IBM solidDB
IBM solidDB Universal Cache

バージョン 6.3

リンク・ライブラリー・アクセス・ユーザー・ガイド

Information Management software

SC88-5819-00
(英文原典：SC23-9831-00)

IBM solidDB
IBM solidDB Universal Cache

バージョン 6.3

リンク・ライブラリー・アクセス・ユーザー・ガイド

Information Management software

SC88-5819-00
(英文原典：SC23-9831-00)

注記

本書および本書で紹介する製品をご使用になる前に、 71 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 6、リリース 3 の IBM solidDB (プロダクト番号 5724-V17) および IBM solidDB Universal Cache (プロダクト番号 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： SC23-9831-00
IBM solidDB
IBM solidDB Universal Cache
Version 6.3
Linked Library Access User Guide

発行： 日本アイ・ビー・エム株式会社

担当： ナショナル・ランゲージ・サポート

第1刷 2009.2

目次

図

表

本書について

IBM® solidDB® リンク・ライブラリー・アクセスは、solidDB データ管理ソリューションの高性能バージョンです。ネットワーク遅延を防止するため、solidDB 実行可能プログラムとユーザー・アプリケーションは、同じプログラム・スペースでリンクされ、単一の実行可能プログラムが生成されます。ネットワーク接続とリモート・プロシージャー・コール (RPC) をローカル関数呼び出しで置き換えることにより、パフォーマンスが大きく向上します。

本書には、リンク・ライブラリー・アクセスに固有の情報が記載されています。本書は、solidDB の管理と保守について詳細に説明している「*IBM solidDB 管理者ガイド*」に含まれている情報を補足するものです。

本書は、C プログラミング言語に関する実用的な知識、一般的な DBMS の知識、SQL に精通していること、および solidDB インメモリー・データベースや solidDB ディスク・ベース・エンジンなどの solidDB データ管理製品に関する知識を前提としています。solidDB Java™ アクセラレーターで作業を行う場合には、Java に関する実用的な知識も前提としています。

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 1. 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 1. 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	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 2. 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 リンク・ライブラリー・アクセスの紹介

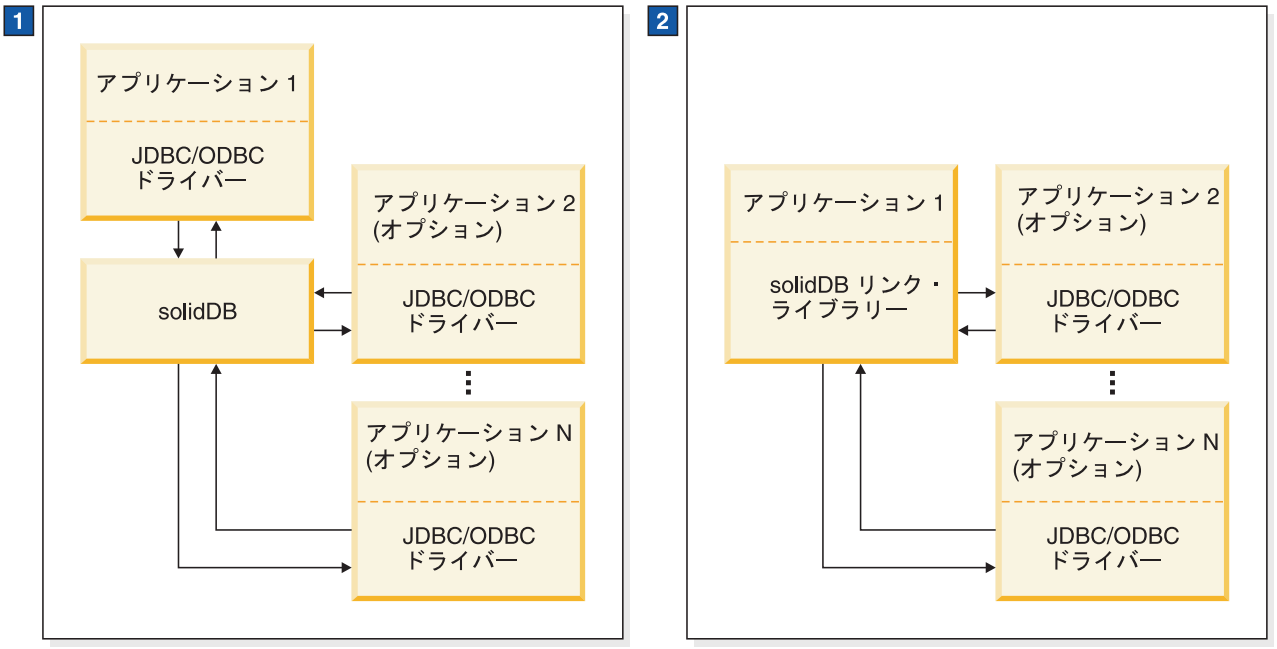
IBM® solidDB リンク・ライブラリー・アクセスは、 solidDB で使用可能な機能およびインターフェースと同じものを提供する関数ライブラリーです。このライブラリーにユーザー・アプリケーションをリンクすることができます。リンクされたアプリケーションは、直接関数呼び出しを使用してサーバーと通信するので、クライアントとサーバーが TCP/IP などのネットワーク・プロトコルを通じて通信する際に必要となるオーバーヘッドがかかりません。アプリケーションとサーバーをリンクして 1 つの実行可能プログラムにすることで、パフォーマンスが向上します。

