN COMMAND 'hotstandby set standalone'**	STANDALONE	未変更
SECONDARY ALONE	2 次サーバーで **ADMIN COMMAND 'hotstandby set primary alone'** または **ADMIN COMMAND 'hotstandby switch primary'**	PRIMARY ALONE	未変更

付録 E. HotStandby システム・イベント

この付録では、HSB 固有のイベントのみ扱います。他のタイプのイベントについては、「*IBM solidDB SQL ガイド*」など、他のマニュアルを参照してください。

HotStandby 操作はそれぞれがイベントを生成します。これらのイベントをモニターするには、Watchdog アプリケーションなどのアプリケーションを使用することができます。

イベントは、特定のアクションがサーバーで起きたことをシグナル通知する名前付きオブジェクトです。ストアード・プロシージャーの特殊ステートメントがイベントの受信用に必要です。HotStandby イベントは、solidDB によって作成されサポートされる他のイベントと異なる点はまったくありません。それらのイベントは、ストアード・プロシージャーのイベントを受信するように登録済みのユーザーに送信されます。イベントの通知、登録、および待機の詳細については、「*IBM solidDB SQL ガイド*」の『ストアード・プロシージャー、イベント、トリガー、およびシーケンス』と、やはり「*IBM solidDB SQL ガイド*」にある『solidDB SQL 構文』を参照してください。

HotStandby に現在使用可能なイベントを以下の表にリストしてあります。ほとんどのイベントが 5 つのパラメーターを含みますが、そのすべてのパラメーターが必ずしも使用されるわけではない点に注意してください。

表 40. HotStandby イベント

HSB イベント	**イベント・パラメーター**	**イベントの原因**

SYS_EVENT_HSBCONNECTSTATUS	ENAME
	WVARCHAR、POSTSRVTIME
	TIMESTAMP、UID
	INTEGER、NUMDATAINFO
	INTEGER、TEXTDATA WVARCHAR
	
	TEXTDATA の場合、可能な有効値は以下のとおりです。
	`TEXTDATA = {`
	`CONNECTED |`
	`CONNECTING |`
	`CATCHUP |`
	`BROKEN}`
		1 次および 2 次サーバー間の接続状況の変更

表 40. HotStandby イベント (続き)

HSB イベント	イベント・パラメーター	イベントの原因

SYS_EVENT_HSBSTATESWITCH	ENAME
	WVARCHAR、POSTSRVTIME
	TIMESTAMP、UID
	INTEGER、NUMDATAINFO
	INTEGER、TEXTDATA WVARCHAR
	
	TEXTDATA の場合、可能な有効値は以下のとおりです。
	`TEXTDATA = {`
	`PRIMARY ACTIVE |`
	`PRIMARY ALONE |`
	`PRIMARY UNCERTAIN |`
	`SECONDARY ACTIVE |`
	`SECONDARY ALONE |`
	`STANDALONE`
	`}`
		それぞれの状態切り替えで状態切り替えイベントが送信される。

SYS_EVENT_NETCOPYEND	ENAME WVARCHAR、POSTSRVTIME TIMESTAMP、UID INTEGER、NUMDATAINFO INTEGER、TEXTDATA WVARCHAR どのパラメーターも使用されません。	HotStandby NETCOPY 操作が終了した。 このイベントは、ユーザーが共有メモリー・アクセスまたはリンク・ライブラリー・アクセスを使用している場合にのみ ユーザーによってキャッチされます。
SYS_EVENT_NETCOPYREQ	ENAME WVARCHAR、POSTSRVTIME TIMESTAMP、UID INTEGER、NUMDATAINFO INTEGER、TEXTDATA WVARCHAR どのパラメーターも使用されません。	HotStandby NETCOPY が要求された。 ユーザー・アプリケーションのコールバック関数がゼロ以外を返した場合、ネットコピー (netcopy) は実行されません。 このイベントは、ユーザーが共有メモリー・アクセスまたはリンク・ライブラリー・アクセスを使用している場合にのみ ユーザーによってキャッチされます。

付録 F. Watchdog サンプル

このセクションでは、solidDB インストール済み環境に組み込まれているサンプルで使用可能な Watchdog サンプル・アプリケーションについて説明します。

Watchdog は 1 次サーバーと 2 次サーバーをモニターおよび制御する別個のプログラムです。Watchdog は、両方のホット・スタンバイ・サーバーをモニターし、必要なときはそれらの状態を切り替えます。これにより、データベース管理者によるサーバーのモニター作業の必要性が軽減されます。

solidDB には、必要に合わせてカスタム Watchdog を構築するための基礎として使用できる、サンプルの Watchdog が用意されています。このサンプル・アプリケーションは、Watchdog と呼ばれます。プログラミングを開始する前に、Watchdog サンプルの以下の機能に注意してください。

- Watchdog は、Watchdog の例として使用するためのものです。
- Watchdog は、ポーリングを使用して最新のサーバー状態を把握します。
- Watchdog は、1 スレッドのプログラムです。
- Watchdog は、ODBC を通じて HSB API を使用します。この API 実装は、お客様独自の Watchdog アプリケーションのモデルとして使用できます。
- Watchdog には、ユーザー・インターフェースがありません。

Watchdog を使用する場合は、Watchdog の現行作業ディレクトリーにある solidDB 構成ファイル (`solid.ini`) の `[WatchDog]` セクションを構成する必要があります。Watchdog が 1 次サーバーまたは 2 次サーバーと同じディレクトリーで実行されている場合は、1 つの `solid.ini` ファイルをサーバーと Watchdog とで共有することになります。Watchdog が別のディレクトリーで実行されている場合、Watchdog はそれ自体の `solid.ini` ファイルを持ちます。

さらに、この付録には、Watchdog に固有の `solid.ini` 構成パラメーターの説明が含まれています。これらのパラメーターは、`solid.ini` 構成ファイルの `[WatchDog]` セクションで設定されます。独自の Watchdog プログラムを作成する場合は、これらパラメーターをどれも使用する必要はありません。

その他の `solid.ini` パラメーターについては、「*IBM solidDB 管理者ガイド*」を参照してください。

F.1 Watchdog を使用した HotStandby 構成

HotStandby 構成では 1 次サーバー、2 次サーバー、および Watchdog を異なるマシンに常駐させることができ、それらは、 190 ページの『F.1.2, システム設計の問題』の例に示すように、異なるオペレーティング・システムおよび API を使用できます。異機種混合構成の実装の詳細については、 190 ページの『F.1.2, システム設計の問題』を参照してください。

1 次と 2 次のデータベース間のすべての通信は (障害を起こしたシステムをサービスに復帰させ、1 次と 2 次のデータベースを再同期させることも含め)、例えば

TCP/IP など、既存の通信レイヤー内で行われます。HotStandby は、共有ディスクや FTP 転送など、補助ストレージや転送方式を必要としません。

重要: 2 次サーバーが常駐するマシン上で Watchdog を実行している場合は必ず、パラメーター **AutoPrimaryAlone** を no に設定してください。その状況では、**AutoPrimaryAlone** を no に設定することは、非常に重要です。この設定は、1 次サーバーが 2 つになる潜在的なエラーを防止するからです。1 次サーバーは PRIMARY ALONE 状態に置かれる場合があり、サーバー障害時の Watchdog は、2 次サーバーを PRIMARY ALONE 状態に切り替える可能性があります。このエラーは、ユーザーが偶然、旧 2 次サーバーを新規 1 次サーバーになるように設定した場合にも発生することがあります。二重 1 次サーバーの詳細については、74 ページの『3.6.2, ネットワーク分割と二重 1 次サーバー』を参照してください。

F.1.1 Watchdog アプリケーションの動作

Watchdog サンプル・アプリケーションは、1 次サーバーがダウンしたとき、ユーザーに通知します。通常モードでは、Watchdog は 1 次サーバーと 2 次サーバーの両方で、**hotstandby status connect** コマンドを使用してサーバーの接続状況を検査します。

Watchdog プログラムは、サーバー間で、この検査を一定の間隔で実行します。その間隔時間は、Watchdog の solid.ini 構成ファイル内の **PingInterval** パラメーターで設定します。

Watchdog は、指定された回数だけポーリングを試みても、1 次サーバーか 2 次サーバー、またはその両方のノードから応答を受信しなかった場合、HotStandby システムに問題があるという結論に到達します。その試行回数は、Watchdog 構成ファイル (solid.ini 内の [Watchdog] セクション) の **NumRetry** パラメーターで設定されます。

Watchdog は、1 次サーバーと 2 次サーバーが互いに接続されているかどうかも監視します。1 次サーバーまたは 2 次サーバーが接続成功の状況を Watchdog に返した場合、それは 1 次サーバーと 2 次サーバーがまだ接続されていることを意味します。逆に、エラーが返された場合、1 次サーバーと 2 次サーバーは、もはや接続されていません。

Watchdog 構成ファイル内の **AutoSwitch** パラメーターが YES に設定されている場合、Watchdog は、1 次サーバーに障害が起きたときの自動的なサーバー状態の切り替えにも責任を負います。例えば、1 次サーバーがダウンした場合、Watchdog は 2 次サーバーを新規 1 次サーバーになるように切り替え、PRIMARY ALONE 状態にします。**AutoSwitch** パラメーターが NO に設定されている場合、Watchdog はサーバー状態自体を変更せず、代わりに Watchdog ログにメッセージを書き込み、ユーザーにサーバー状態を切り替えるよう通知します。

モニターを続行するために、Watchdog は障害モードに切り替わります。これは、接続が機能するよう、障害を起こしたサーバーの検査を継続することを意味します。

障害モード

Watchdog サンプル・アプリケーションが HotStandby の 1 次サーバーと 2 次サーバーが接続していることを認識した場合、Watchdog は通常モードのままです。いず

れかのサーバーが障害を起こした場合、またはそれらのサーバー間の通信リンクに障害がある場合、Watchdog は何らかのアクションを実行します。そのアクションでサーバーの接続に失敗した場合、Watchdog は障害モードに入ります。

Watchdog は障害モードに入った後、システム管理者が 1 次サーバーまたは 2 次サーバーの問題を修正するのを待ちます。その間に 2 番目の障害が発生した場合、Watchdog はその障害を処理しません。この Watchdog での制限は、意図的なものです。状況によっては、連続した障害により、また、外見上では適切に見える応答さえ、2 つの 1 次サーバー (PRIMARY ALONE または STANDALONE 状態) というエラーを起こす可能性があります。これは特に、ネットワーク内に短時間の障害があっても、データベース・サーバー自体には障害がない場合に当てはまります。2 つの 1 次サーバーを生成する例は、『複数障害用の Watchdog のコーディング』で提供されています。

障害モードの間、Watchdog は 1 次サーバーと 2 次サーバーの両方に対してポーリングを行います。両方のサーバーに接続できる場合は、両方のサーバーに **hotstandby state** コマンドを送信し、それらのサーバーと通信できるかどうか、およびそれらがどのような状態にあるかを調べます。