リンク・ライブラリー・アクセス・ライブラリーを使用するためにアプリケーションを書き直す必要はありません。例えば、独自の関数を呼び出す必要はありません (データベース・サーバーを始動および停止するための関数を除く) 。アプリケーションでは、それまで使用していた ODBC 関数呼び出しを引き続き使用できます。アプリケーションにリンク・ライブラリー・アクセス・ライブラリーがリンクされると、それらの ODBC 関数呼び出しがネットワークを介さずにサーバーに直接送信されるようになります。

アプリケーションでは、サーバー内でタスクのスケジューリングなどを行うその他のリンク・ライブラリー・アクセス関数呼び出しにもアクセスできます。ただし、そのような関数呼び出しは必要でない限り使用する必要はありません。

サーバーにリンクされたアプリケーションだけが、そのサーバーを使用できるわけではありません。リンク・ライブラリー・アクセス関数ライブラリーとして実行される solidDB サーバーには、ローカルのクライアント・アプリケーション (ライブラリーに直接リンクされたアプリケーション) だけでなく、リモートのクライアント・アプリケーション (TCP/IP などの通信プロトコルを通じてサーバーに接続するアプリケーション) もアクセスできます。リモート・クライアントから見たリンク・ライブラリー・アクセス・サーバーは、他の solidDB サーバーとほとんど変わりありませんが、ローカル・クライアントから見たリンク・ライブラリー・アクセス・サーバーは、他の solidDB サーバーよりも高速で、詳細な制御が可能です。

注: リモート・アプリケーションは、通常はサーバーが稼働するコンピューターとは別のコンピューターで実行されますが、データベース・サーバーと同じコンピューターで実行されているアプリケーションでも、ネットワーク通信プロトコルを通じてサーバーと通信していれば「リモート」と見なされます。



1. 標準 solidDB データベース構成では、アプリケーションとサーバーは別々のプログラムです。
2. solidDB リンク・ライブラリーは、アプリケーションへリンクされるサブルーチンのライブラリーです。その他のアプリケーションもサーバーと通信できます。

図 1. リンク・ライブラリー・アクセスを使用する solidDB

上の図は、リンク・ライブラリー・アクセス・ライブラリーを使用する solidDB の例を示しています。

注:

ローカル・アプリケーション要求は、solidDB SA API または ODBC API の直接関数呼び出しによって処理されます。ローカル・アプリケーションには、リンク・ライブラリー・アクセスに、solidDB のバックグラウンド・プロセスとクライアント・タスクを制御するためのローカル要求を処理する制御 API も用意されています。リンク・ライブラリー・アクセスで JDBC 呼び出しを使用することもできます。詳しくは、55 ページの『5 章 Java での solidDB リンク・ライブラリー・アクセスの使用』を参照してください。

この図からわかるように、リモート・クライアントは、クライアント・アプリケーションにリンクされた ODBC ドライバーまたは JDBC ドライバーを介して通信しますが、ローカル・クライアント・アプリケーションはリモート通信ドライバーをいっさい必要としません。

リンク・ライブラリー・アクセス・ライブラリー

標準の (リンク・ライブラリー・アクセスを使用しない) solidDB 構成では、アプリケーション (クライアント) とデータベース・エンジン (サーバー) が別々のプロセスとしてネットワーク・プロトコルを介して通信します。クライアントは、ネットワークを介してデータベース・サーバーと通信する通信ドライバー (ODBC ドライバーや JDBC ドライバーなど) にリンクする必要があります。

リンク・ライブラリー・アクセスを使用する場合は、完全なデータベース・サーバーの機能を備えた静的ライブラリー (.lib や UNIX の .a など) にアプリケーションがリンクします。つまり、solidDB がアプリケーションと同じ実行可能プログラムで実行されるので、ネットワークを介してデータを転送する必要がありません。リンク・ライブラリー・アクセス・ライブラリーにリンクしているアプリケーションは、ODBC API と SA API の両方を使用して複数の接続を作成することも可能です。これらの API はどちらも再入可能なので、別々のスレッドからの同時接続が可能となります。

リンク・ライブラリー・アクセス・ライブラリーに直接リンクしているユーザー・アプリケーションでは、他のデータベース・サーバーとのリモート接続を作成することもできます。接続タイプがローカルとリモートのどちらであるかは、ODBC API または SA API の接続関数に渡される接続ストリングで定義されます。

アプリケーションのリンクについて詳しくは、 12 ページの『リンク・ライブラリー・アクセスへのアプリケーションのリンク』を参照してください。

アプリケーションを開始すると、アプリケーション内のコードのみが自動的に実行を開始します。サーバー・コードは大部分がアプリケーション・コードとは独立しており、関数を呼び出してサーバーを明示的に始動する必要があります。(ほとんどのインプリメンテーションで、サーバーはアプリケーションが使用するスレッドとは別のスレッドで実行されます。関数を呼び出してサーバーを始動すると、サーバー・コードに必要な初期化ステップが実行され、必要に応じて適切なスレッドが追加で作成され、そのスレッドでサーバーの実行が開始されます。)

ディスク・ベース・サーバーとディスクレス・サーバー

リンク・ライブラリー・アクセス・ライブラリーには、サーバーを始動する関数呼び出しが 2 種類あります。一方の関数呼び出しでは通常の (つまりディスク・ベースの) サーバーを始動し、もう一方の関数呼び出しではディスク・ドライブを使用しないサーバーを始動します。詳しくは、 51 ページの『4 章 ディスクレス機能の使用』および SSCStartServer 関数と SSCStartDisklessServer 関数の説明を参照してください。