Watchdog は、両方のサーバーと通信できる場合、solid.ini のパラメーター **DualSecAutoSwitch** に基づいて、次に何をすべきかを決めます。**DualSecAutoSwitch** = Yes で、両方のサーバーが 2 次サーバーである場合、Watchdog はその 2 つの 2 次サーバーのうちの 1 つを新規 1 次サーバーとして自動的に選択し、それを 1 次サーバーに切り替えます。 **DualSecAutoSwitch** = No の場合は、システム管理者が一方のサーバーを 1 次サーバーに切り替える必要があります。 **DualSecAutoSwitch** は、Watchdog が「通常」モードであっても「障害」モードであっても適用されることに注意してください。

複数障害用の Watchdog のコーディング

Watchdog で複数の障害を処理するには、以下の 2 とおりの方法があります。以下のことができます。

- それぞれの障害 (および Watchdog による自動応答) の後、状態を検査するために、手操作 (人) による介入を義務付けることができます。手操作による介入には、サーバーの再始動やネットワーク問題の修正などが必要になることがあります。これは、1 次サーバーが 2 つになる危険を少なくするために Watchdog が使用する手法です。
- 長い時間をかけて、複数の障害を処理できる Watchdog アプリケーションを作成できます。

  この方式では、以下の例に示すように、1 次サーバーが 2 つになる危険があります。

二重 1 次サーバー

この例では、Server1 が初期の 1 次サーバーで、Server2 が初期の 2 次サーバーです。

1. ネットワーク障害が発生し、Server1 がアクセス不能になります。
2. Watchdog は Server2 を SECONDARY から PRIMARY ALONE に切り替えます。

3. 2 番目のネットワーク障害が発生し、Server2 がアクセス不能になります。
4. 最初のネットワーク障害が修復され、Server1 が再びアクセス可能になります。
5. Watchdog は、Server1 がアクセス可能で Server2 がそうでないことを認識し、Server1 を PRIMARY ALONE に切り替えます。
6. 2 番目のネットワーク障害が修正され、Server2 が再びアクセス可能になります。
7. この時点で、Server1 と Server2 の両方が PRIMARY ALONE 状態になります。

F.1.2 システム設計の問題

HotStandby をどのように構成するか (ローカルか、リモートか、1 つ以上の異なる場所か、インターネット上か、Watchdog プログラムと一緒か) によって、システムの信頼性と効率が影響を受けることがあります。このセクションでは、それらの問題について説明します。

以下の図は、異機種混合システムの 1 つの例を示しており、ここでは、1 次サーバーと 2 次サーバーが使用するハードウェアとオペレーティング・システムさえ異なっています。

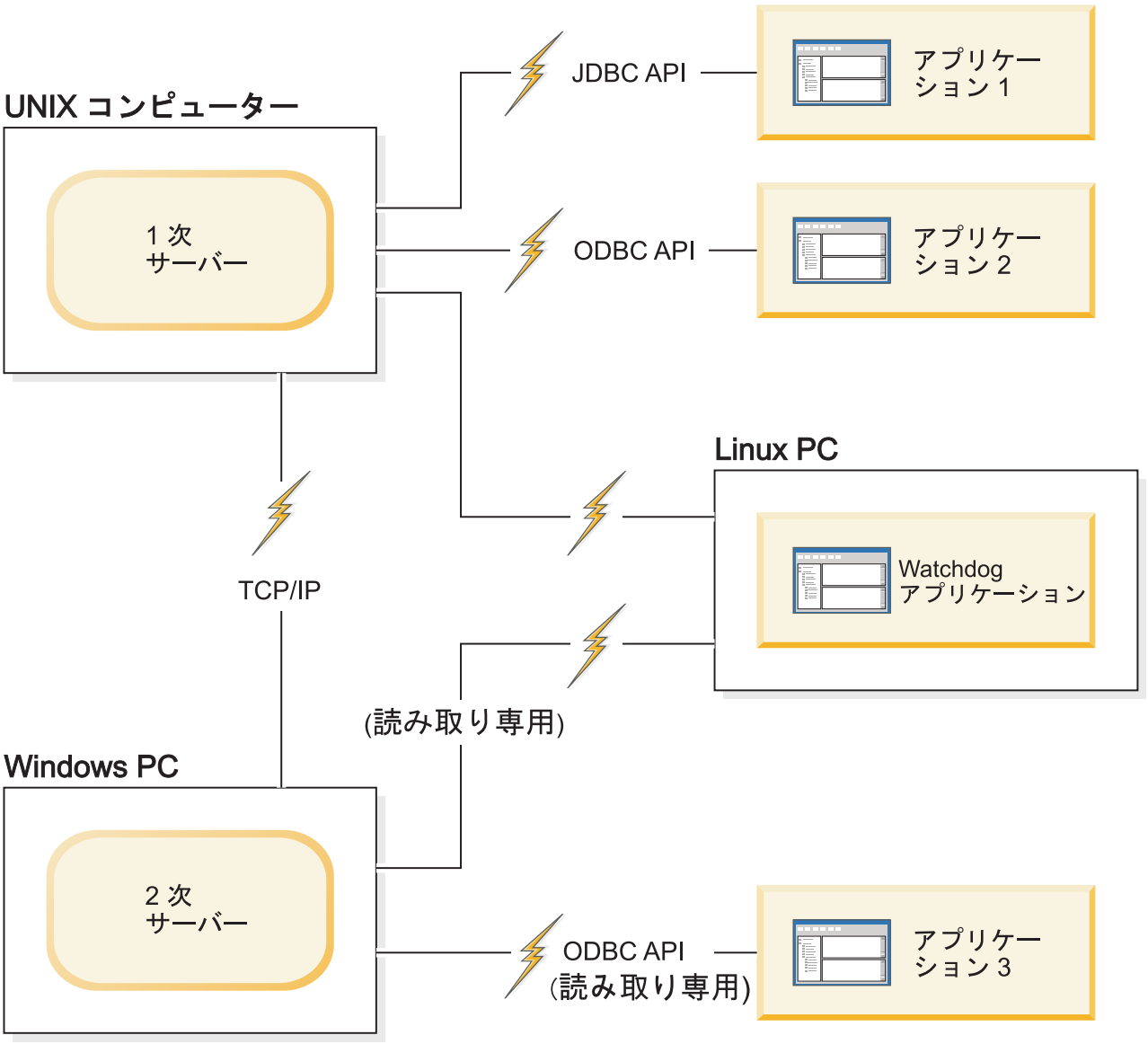


図 20. Watchdog を使用した異機種混合 HotStandby 構成

F.1.3 Watchdog 構成

サーバー状態のモニターの効率と精度を高めるために、Watchdog を HotStandby 構成とは別のコンポーネントとして使用することをお勧めします。

2 台のマシンだけが使用可能で、Watchdog プログラミングを別のマシンで実行できない場合は、Watchdog を 2 次サーバーが常駐するマシンで実行し、1 次サーバーと 2 次サーバーの両方の構成ファイル (`solid.ini`) でパラメーター **`AutoPrimaryAlone`** を `no` に設定してください。このパラメーターを `no` に設定することは、きわめて重要です。なぜなら、これは 1 次サーバーが 2 つになる潜在的なエラーを防止するからです。

注意:
両方のサーバーが書き込みを許可する状態 (PRIMARY ALONE または STANDALONE) にあり、両方のサーバーのデータベースが独立して更新された場合、2 つのデータベースを再同期させることができなくなります。両方のサーバーが同時に PRIMARY ALONE 状態または STANDALONE 状態に置かれることを、Watchdog が許可しないようにしてください。 74 ページの『3.6.2, ネットワーク分割と二重 1 次サーバー』を参照してください。

1 次サーバーが失敗した場合、Watchdog は 2 次サーバーを切り替えて、新規 1 次サーバーにすることができます。

Watchdog を 2 次サーバーと同じマシンに配置することには、いくつかの欠点があります。それらの欠点は、以下のとおりです。

- この構成では、Watchdog と 1 次サーバーの間の通信リンクがダウンしただけで、1 次サーバーと 2 次サーバーの間で誤った切り替えが起きる可能性があります。
- 通信リンクは「Single Point of Failure」になります。つまり、単一の障害でシステム全体が使用不可になる可能性があります。(ほとんどの HotStandby 構成では、少なくとも 2 つの障害が存在しなければ、システム全体は使用不可になりません。)
- ネットワーク障害が存在し、2 次サーバー・マシンが 1 次サーバー・マシンと通信できない場合でも、ユーザーとアプリケーションは 1 次サーバーに引き続きアクセスでき、理論的には 1 次サーバーを使用して操作を続行できます。しかし、1 次サーバーはトランザクションの受け入れを停止します。Watchdog が 1 次サーバーに、例えば、PRIMARY ALONE 状態に切り替えるなどの方法で操作を続行するよう通知できないからです。

F.1.4 サンプルの Watchdog アプリケーションの使用

このタスクについて

初期には、両方のサーバーが始動して接続された後に Watchdog を開始してください。

手順

Watchdog を開始するには、Watchdog の現行作業ディレクトリーに移動し、プロンプトで以下のコマンドを発行します。

```
watchdog
```

solid.ini ファイルで (1 次サーバーと 2 次サーバーとしてサービスを提供できる) connect1 サーバーと connect2 サーバーのユーザー名とパスワードを指定しなかった場合、Watchdog はそれらを入力するためのプロンプトを出します。

タスクの結果

開始されると、Watchdog は両方のサーバーに ping を実行し、どちらが 1 次サーバーであるかを検査します。Watchdog は、再試行回数を超えた後にサーバー障害を検出した場合以外、通常モードのままです。Watchdog が最後の再試行をサーバーへ送信した後に障害が発生した場合、Watchdog は障害モードに切り替わります。1 次

サーバーと 2 次サーバーの両方が始動して再接続されると、Watchdog は通常モードに切り替わります。

F.2 障害の状態と Watchdog のアクション

このセクションでは、よくある具体的な障害のシナリオで、標準的な Watchdog プログラムがどのように機能するかについて説明します。

これらのシナリオは、どちらかのサーバーの障害か、1 次サーバーと 2 次サーバーの間の通信リンクの切断、あるいはサーバーの 1 つと Watchdog との間の通信リンクの切断というコンテキストでのシナリオです。

これらのコマンドは、人間の管理者とソフトウェア・プログラムのどちらが発行してもかまいませんが、分かりやすくするために、コマンドが Watchdog サンプルによって発行されることを想定しています。

F.2.1 1 次サーバーがダウンした場合

シナリオ

1 次サーバーへのすべての接続が切断されました。

修復方法

1 次サーバーがダウンしたときは、2 次サーバーを切り替えて新規 1 次サーバーとし、新規 1 次サーバーを PRIMARY ALONE 状態に設定します。後で、旧 1 次サーバーを新規 2 次サーバーにすることができます。