ライブラリーの内容

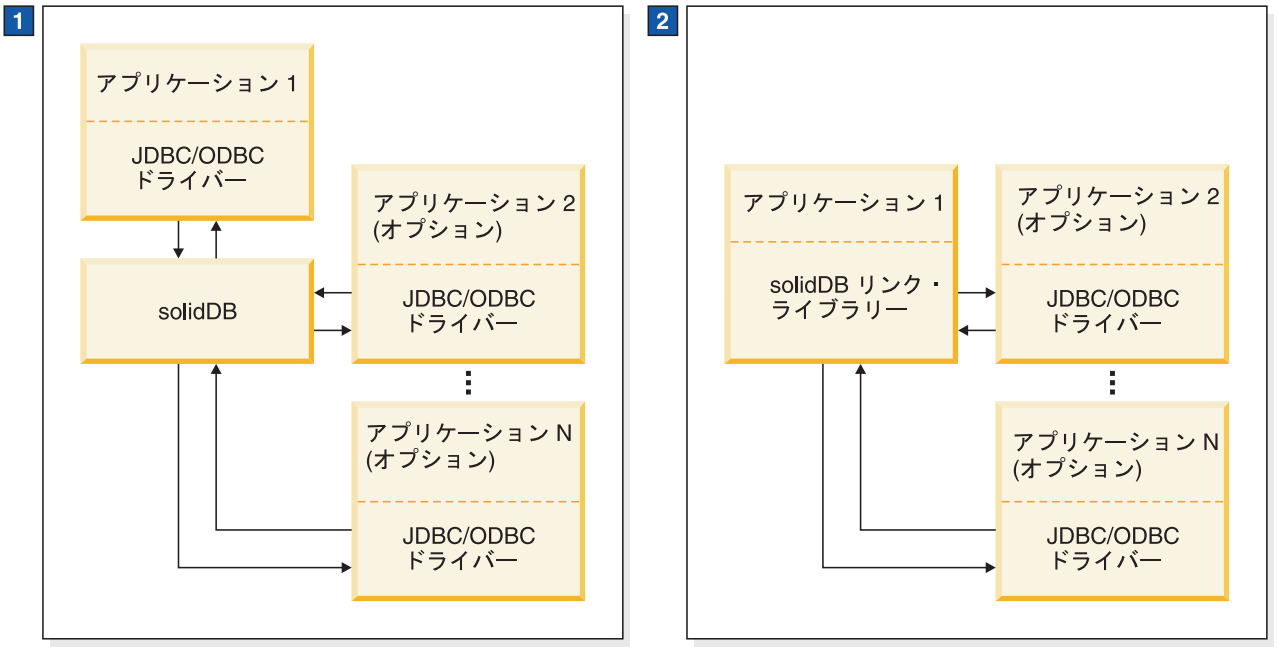
リンク・ライブラリー・アクセス・ライブラリーには、3 つの異なる API の関数が含まれています。

- タスクのスケジューリングを制御する関数で構成される solidDB 制御 API (SSC API) ライブラリー
- ネットワークを介さずにサーバー・ライブラリーと直接通信できるようにする solidDB ODBC ドライバー関数
- リンク・ライブラリー・アクセスを使用するその他の機能に必要となる solidDB SA API ライブラリー。例えば、このライブラリーを使用して、表におけるレコードの挿入、削除、選択を実行できます。

この 3 つの API (SSC、SA、および ODBC) をすべて含んだライブラリーにアプリケーションがリンクされるので、アプリケーション・プログラムからこれらの API を任意に組み合わせて関数を呼び出すことができます。それぞれの API について詳

しくは、 6 ページの『solidDB のリンク・ライブラリー・アクセス用クライアント API とドライバー』を参照してください。

注: リモート・アプリケーションもこの 3 つの API (SSC、SA、および ODBC) にアクセスできます。ただし、リモート・アプリケーションの場合は、この 3 つの API の関数すべてを 1 つにまとめたファイルがありません。リモート・アプリケーションおよび二重の役割を持つアプリケーションについて詳しくは、 『リンク・ライブラリー・アクセスで使用されるアプリケーション・タイプ』を参照してください。リモート・アプリケーション用の API ファイルについては、 6 ページの『solidDB のリンク・ライブラリー・アクセス用クライアント API とドライバー』を参照してください。



1. 標準 solidDB データベース構成では、アプリケーションとサーバーは別々のプログラムです。
2. solidDB リンク・ライブラリーは、アプリケーションへリンクされるサブルーチンのライブラリーです。その他のアプリケーションもサーバーと通信できます。

図 2. solidDB へのリンク

リンク・ライブラリー・アクセスで使用されるアプリケーション・タイプ

リンク・ライブラリー・アクセス・アプリケーションは、サーバーにとって「ローカル」です。つまり、サーバーとアプリケーションが結合されて 1 つのプログラムになります。ODBC 関数の呼び出しは、実際には ODBC ドライバーと通信プロトコル (TCP/IP など) を経由せずにサーバーに直接送信されます。

サーバーは、リンク・ライブラリー・アクセス・ライブラリーにリンクされているローカル・アプリケーションからの要求を処理するだけでなく、リモート・アプリケーションからの要求も処理します。

リモート・アプリケーションはリンク・ライブラリー・アクセス・ライブラリーにリンクされていません。このようなアプリケーションは独立した実行可能プログラムであり、ネットワーク接続 (TCP/IP など) またはその他の接続 (共用メモリーなど) を使用してサーバーと通信する必要があります。リモート・アプリケーションは、常時ではありませんが通常は、サーバーを実行するコンピューターとは別のコンピューターで実行されます。ただし、1 台のコンピューターで、リンク・ライブラリー・アクセスのローカル・アプリケーションを実行しながら 1 つ以上のリモート・アプリケーションを別々のプロセスとして実行することができます。

ほとんどのアプリケーションは、ローカル (1 つの実行可能プログラム内でリンク・ライブラリー・アクセス・ライブラリーにリンクされる) またはリモート (リンク・ライブラリー・アクセス・ライブラリーにリンクされない) のいずれかです。ただし、ローカルにもリモートにもなるアプリケーションを作成することも可能です。この場合、アプリケーションのコンパイルおよびリンク方法に応じてモードが切り替わります。このような二重モード・アプリケーション では、例えば同じ C 言語アプリケーション・コードをローカル・モードまたはリモート・モードで使用しますが、それぞれのモードで異なるライブラリーにリンクされます。

リンク・ライブラリー・アクセスでの二重モード・アプリケーションの使用

例えばリンク・ライブラリー・アクセスの場合は、二重モード・アプリケーションをローカルで実行するときに、そのアプリケーションをリンク・ライブラリー・アクセス・ライブラリーにリンクする必要があります。一方、二重モード・アプリケーションをリモートで実行するときは、このアプリケーションをリンク・ライブラリー・アクセスの制御 API スタブ・ライブラリー (例えば Windows での solidctrlstub.lib) にリンクする必要があります。これにより、このアプリケーションをコンパイル、リンク、および実行するときにリンク時エラーを回避できます。

リモート・アプリケーションには「制御 API スタブ・ライブラリー」が必要です。これは、リンク・ライブラリー・アクセス専用の制御 API (ローカルのリンク・ライブラリー・アクセス・ライブラリーで提供) をリモート・アプリケーションで使用できないためです。例えば、標準の ODBC ライブラリーにリンクするローカル・アプリケーション (制御 API 関数を使用) があるとします。このアプリケーションをリモートでも実行するとします。制御 API スタブ・ライブラリーにリンクすることで、コードから制御 API 関数呼び出しを削除する必要がなくなります。このように、リンク・ライブラリー・アクセス・ローカル・アプリケーションを、通常のリモート・クライアント・アプリケーションに簡単に変えることができます。

注:

制御 API スタブ・ライブラリーには、「何もしない」関数が含まれています。この関数をリモート・アプリケーションで呼び出しても、サーバーには影響しません。

二重モード・アプリケーションが役立つ理由はほかにもあります。

- ローカル・アプリケーションをリンク・ライブラリー・アクセス・ライブラリーにリンクする前にテストできます。
-

すべてのユーザー/プロセスで、それがローカルかリモートかにかかわらず、同じアプリケーション・ロジックを使用できます。

二重モード・アプリケーション

2 人のユーザーが同じアプリケーションを実行しているとします。ユーザー 1 はアプリケーションをローカルで実行します (これによりパフォーマンスが向上します)。ユーザー 2 は、同じアプリケーションをリモートで実行します。

ユーザー 1 (ローカル・ユーザー) は、サーバー・ライブラリー (solidac.a など) でコンパイルとリンクを行い、リンク・ライブラリー・アクセスの制御 API を使用して、サーバーの始動と停止、およびその他のスケジューリング・タスクを実行します。ユーザー 2 (リモート・ユーザー) は同じアプリケーションを実行しますが、ユーザー 1 がサーバーを始動するまでサーバーに接続できません。したがって、ユーザー 1 だけがタスク処理システムを制御できます。

solidDB のリンク・ライブラリー・アクセス用クライアント API とドライバー

以下の項目では、リンク・ライブラリー・アクセスで使用可能な API について簡単に説明します。

注:

各説明で使用されている「ローカル」アプリケーションおよび「リモート」アプリケーションの定義は、4 ページの『リンク・ライブラリー・アクセスで使用されるアプリケーション・タイプ』で示されているとおりです。

solidDB SA API

SA API は、solidDB データ管理サービスに対するロー・レベル・プロプラエタリー C 言語 API です。この API はリンク・ライブラリー・アクセス・ライブラリー (Windows での ssolidacxx.dll や UNIX での solidac.a など) に含まれています。リンク・ライブラリー・アクセス・ライブラリーには、SA API 関数呼び出しを使用するローカル・アプリケーションをサポートする SA-API ライブラリーが含まれています。

SA API ライブラリーは solidDB 製品で内部的に使用され、 solidDB データベース表のデータにアクセスできるようにします。このライブラリーに含まれる 90 個の関数によって、データベースの接続およびカーソル・ベースの操作を実行するための低レベルのメカニズムが構成されます。 solidDB SA API を使用することで、パフォーマンスは大幅に向上します。例えば、SA API を使用してバッチ挿入操作のパフォーマンスを最適化できます。

リモート・アプリケーションについては、リンク・ライブラリー・アクセス・ライブラリーで SA API 関数呼び出しもサポートされます。ただし、別の SA API ライブラリー・ファイル (例えば Windows での solidimpsa.lib) にリンクする必要があります。

solidDB SA API について詳しくは、「*solidDB プログラマー・ガイド*」を参照してください。

solidDB ODBC API

solidDB ODBC API は、ローカルまたはリモートの solidDB データベースに SQL を使用してアクセスするための標準に準拠した手段です。この API が提供する関数では、データベース接続の制御、SQL ステートメントの実行、結果セットの取得、トランザクションのコミット、およびその他のデータ管理機能を実行できます。

solidDB データベースのコール・レベル・インターフェース (CLI) である ODBC API は、ANSI X3H2 SQL CLI に準拠しており、リンク・ライブラリー・アクセス・ライブラリー (Windows での ssolidacxx.dll や UNIX での solidac.a など) に組み込まれています。

リンク・ライブラリー・アクセスは、ODBC 3.51 標準をサポートしています。リンク・ライブラリー・アクセス・ライブラリーには solidDB ODBC 3.x が組み込まれており、これによってサーバーへの直接関数呼び出しを必要とするローカル・アプリケーションがサポートされます。

ローカル・アプリケーションに対しては、リンク・ライブラリー・アクセス・ライブラリーで ODBC 関数呼び出しがサポートされます。リモート・アプリケーション (またはリモートで実行される二重モード・アプリケーション) で同じ機能を実現するには、ODBC ドライバーにリンクする必要があります。

アプリケーションが二重モード・アプリケーション (ローカルでもリモートでも実行できるアプリケーション) で、かつリンク・ライブラリー・アクセスの制御 API と ODBC を使用する場合は、ローカルで実行する実行可能プログラムとリモートで実行する実行可能プログラムの 2 種類が必要です。ローカルで実行する場合は、アプリケーションをリンク・ライブラリー・アクセス・ライブラリーにリンクします。これにより、ODBC 関数と制御 API ライブラリーの両方がサポートされます。リモートで実行する場合は、アプリケーションを ODBC ドライバーと制御 API スタブ・ライブラリー (例えば Windows での solidctrlstub.lib) の両方にリンクする必要があります。このスタブ・ライブラリーは、実際にご使用のリモート・アプリケーションからサーバーの制御を可能にするものではなく、単に、「未解決シンボル」に関するエラーを出さずにプログラムのコンパイルおよびリンクを行えるようにするものです。

注:

ODBC 関数がリモートで呼び出されたときは (二重モード・アプリケーションで)、その呼び出しがネットワークを介してクライアントからサーバーに送信されます。ODBC 関数がローカルで呼び出されたときは (アクセラレーターを使用するアプリケーションで)、ODBC サブルーチン・ライブラリーがネットワークをバイパスしてローカル・アプリケーションをサーバーに直接接続します。