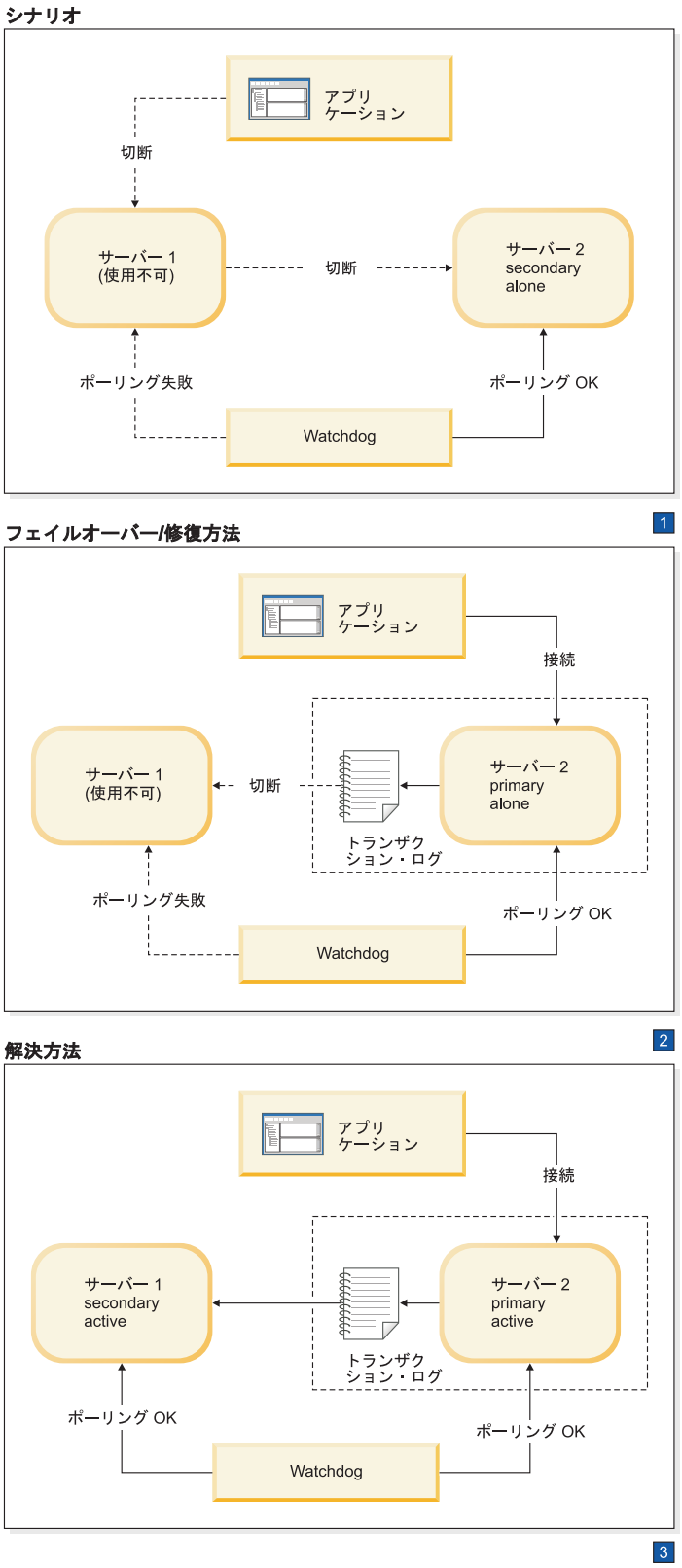


図 21. 1 次サーバーがダウンしたシナリオと修復方法

1. Watchdog がサーバー 2 に指示します。

HSB SET PRIMARY ALONE

アプリケーションがサーバー 1 からサーバー 2 に切り替えます。

HSB SET STANDALONE

2. サーバー 1 が修正された後に、サーバー 1 が 2 次サーバーとして復帰します。

   Watchdog がサーバー 2 に指示します。HSB COPY または NETCOPY HSB CONNECT

3. トランザクション・ログが満杯である場合は、PRIMARY ALONE サーバーを STANDALONE に切り替えなければならないことがあります。その場合は、サーバーを再接続する前に HSB COPY または HSB NETCOPY を実行する必要もあります。トランザクション・ログが満杯でない場合は、COPY/NETCOPY コマンドをスキップする必要があります。

症状

アプリケーションが 1 次サーバーに接続できません。また、Watchdog によるポーリングが 1 次サーバーで失敗します。2 次サーバーの HSB 状態は SECONDARY ALONE です。

1 次サーバーがダウンしたときのリカバリーの方法

このタスクについて

「HotStandby」(2 次サーバー) が 1 次サーバーに置き換わることができるようにするには、以下の手順を実行します。

手順

1. 以下のコマンドを使用して、新規 1 次サーバーを PRIMARY ALONE 状態に設定します。

   ```
   ADMIN COMMAND 'hotstandby set primary alone';
   ```

2. アプリケーションを新規 1 次サーバーに再接続します。
3. アプリケーションの使用を開始します。
4. 旧 1 次サーバーを修正し、新規 2 次サーバーとして始動します。
5. 必要であれば、以下のコマンドを使用して、データベースを新規 1 次サーバーから新規 2 次サーバーにコピーします。

   ```
   ADMIN COMMAND 'hotstandby netcopy';
   ```

   詳細については、 56 ページの『3.4.5, 1 次サーバーと 2 次サーバーの同期』を参照してください。

6. 以下のコマンドを使用して、新規 1 次サーバーを新規 2 次サーバーに再接続します。

   ```
   ADMIN COMMAND 'hotstandby connect';
   ```

F.2.2 2 次サーバーがダウンした場合シナリオ

2 次サーバーへのすべての接続が切断されました。この原因としては、2 次サーバー内の問題またはネットワーク内の障害により、1 次サーバーと Watchdog がどち

らも 2 次サーバーと通信できなくなったことが考えられます。このセクションでは、2 次サーバーに障害があるとしていますが、実際には、2 次サーバーかネットワークに問題があることも考えられます。

修復方法

標準的な修復方法は、1 次サーバーを PRIMARY ALONE 状態に切り替えることです。2 次サーバーが再び稼働したら、1 次サーバーとの同期をとります。

2 次サーバーとの接続に問題が見つかると、1 次サーバーは以下のことを行います。

1. オープン・トランザクションを一時的に中断し、トランザクションのコミットもロールバックも行いません (1 次サーバーは、クライアントにエラー・メッセージ、または「成功」メッセージを送信しません)。しかも、
2. 自身の状態を自動的に PRIMARY ACTIVE から PRIMARY UNCERTAIN に切り替えます。

一般に、Watchdog は、2 次サーバーが使用不可であることを確認した後、1 次サーバーを PRIMARY UNCERTAIN から PRIMARY ALONE に切り替えます。1 次サーバーが PRIMARY ALONE 状態に切り替えられた後、1 次サーバーはトランザクションの受け入れを続行でき、トランザクションを 2 次サーバーへ送信するために保存します。その後、2 次サーバーが復帰した時点で、2 次サーバーにトランザクション・ログを送信し、2 次サーバーが 1 次サーバーに「キャッチアップ」することができるようになります。

1 次サーバーは、PRIMARY ALONE 状態に設定された後、オープン・トランザクションをコミットします。2 次サーバーがまだコミットしていないトランザクションを 1 次サーバーがコミットする可能性を回避するために、トランザクションは、あたかも 2 次サーバーへ送信されなかったかのように、トランザクション・ログ内に保持されます。2 次サーバーが復帰してキャッチアップを開始すると、1 次サーバーは上記のトランザクション・ログを送信し、2 次サーバーは各トランザクションを検査します。トランザクションが重複している場合 (つまり、2 次サーバーが、障害を起こす前にそのトランザクションを既にコミットしてある場合)、重複するトランザクションは 2 次サーバー上で再実行されません。

Watchdog またはシステム管理者は、1 次サーバーを PRIMARY ALONE 状態にするか、それとも代替アクションを選択するかを慎重に選択する必要があります。1 次サーバーを PRIMARY ALONE 状態に切り替える以外のアクションを Watchdog またはシステム管理者が選択する場合は、2 次サーバーおよび 1 次サーバーが同じデータを持っていない可能性 (つまり、両方のサーバーがトランザクションのロールバックを済ませていない可能性) を考慮に入れる必要があります。障害を起こした 2 次サーバーが実際にデータをコミットし、コミット後、1 次サーバーへ確認を送信する前に破損したために、1 次サーバーがコミットしていないことも考えられます。その状態では、2 次サーバーは実際には 1 次サーバーより遅れているのではなく、1 次サーバーの「先」にいる可能性があります。

また、いつもそうですが、Watchdog または管理者は、両方のサーバーが同時に PRIMARY ALONE 状態に入らないように注意する必要があります。

下の図は、3 つのフレームに分かれています。最初のフレームは、1 次サーバーと Watchdog が 2 次サーバーとの接続を失ったシナリオを示しています。その次のフレームは、問題を完全に解決できるまでの間、システムを機能させておくための対応の方法を示しています。 3 番目のフレームは、問題が解決した後、つまり、壊れたサーバーが修正された後か通信が復元された後の最終状態を示しています。

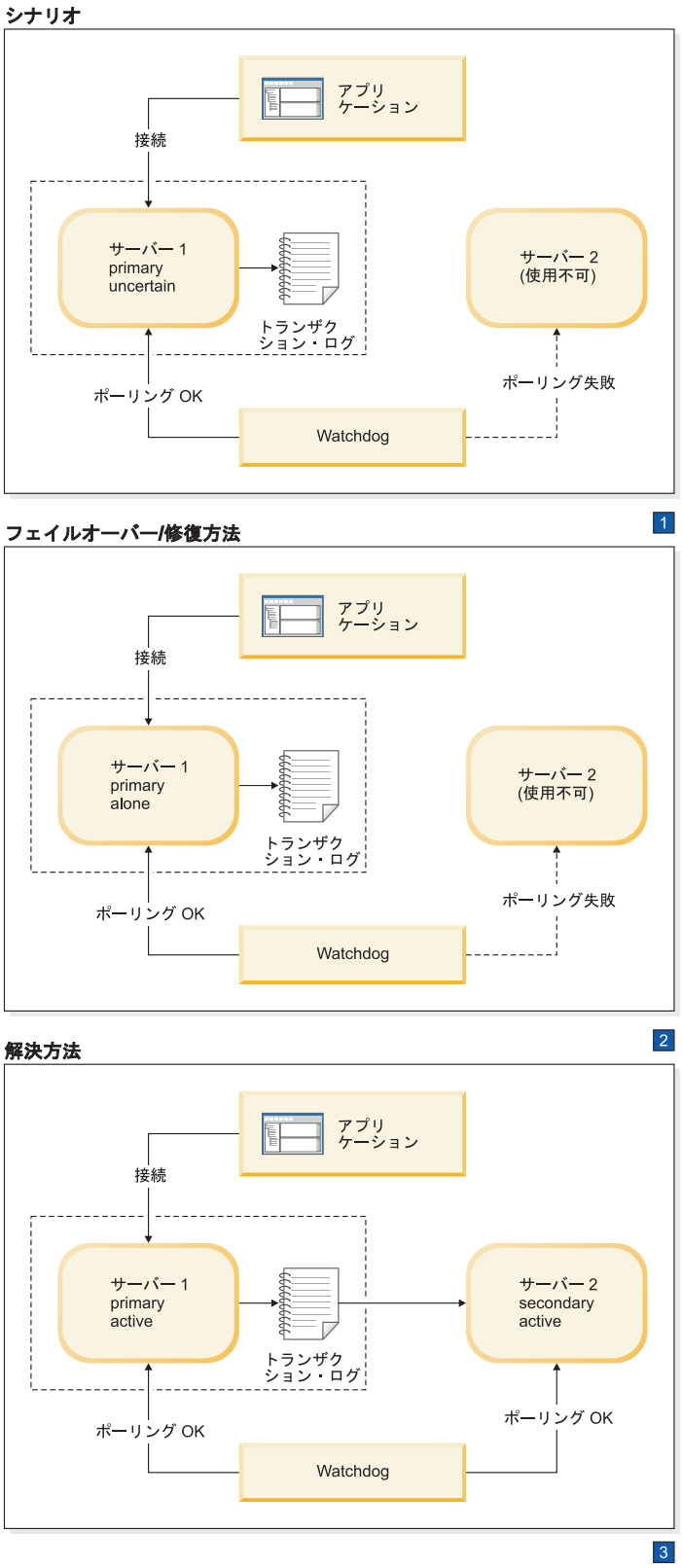


図 22. 2 次サーバーがダウンしたシナリオと修復方法

1. Watchdog がサーバー 1 に指示します。

HSB SET PRIMARY ALONE

HSB SET STANDALONE

2. サーバー 2 が復帰した後に、Watchdog がサーバー 1 に指示します。

   HSB COPY または

   NETCOPY HSB CONNECT

3. トランザクション・ログが満杯である場合は、PRIMARY ALONE サーバーを STANDALONE に切り替えなければならないことがあります。その場合は、サーバーを再接続する前に HSB COPY または HSB NETCOPY を実行する必要もあります。トランザクション・ログが満杯でない場合は、COPY/NETCOPY コマンドをスキップする必要があります。

症状

Watchdog のポーリングが 2 次サーバーで失敗します。1 次サーバーの状態は、PRIMARY ALONE か PRIMARY UNCERTAIN です。

2 次サーバーがダウンしたときのリカバリーの方法

このタスクについて

1 次サーバーが 2 次サーバーから独立して作動し、トランザクションの受信を続行できるようにするには、以下の手順を実行します。

手順

1. 1 次サーバーが PRIMARY UNCERTAIN 状態にある場合は、以下のコマンドを使用して、1 次サーバーを PRIMARY ALONE に設定します。

   ```
   ADMIN COMMAND 'hotstandby set primary alone';
   ```

2. 2 次サーバーが修復されて再始動されるか、2 次サーバーのネットワーク接続が再確立された後 (またはその両方の後)、以下のコマンドを使用して 1 次サーバーの状態を検査します。

   ```
   ADMIN COMMAND 'hotstandby state';
   ```

3. 1 次サーバーの状態が PRIMARY ALONE の場合は、以下のコマンドを使用して、1 次サーバーを 2 次サーバーに再接続します。

   ```
   ADMIN COMMAND 'hotstandby connect';
   ```

4. 以前に 1 次サーバーの状態を STANDALONE に変更した場合は、以下のようにします。

   a. 以下のコマンドを使用して、データベースを新規 1 次サーバーから新規 2 次サーバーにコピーします。

      ```
      ADMIN COMMAND 'hotstandby netcopy';
      ```

   b. 詳細については、56 ページの『3.4.5, 1 次サーバーと 2 次サーバーの同期』を参照してください。

5. 以下のコマンドを使用して、1 次サーバーを 2 次サーバーに再接続します。

   ```
   ADMIN COMMAND 'hotstandby connect';
   ```

2 次サーバーがダウンした場合の追加シナリオ

アプリケーションが 1 次サーバーからエラー・メッセージ 10047 または 14537 を受け取った場合は、以下のようにします。

- 2 次サーバーの状態が新規 1 次サーバーに切り替えられているかどうかを検査するために、2 次サーバーに接続を試みます。
- 2 次サーバーの状態が 1 次サーバーのいずれかの状態 (PRIMARY ACTIVE または PRIMARY ALONE) でない場合は、 193 ページの『F.2.1, 1 次サーバーがダウンした場合』のシナリオを参照してください。

F.2.3 Watchdog がダウンした場合

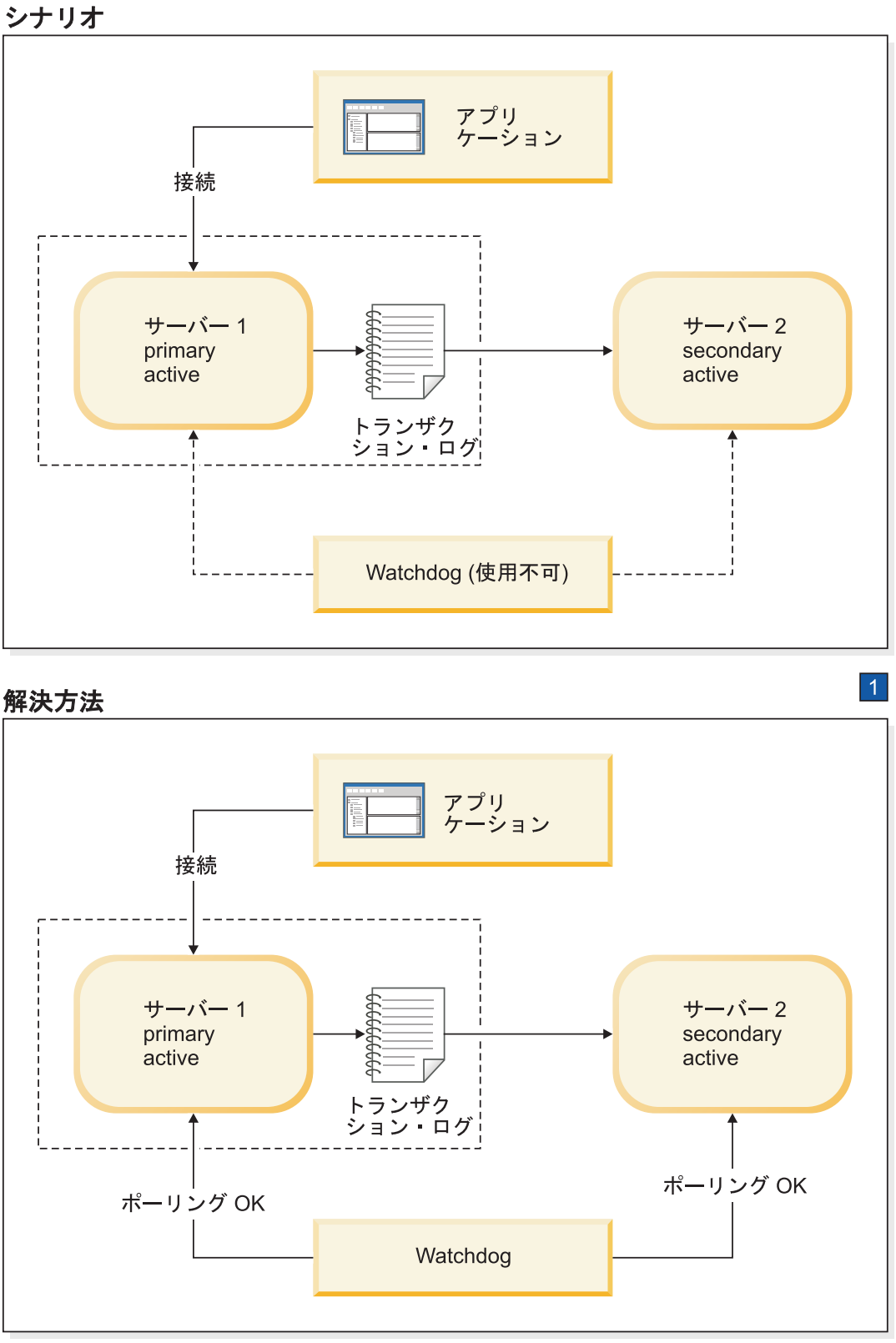
このセクションでは、Watchdog に障害が起きた場合にどうなるかを説明します。

シナリオ

Watchdog へのすべての接続が切断されました。

修復方法

手操作による介入が必要です。Watchdog が起動した時点で、必ず 1 次サーバーと 2 次サーバーを検査して、それらの状態を確認してください。



1. Watchdog を復帰させるか、ネットワークを修正します。

図 23. Watchdog がダウンしたシナリオと修復方法

症状

Watchdog プロセスがダウンするか、Watchdog から両方のサーバーへのネットワーク接続が使用不可になります。

追加シナリオ

各サーバーが状態を変更し、1 つのサーバーがもはや機能しなくなった場合は、このセクションにある該当するシナリオの説明を参照してください。

Watchdog がダウンした場合のリカバリーの方法

このタスクについて

Watchdog へのすべての接続が切断されたシナリオからリカバリーを行うには、以下の手順を実行します。

手順

1. 1 次サーバーと 2 次サーバーが通常操作を続行できるようにします。
2. Watchdog が起動した後、以下のコマンドで Watchdog に各サーバーの状態を検査させます。

   ```
   ADMIN COMMAND 'hotstandby state';
   ```

F.2.4 1 次サーバーと 2 次サーバーの間の通信リンクがダウンした場合

シナリオ

1 次サーバーと 2 次サーバーの間の接続が切断されました。

1 次サーバーは、自身を PRIMARY UNCERTAIN 状態に切り替えます。(**AutoPrimaryAlone** が Yes に設定されている場合、サーバーは自身を PRIMARY ALONE 状態に切り替えます。)

注: 1 次サーバーが 2 次サーバーにコミット・メッセージを送信し、2 次サーバーの障害を検出した場合、1 次サーバーがどのように進行するかは Watchdog または管理者に依存します。その理由は、2 次サーバーが障害を起こす前にトランザクションが 2 次サーバー内でコミットされたかロールバックされたかを、1 次サーバーは検出できないからです。

1 次サーバーは Watchdog または管理者からコマンドを受け取るまで、トランザクションを受け入れなくなります。この段階で、1 次サーバーに操作を続行させるために、Watchdog または管理者は 1 次サーバーを PRIMARY ALONE 状態に設定できます。