ODBC API について詳しくは、「*solidDB プログラマー・ガイド*」を参照してください。

solidDB JDBC API

JDBC API は、リモート・アプリケーションでのみ使用されます。この API は、JDK 1.2 のコア API として、データベース接続、SQL ステートメント、結果セット、データベース・メタデータなどを表す Java クラスを定義します。この API を

使用することで、SQL ステートメントを発行し、結果を処理できます。 JDBC は、Java でのデータベース・アクセスに使用される基本 API です。リンク・ライブラリー・アクセス・ライブラリーは、JDBC 1.x と 2.x の両方をサポートしています。詳しくは、「*solidDB プログラマー・ガイド*」を参照してください。

solidDB 制御 API (SSC API)

solidDB 制御 API (SSC API) は、solidDB データベース製品でのサーバーの動作を制御する C 言語のスレッド・セーフなインターフェースです。

制御 API は、リンク・ライブラリー・アクセス・ライブラリー (例えば Windows での ssolidacxx.dll や UNIX での solidac.a など) に含まれています。リンク・ライブラリー・アクセス・ライブラリーは、制御 API 関数呼び出しを使用するローカル・アプリケーションをサポートするもので、リモート専用のアプリケーションには別のライブラリーが提供されています。

リモートで実行されるアプリケーションに制御 API 関数呼び出しが含まれている場合は、制御 API スタブ・ライブラリー (例えば Windows での solidctrlstub.lib) にリンクする必要があります。このライブラリーは、実際にご使用のリモート・アプリケーションからサーバーの制御を可能にするものではなく、単に、リンク・ライブラリー・アクセスを使用する solidDB からリンク時エラーを出さずに、アプリケーションのリモート・アプリケーションとしてのコンパイルおよびリンクを行えるようにするものです。