修復方法

1 次サーバーは、2 次サーバーへのリンクがダウンした場合でも、操作を続行できます。1 次サーバーがまだ PRIMARY ALONE 状態でない場合は、1 次サーバーを PRIMARY ALONE 状態に切り替えてください。1 次サーバーと 2 次サーバーの間のリンクが復元された後、データベース同士の同期をとります。

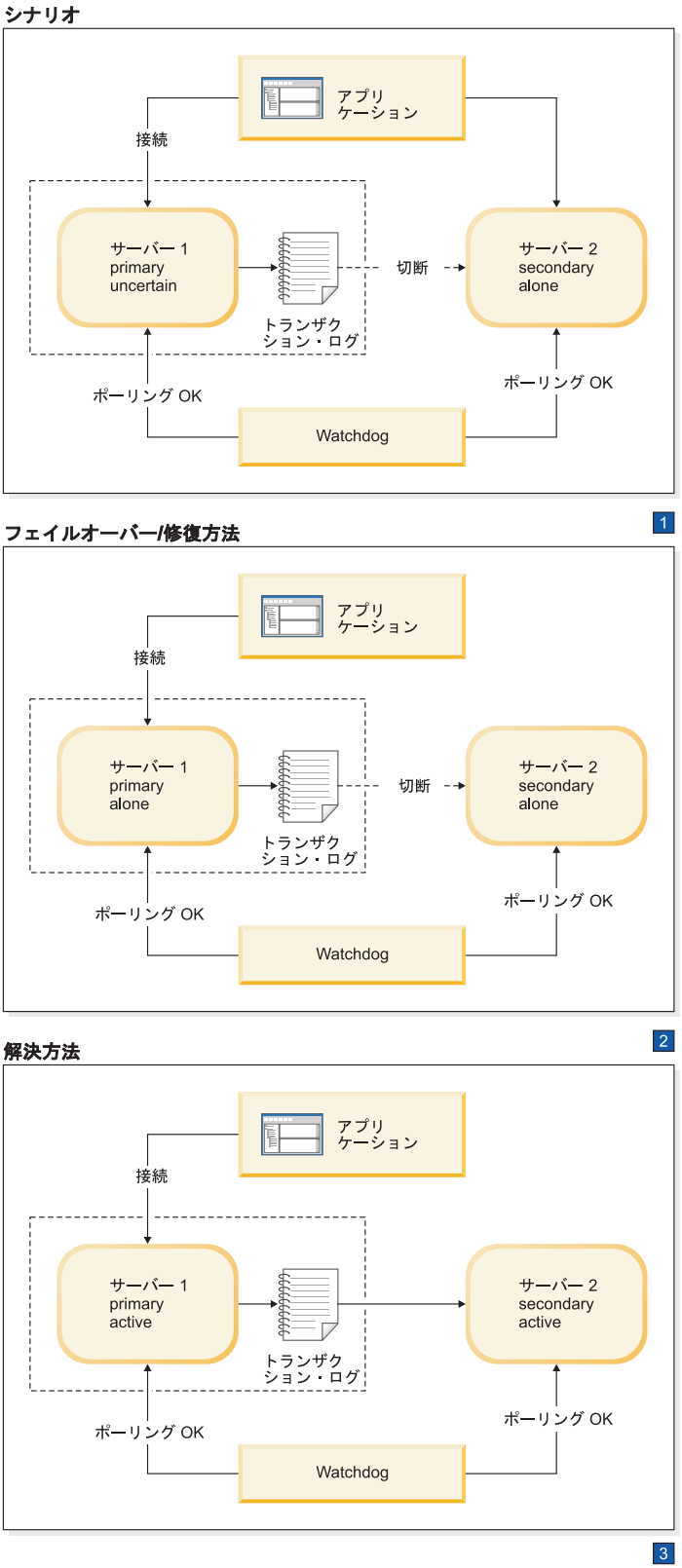


図 24. 1 次サーバーと 2 次サーバーの間のリンクが切断されたシナリオと修復方法

1. Watchdog がサーバー 1 に指示します。

HSB SET PRIMARY ALONE

HSB SET STANDALONE

2. 1 次サーバーと 2 次サーバーの間の接続が修正された後に、Watchdog がサーバー 1 に指示します。

   HSB COPY または

   NETCOPY HSB CONNECT

3. トランザクション・ログが満杯である場合は、PRIMARY ALONE サーバーを STANDALONE に切り替えなければならないことがあります。その場合は、サーバーを再接続する前に HSB COPY または HSB NETCOPY を実行する必要もあります。トランザクション・ログが満杯でない場合は、COPY/NETCOPY コマンドをスキップする必要があります。

症状

1 次サーバーは 2 次サーバーと接続しておらず、状態は PRIMARY UNCERTAIN または PRIMARY ALONE です。

1 次サーバーと 2 次サーバーの間の通信リンクがダウンした場合のリカバリーの方法

このタスクについて

1 次サーバーと 2 次サーバーの間の接続が切断されたシナリオからリカバリーを行うには、以下の手順を実行します。

手順

1. 1 次サーバーと 2 次サーバーの間のネットワーク接続を修正します。
2. 以下のコマンドを使用して、1 次サーバーの状態を検査します。

   ```
   ADMIN COMMAND 'hotstandby state';
   ```

3. 1 次サーバーの状態が PRIMARY ALONE の場合は、以下のコマンドを使用して、1 次サーバーを 2 次サーバーに再接続します。

   ```
   ADMIN COMMAND 'hotstandby connect';
   ```

4. 1 次サーバーの状態が STANDALONE の場合は、以下の手順を実行します。
   a. データベースを 1 次サーバーから 2 次サーバーにコピーします。詳細については、56 ページの『3.4.5, 1 次サーバーと 2 次サーバーの同期』を参照してください。

      コマンド **ADMIN COMMAND 'hotstandby netcopy';** を使用する前に、2 次サーバーが稼働中であり、**netcopy** を受信する準備が整っていることを確認してください。また、1 次サーバーの状態を PRIMARY ALONE に設定してあることも確認してください。

   b. 以下のコマンドを使用して、1 次サーバーを 2 次サーバーに再接続します。

      ```
      ADMIN COMMAND 'hotstandby connect';
      ```

1 次サーバーと 2 次サーバーの間の通信リンクがダウンした場合の追加シナリオ

アプリケーションが 1 次サーバーからエラー・メッセージ 10047 または 14537 を受け取った場合は、以下のようにします。

- 2 次サーバーが新規 1 次サーバーとして切り替えられているかどうかを検査するために、2 次サーバーに接続を試みます。
- 旧 2 次サーバーが新規 1 次サーバーとして切り替えられていない場合は、193 ページの『F.2.1, 1 次サーバーがダウンした場合』のシナリオを参照してください。

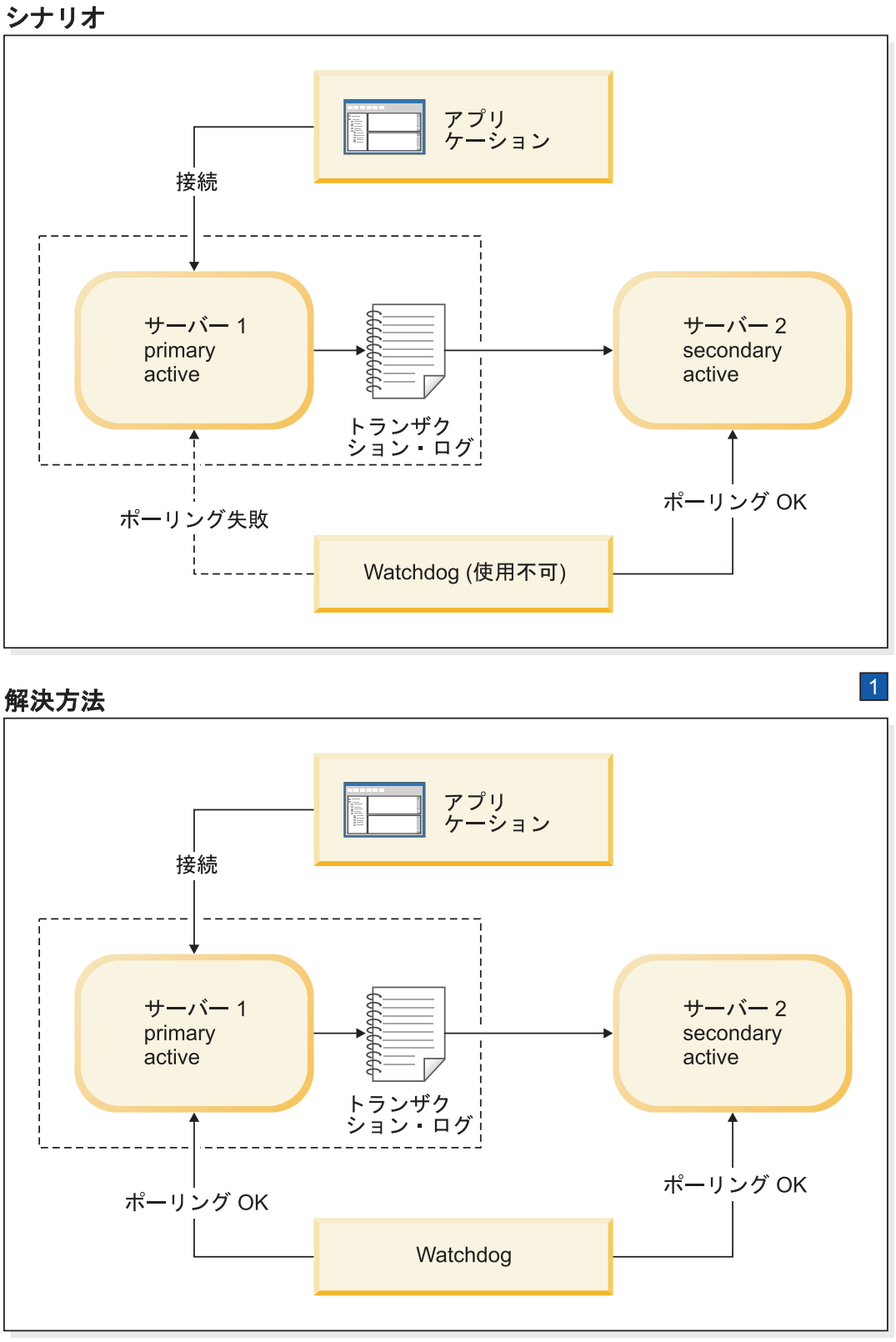
F.2.5 Watchdog と 1 次サーバーの間の通信リンクがダウンした場合

シナリオ

Watchdog と 1 次サーバーの間の接続が切断されました。

修復方法

1 次サーバーおよび 2 次サーバーは、Watchdog から 1 次サーバーへのリンクがダウンした場合でも、操作を続行できます。Watchdog から 1 次サーバーへのリンクが修正されたら、必ず 1 次サーバーと 2 次サーバーの状態を検査してください。



1. サーバー 1 との Watchdog のネットワーク接続を修正します。

図 25. Watchdog と 1 次サーバーの間のリンクが切断されたシナリオと修復方法

症状

Watchdog のポーリングが 1 次サーバーで失敗します。しかし、2 次サーバーの状態は SECONDARY ACTIVE として報告されます。これは、1 次サーバーには問題がない可能性が大きく、単に Watchdog と 1 次サーバーとの接続が失われたことを意味しています。

追加シナリオ

各サーバーの状態が変更され、1 つのサーバーがもはや機能しなくなった場合は、このセクションにある該当するシナリオの説明を参照してください。

Watchdog と 1 次サーバーの間の通信リンクがダウンした場合のリカバリーの方法

このタスクについて

Watchdog と 1 次サーバーの間の接続が切断されたシナリオからリカバリーを行うには、以下の手順を実行します。

手順

1. 1 次サーバーと 2 次サーバーが通常操作を続行できるようにします。
2. Watchdog と 1 次サーバーの間のネットワーク接続を修正します。
3. ネットワークが接続されたら、以下のコマンドで Watchdog に各サーバーの状態を検査させます。

   ```
   ADMIN COMMAND 'hotstandby state';
   ```

F.2.6 Watchdog と 2 次サーバーの間の通信リンクがダウンした場合

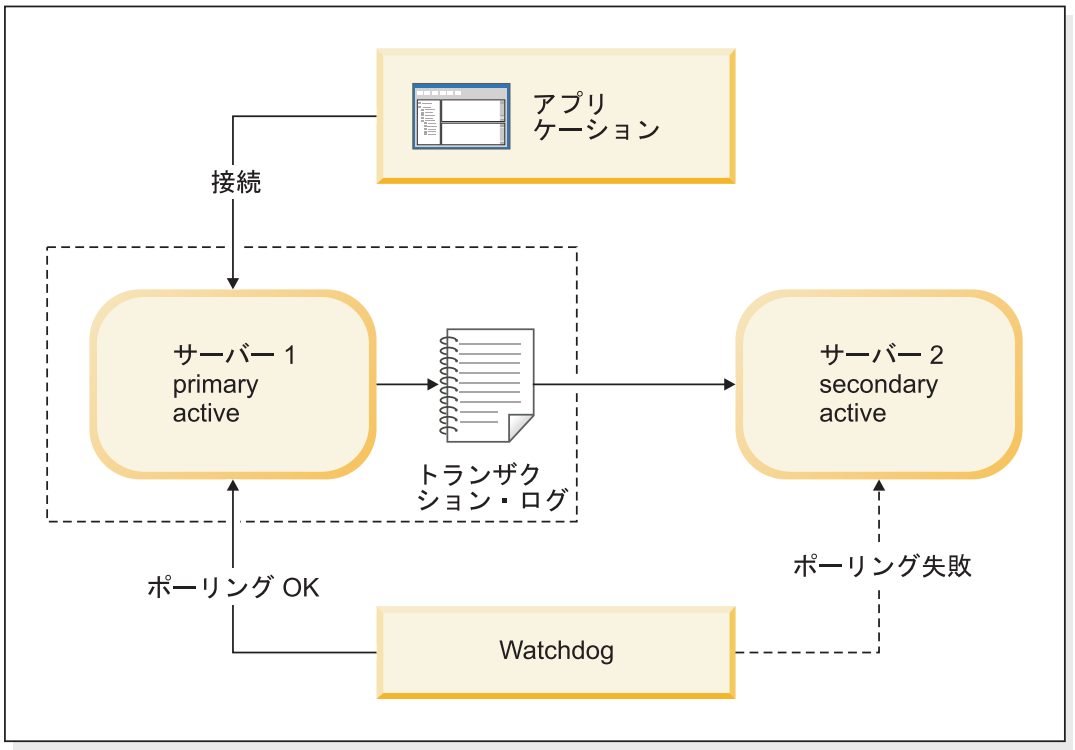
シナリオ

Watchdog と 2 次サーバーの間の接続が切断されました。

修復方法

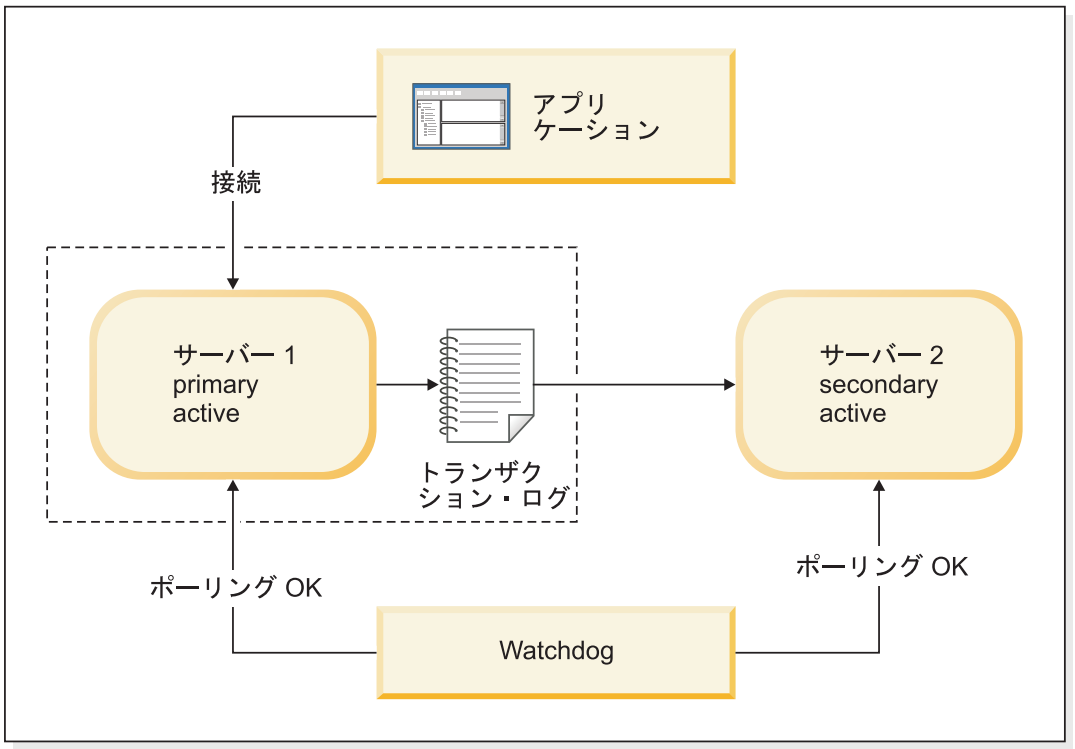
1 次サーバーおよび 2 次サーバーは、Watchdog から 2 次サーバーへのリンクがダウンした場合でも、操作を続行できます。Watchdog から 2 次サーバーへのリンクが修正された時点で、必ず 1 次サーバーと 2 次サーバーを検査して、それらの状態を確認してください。

シナリオ



解決方法

1



1. サーバー 2 との Watchdog のネットワーク接続を修正します。

図 26. Watchdog と 2 次サーバーの間のリンクが切断されたシナリオと修復方法

症状

Watchdog のポーリングが 2 次サーバーで失敗します。

追加シナリオ

各サーバーの状態が変更され、1 つのサーバーがもはや機能しなくなった場合は、このセクションにある該当するシナリオの説明を参照してください。

Watchdog と 2 次サーバーの間の通信リンクがダウンした場合のリカバリーの方法

このタスクについて

Watchdog と 2 次サーバーの間の接続が切断されたシナリオからリカバリーを行うには、以下の手順を実行します。

手順

1. 1 次サーバーと 2 次サーバーが通常操作を続行できるようにします。
2. Watchdog と 2 次サーバーの間のネットワーク接続を修正します。
3. ネットワークが接続されたら、以下のコマンドで Watchdog に各サーバーの状態を検査させます。

   ```
   ADMIN COMMAND 'hotstandby state';
   ```

F.2.7 Watchdog と 1 次サーバーの間、および 1 次サーバーと 2 次サーバーの間の通信リンクがダウンした場合

シナリオ

Watchdog と 1 次サーバーの間、および 1 次サーバーと 2 次サーバーの間の接続が切断されました。

修復方法

Watchdog が 1 次サーバーのモニターを続行するためには、2 次サーバーを新規 1 次サーバーになるように切り替え、その新規 1 次サーバーを PRIMARY ALONE 状態に設定します。後で、新規 2 次サーバーをセットアップし、1 次サーバーと同期させます。

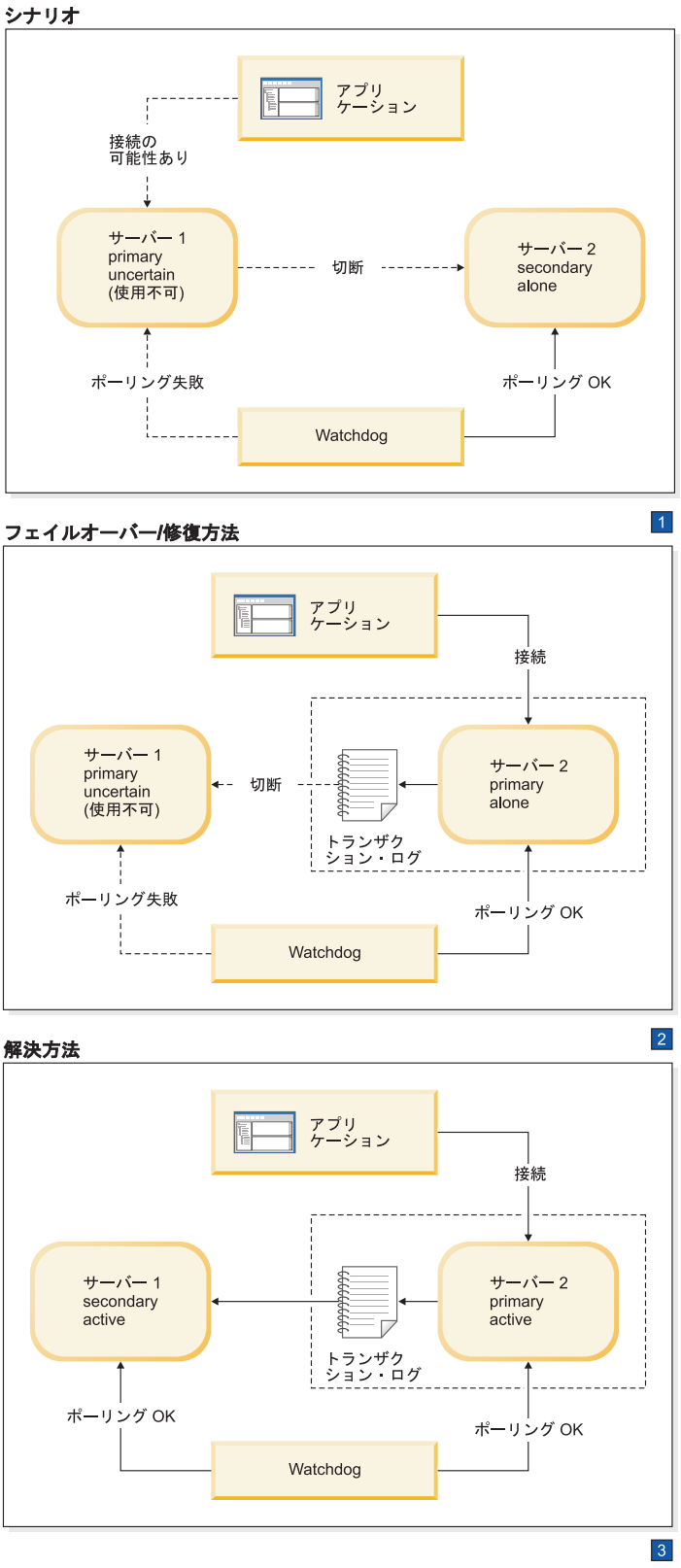


図 27. Watchdog と 1 次サーバーの間、および 1 次サーバーと 2 次サーバーの間のリンクが切断されたシナリオと修復方法

1. サーバー 1 のロールは Primary Uncertain です。ただし、Watchdog から見たサーバー 1 は使用不可であり、Primary Uncertain ではありません。

   Watchdog がサーバー 2 に指示します。

   HSB SET PRIMARY ALONE

   アプリケーションがサーバー 1 からサーバー 2 に切り替えます。
2. 両方のサーバーが自身を 1 次サーバーと認識しています。プログラムまたは管理者 (手操作による介入) がサーバー 1 を Primary Uncertain から Primary Alone に切り替えると、アクティブな 1 次サーバーが 2 つ存在することになり、両方がデータを更新する可能性があり、その差違は解決不可能です。

   ネットワーク接続が修正された後に、Watchdog がサーバー 1 に指示します。

   HSB SWITCH SECONDARY

   Watchdog がサーバー 2 に指示します。

   HSB COPY または

   NETCOPY HSB CONNECT
3. トランザクション・ログが満杯である場合は、PRIMARY ALONE サーバーを STANDALONE に切り替えなければならないことがあります。その場合は、サーバーを再接続する前に HSB COPY または HSB NETCOPY を実行する必要もあります。トランザクション・ログが満杯でない場合は、COPY/NETCOPY コマンドをスキップする必要があります。

症状

Watchdog のポーリングが 1 次サーバーで失敗します。2 次サーバーおよび 1 次サーバーが相互に相手との接続を失います。このため、サーバー 2 は SECONDARY ALONE 状態になり、1 次サーバーは (接続できる場合に) その状態が PRIMARY UNCERTAIN または PRIMARY ALONE であることを報告します。

このシナリオの始めには、アプリケーションが旧 1 次サーバーに接続している可能性もあることを想定しています。しかし、旧 1 次サーバーが PRIMARY UNCERTAIN 状態にあるので、アプリケーションは更新を行うことができません。サーバー 1 に接続しているアプリケーションが通信リンクを失い、もはや旧 1 次サーバーの存在を認識できなくなっている可能性もあることに注意してください。

Watchdog と 1 次サーバーの間、および 1 次サーバーと 2 次サーバーの間の通信リンクがダウンした場合のリカバリーの方法

Watchdog と 1 次サーバーの間、および 1 次サーバーと 2 次サーバーの間の接続が切断されたシナリオからリカバリーを行うには、ホット・スタンバイ・サーバー (2 次サーバー) を 1 次サーバーに置き換えるために必要な手順を実行します。

このタスクについて

2 次サーバーが 1 次サーバーに置き換わることができるようにするには、以下の手順を実行します。

手順

1. 1 次サーバーが PRIMARY UNCERTAIN 状態にあるか、2 次サーバーおよびアプリケーションから切り離されている場合は、以下のコマンドを使用して、2 次サーバーを PRIMARY ALONE 状態に設定します。

   ```
   ADMIN COMMAND 'hotstandby set primary alone';
   ```

2. アプリケーションを新規 1 次サーバーに再接続します。
3. 旧 1 次サーバーへのネットワーク接続または切断された接続を修正します。
4. サーバーの状態を検査します。この時点で、両方のサーバーが稼働している必要があります。
5. 新規 1 次サーバーが (例えば、接続を修正中に新規 1 次サーバーのトランザクション・ログが満杯になったために) STANDALONE 状態にある場合は、以下の手順を実行します。
   a. 以下のコマンドを使用して、新規 1 次サーバーを PRIMARY ALONE 状態に設定します。

      ```
      ADMIN COMMAND 'hotstandby set primary alone';
      ```

   b. データベースを新規 1 次サーバーから新規 2 次サーバーにコピーします。詳細については、56 ページの『3.4.5, 1 次サーバーと 2 次サーバーの同期』を参照してください。
6. 新規 1 次サーバーが PRIMARY ALONE 状態にある場合は、以下のようにします。
   a. 以下のコマンドを使用して、旧 1 次サーバーが新規 2 次サーバーになるように切り替えます。

      ```
      ADMIN COMMAND 'hotstandby switch secondary';
      ```

7. 以下のコマンドを使用して、新規 1 次サーバーを新規 2 次サーバーに再接続します。

   ```
   ADMIN COMMAND 'hotstandby connect';
   ```

Watchdog と 1 次サーバーの間、および 1 次サーバーと 2 次サーバーの間の通信リンクがダウンした場合の追加シナリオ

アプリケーションが新規 1 次サーバーからエラー・メッセージ 10047 または 14537 を受け取った場合は、以下のようにします。

- 2 次サーバーが新規 1 次サーバーになるように切り替えられたかどうかを検査するために、旧 2 次サーバーに接続を試みます。
- 旧 2 次サーバーが新規 1 次サーバーになるように切り替えられていない場合は、元の 1 次サーバーを PRIMARY ALONE 状態にして、トランザクションを再実行します。

F.2.8 Watchdog と 2 次サーバーの間、および 1 次サーバーと 2 次サーバーの間の通信リンクがダウンした場合

シナリオ

Watchdog と 2 次サーバーの間の接続、および 1 次サーバーと 2 次サーバーの間の接続が切断されました。

修復方法

1 次サーバーは、2 次サーバーおよび Watchdog へのリンクがダウンした場合でも、操作を続行できます。1 次サーバーを PRIMARY ALONE 状態に切り替えます (まだ PRIMARY ALONE 状態でない場合)。後で 2 次サーバーが再び稼働したときに、1 次サーバーとの同期をとります。

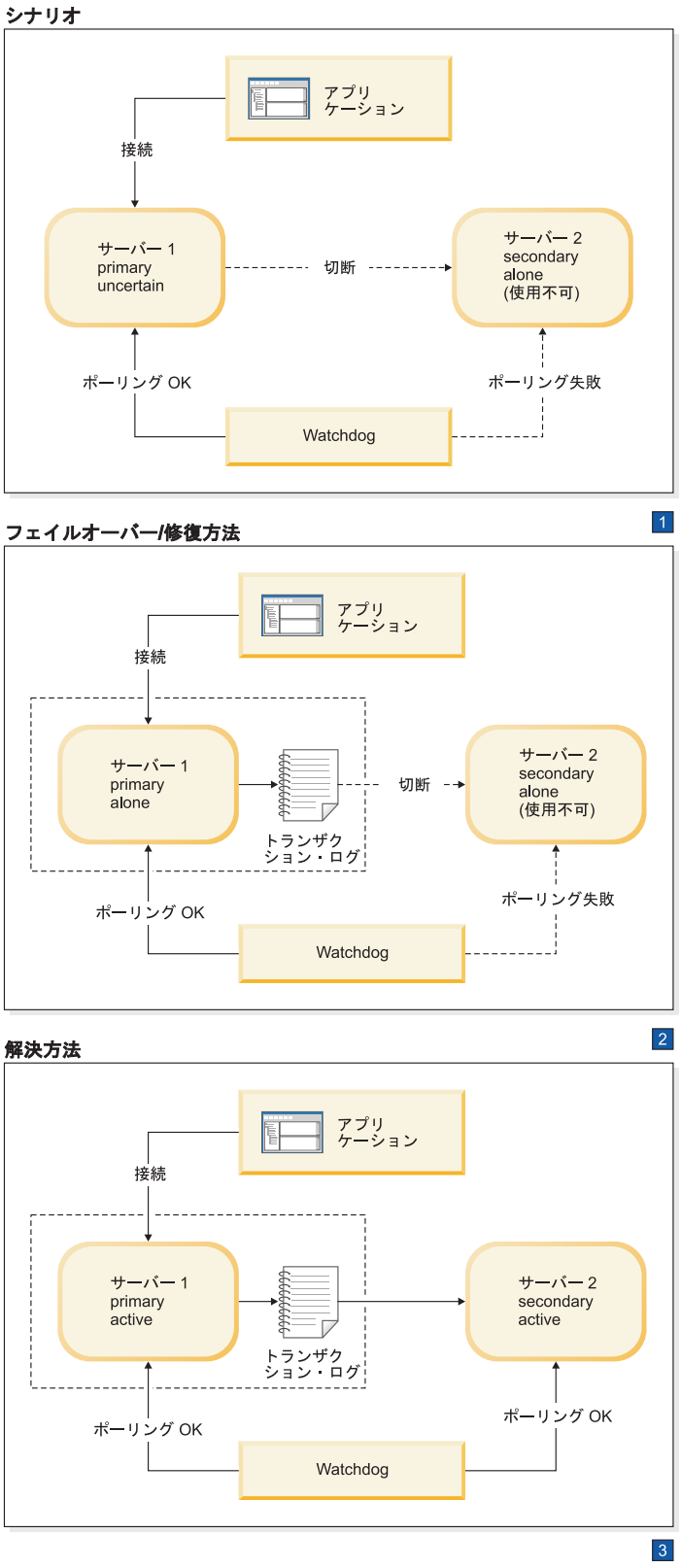


図 28. Watchdog と 2 次サーバーの間、および 1 次サーバーと 2 次サーバーの間のリンクが切断されたシナリオと修復方法

1. サーバー 2 は自身のロールを Secondary Alone と見なしますが、Watchdog はサーバー 2 を認識できないため、サーバー 2 を使用不可と見なします。

   Watchdog がサーバー 1 に指示します。

   HSB SET PRIMARY ALONE
2. 2 次サーバーとの接続が修正された後に、Watchdog がサーバー 1 に指示します。

   HSB COPY

   NETCOPY HSB CONNECT
3. トランザクション・ログが満杯である場合は、PRIMARY ALONE サーバーを STANDALONE に切り替えなければならないことがあります。その場合は、サーバーを再接続する前に HSB COPY または HSB NETCOPY を実行する必要もあります。トランザクション・ログが満杯でない場合は、COPY/NETCOPY コマンドをスキップする必要があります。

症状

Watchdog によるポーリングは 2 次サーバーで失敗し、1 次サーバーは 2 次サーバーと接続しておらず、状態が PRIMARY UNCERTAIN または PRIMARY ALONE に切り替えられます。

Watchdog と 2 次サーバーの間、および 1 次サーバーと 2 次サーバーの間の通信リンクがダウンした場合のリカバリーの方法

このタスクについて

Watchdog と 2 次サーバーの間の接続、および 1 次サーバーと 2 次サーバーの間の接続が切断されたシナリオからリカバリーを行うには、以下の手順を実行します。

手順

1. 接続の修正を試みます。
2. 接続が修正された後、コマンド **ADMIN COMMAND 'hotstandby state'** を使用して 1 次サーバーの状態を検査します。
3. 1 次サーバーの状態が STANDALONE の場合は、以下の手順を実行します。
   a. 両方のサーバーが稼働していることを確認します。
   b. 以下のコマンドを使用して、1 次サーバーの状態を PRIMARY ALONE に設定します。

      ```
      ADMIN COMMAND 'hotstandby set primary alone';
      ```
   c. 以下のコマンドを使用して、データベースを 1 次サーバーから 2 次サーバーにコピーします。

      ```
      ADMIN COMMAND 'hotstandby netcopy';
      ```

      詳細については、 56 ページの『3.4.5, 1 次サーバーと 2 次サーバーの同期』を参照してください。
4. 以下のコマンドを使用して、1 次サーバーを 2 次サーバーに再接続します。

```
ADMIN COMMAND 'hotstandby connect';
```

Watchdog と 2 次サーバーの間、および 1 次サーバーと 2 次サーバーの間の通信リンクがダウンした場合の追加シナリオ

アプリケーションが 1 次サーバーからエラー・メッセージ 10047 または 14537 を受け取った場合は、以下のようにします。

- 2 次サーバーが 1 次サーバーになるように切り替えられたかどうかを検査するために、2 次サーバーに接続を試みます。
- 2 次サーバーが新規 1 次サーバーとして切り替えられていない場合は、当初の 1 次サーバーを PRIMARY ALONE 状態にして、トランザクションを再実行します。

F.3 solid.ini 構成ファイルの Watchdog セクション

Watchdog 用の solid.ini ファイルにある [Watchdog] 構成セクションは、Watchdog 固有のパラメーターを指定するための場所です。

重要: solid.ini ファイルの [Watchdog] セクションのパラメーターは、solidDB によってすべてが事前定義されるわけではありません。どのように Watchdog を書いたか、およびそこで solid.ini ファイルからパラメーター情報を読み取るかどうかに応じて、ここで定義するパラメーターと定義済みのパラメーターを任意に組み合わせて使用できます。パラメーターを無視することもできます。ここに示したパラメーターは、solidDB で提供されるサンプルの C 言語 Watchdog プログラム用のものです。

表 41. Watchdog パラメーター

[Watchdog]	説明	ファクトリー値

AutoSwitch	**AutoSwitch** パラメーターを yes に設定すると、Watchdog が自動的に以下を行います。
	1. 2 次サーバーで障害が起きた場合は、Watchdog が 1 次サーバーに、PRIMARY UNCERTAIN 状態を継続するのではなく、PRIMARY ALONE 状態へ切り替えるよう指示します。
	2. 1 次サーバーで障害が起きた場合は、Watchdog が自動的に以下のコマンドを送信します。
	`'hsb switch primary'`
	`'hsb set primary alone'`
	その目的は、元の 2 次側を新規の 1 次側にすることです。
	例えば、以下のように指定します。
	`[Watchdog]`
	`AutoSwitch = NO`
	このパラメーターはオプションです。
		Yes

表 41. *Watchdog パラメーター (続き)*

[Watchdog]	説明	ファクトリー値
Connect1	[Watchdog] セクションの **Connect1** パラメーターを使用すると、Watchdog アプリケーションは、1 次サーバーまたは 2 次サーバーに接続できます。これは、**Connect1** サーバーのプロトコルおよびネットワーク・アドレスを定義する必須パラメーターです。 以下に例を示します。 `connect1 = tcp primarymachine 1313`	なし
Connect2	[Watchdog] セクションの **Connect2** パラメーターを使用すると、Watchdog アプリケーションは、1 次サーバーまたは 2 次サーバーに接続できます。これは、**Connect2** サーバーのプロトコルおよびネットワーク・アドレスを定義する必須パラメーターです。 以下に例を示します。 `connect2 = tcp secondarymachine 1313`	なし
DualSecAutoSwitch	**DualSecAutoSwitch** = Yes で、両方のサーバーが 2 次サーバーである場合、Watchdog はその 2 つの 2 次サーバーのうちの 1 つを新規 1 次サーバーとして自動的に選択し、それを 1 次サーバーに切り替えます。**DualSecAutoSwitch** = No の場合は、システム管理者が一方のサーバーを 1 次サーバーに切り替える必要があります。 **DualSecAutoSwitch** は、Watchdog が「通常」モードであっても「障害」モードであっても適用されることに注意してください。	Yes

NumRetry	[Watchdog] セクションの **NumRetry** パラメーターを使用すると、Watchdog が 2 次または 1 次サーバーへの接続試行を何回行った後で、接続試行を応答障害またはエラーであると見なすかの回数を指定できます。
	
	例えば、以下のように指定します。
	`[Watchdog]`
	`NumRetry = 3`
	
	再試行は元の試行に加える回数です。再試行回数を 3 に設定した場合、試行の総数は 4 です。再試行はすぐに行われることに注意してください。Watchdog は障害があるときには再試行の間で (**PingTimeout** などの) 時間間隔を待機しません。
	
	このパラメーターはオプションです。
		0

Password1 Password2	下の **Username1** および **Username2** パラメーターの説明を参照してください。	ファクトリー値なし

表 41. *Watchdog パラメーター (続き)*

[Watchdog]	説明	ファクトリー値
Pessimistic	このパラメーターを Yes に設定すると、Watchdog の反応を高速化できます。 **Pessimistic** = No の場合、Watchdog はサーバーとの接続を検査しますが、いずれかのサーバーが問題の存在を検出してその状態を (例えば、PRIMARY UNCERTAIN へ) 変更するまで、実際には動作しません (例えば、サーバーの状態を PRIMARY ALONE へ変更しません)。 **Pessimistic** = Yes の場合、Watchdog は、それ自体といずれかのサーバーとの接続が失われるとすぐに動作します。つまり、Watchdog は、残りのサーバーが状態を変更するのを待機しません。これにより反応時間の高速化が可能ですが、例えば、ネットワークの問題に起因する誤報の可能性が高くなることもあります。 **Pessimistic** = Yes の場合、Watchdog は以下のように反応します。1 次サーバーとの接続が失われると、Watchdog は 2 次サーバーを 1 次サーバーに切り替えます。2 次サーバーとの接続が失われると、Watchdog は 1 次サーバーを PRIMARY ALONE に設定します。 **注意:** **Pessimistic = Yes に設定すると、余分な切り替えが生じたり、1 次側が二重になることさえ場合によってはあります。このパラメーターは、サーバーに比べ障害が起きそうにもないネットワークでなければ、Yes には設定しないでください。** オプションのコマンド行スイッチ「**-p**」を使用することにより、ペシミスティック動作をオンにすることもできます。	No

PingInterval	[Watchdog] セクションの **PingInterval** パラメーターを使用すると、通常の Watchdog モードで接続状況情報の照会を行うミリ秒単位の間隔を指定できます。サーバーの障害を検出するために、Watchdog は hotstandby status connect コマンドを、毎回の **PingInterval** ミリ秒の後で、1 次サーバーと 2 次サーバーの両方に送信します。
	
	以下に例を示します。
	`[Watchdog]`
	`PingInterval = 5000`
	
	このパラメーターはオプションです。
	
	Watchdog 用の **PingInterval** パラメーターは、サーバー用の **PingTimeout** パラメーターとは異なることに注意してください。
	
	注意:
	以前のサンプル Watchdog では、PingInterval をミリ秒単位ではなく秒単位で指定する必要がありました。古い solid.ini ファイルを使用している場合は、それを更新してください。
		1000
		
		(1 秒)

表 41. *Watchdog* パラメーター *(続き)*

[Watchdog]	説明	ファクトリー値

Username1 Username2	[Watchdog] セクションの **Username** および **Password** パラメーターはオプションです。それらは、connect1 サーバーの使用が許可されるユーザー名とパスワードを指定します。
	
	例えば、以下のように指定します。
	`[Watchdog]`
	`Username1 = Tom`
	`Password1 = dr17xy`
	`Username2 = Jerry`
	`Password2 = M89tvt`
	
	セキュリティー上の理由から、これらのパラメーターが solid.ini 構成ファイルに指定されていない場合、Watchdog は開始時にそれらを求めるプロンプトを出します。
		ファクトリー値なし。

WatchdogLog	[Watchdog] セクションの **WatchdogLog** パラメーターを使用すると、Watchdog ログのファイル名を指定できます。Watchdog ログは、現行作業ディレクトリーに作成されます。これは、Watchdog のメッセージを記録して、Watchdog コマンド発行の必要性を管理者に警告するために使用します。
	
	例えば、以下のように指定します。
	`[Watchdog]`
	`WatchdogLog = Watchdog.log`
	
	空白または特定の句読記号など、特殊文字を含まない限り、ファイル名を引用符で囲む必要はありません。
	
	このパラメーターはオプションです。
		`Watchdog.log`

以下のパラメーターを使用した場合、

```
[Logging]
DurabilityLevel
```

DurabilityLevel パラメーター値は 1 次サーバーのみに影響します。2 次サーバーのロギング・モードは、[HotStandby] セクションの **2SafeAckPolicy** パラメーターに従います。

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]

[ラ行]

[数字]

A

B

C

D

E

G

H

L

M

N

O

P

R

S

T

U

W

[特殊文字]

特記事項

International Business Machines Corporation の書面による明示的な許可がある場合を除き、本製品のいかなる部分も、いかなる方法においても使用することはできません。

本製品は、米国特許 6144941、7136912、6970876、7139775、6978396、7266702、7406489、7502796、および 7587429 により保護されています。

本製品は、米国輸出規制品目分類番号 ECCN=5D992b に指定されています。

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。

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。

〒242-8502
神奈川県大和市下鶴間1623番14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
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

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的創作物にも、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.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

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 Corp. 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。

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。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

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。

Printed in Japan

SA88-4580-00

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