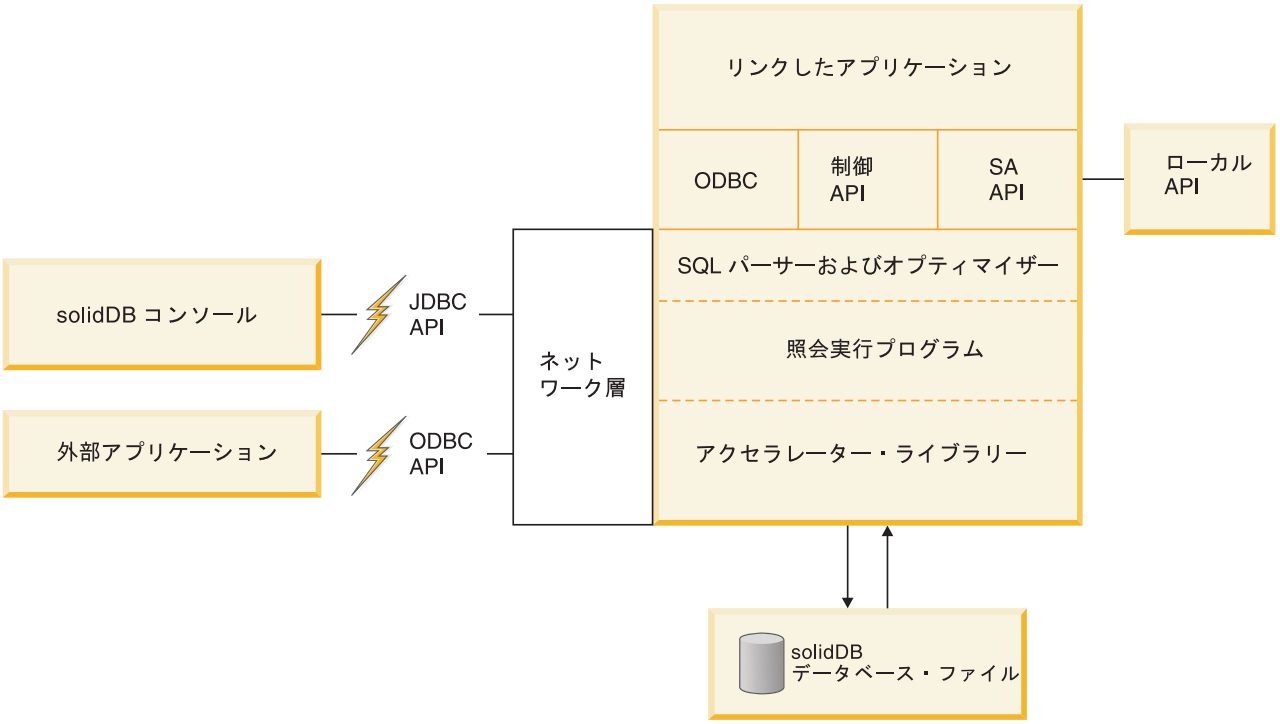


図 3. リンク・ライブラリー・アクセスを使用する solidDB - API

2 リンク・ライブラリー・アクセス・アプリケーションの作成および実行

この章では、リンク・ライブラリー・アクセス・アプリケーションを作成して実行する方法について説明します。この章のトピックは以下のとおりです。

- リンク・ライブラリー・アクセス・ライブラリーのダウンロード
- リンク・ライブラリー・アクセス・ライブラリーへのアプリケーションのリンク
- データベースの作成または既存のデータベースの使用
- リンク・ライブラリー・アクセスを使用する solidDB の始動と停止

注:

この章では、リンク・ライブラリー・アクセス固有の追加事項、補足事項、および solidDB でリンク・ライブラリー・アクセスを使用する場合と使用しない場合の違いを説明します。solidDB SQL、solidDB データ管理ツール、solidDB の全般的な管理と保守、およびデータベース・エラー・コードについては、「*solidDB 管理者ガイド*」を参照してください。リンク・ライブラリー・アクセスでサポートされている API を使用したアプリケーションの開発について詳しくは、25 ページの『3 章 制御 API について』および「*solidDB プログラマー・ガイド*」を参照してください。

リンク・ライブラリー・アクセス・ライブラリーへのアクセス

solidDB リンク・ライブラリー・アクセスは、solidDB Development Kit に含まれているライブラリー・ファイルです。

例えば、HP-UX で solidDB を使用している場合、リンク・ライブラリー・アクセス・ライブラリー・ファイルは solidac.a です。プラットフォーム固有のライブラリーについては、12 ページの『リンク・ライブラリー・アクセスへのアプリケーションのリンク』を参照してください。

全プラットフォームに対応するリンク・ライブラリー・アクセス・ライブラリーには、以下が含まれています。

- solidDB データ管理機能
- ローカル・ユーザー・アプリケーション用の SA API ヘッダー (sa.h)
- ローカル・ユーザー・アプリケーション用の solidDB 制御 API インターフェース・ヘッダー (sscapi.h)

リンク・ライブラリー・アクセス・ライブラリーへのユーザー・アプリケーションのリンクについて詳しくは、 12 ページの『リンク・ライブラリー・アクセスへのアプリケーションのリンク』を参照してください。

リモート・アプリケーション用のライブラリー

このリンク・ライブラリー・アクセス・ガイドでは、サーバーにリンクされていないアプリケーション、つまりリンク・ライブラリー・アクセス・ライブラリーを使用していないアプリケーションが「リモート」アプリケーションと呼ばれます。したがって、データベース・サーバーと同じノードで実行されていてもそのサーバーにリンクされていないアプリケーションは、リモート・アプリケーションと見なされます。リモート・アプリケーションは、TCP/IP などのネットワーク通信プロトコルを介してサーバーと通信します。一方「ローカル」アプリケーションは、リンク・ライブラリー・アクセス・ライブラリーにリンクされており、そのライブラリー内の関数をネットワーク・プロトコルを介さずに直接呼び出すことができます。

リモート・アプリケーションの場合は、ネットワーク通信プロトコルを介するために、リンク・ライブラリー・アクセスを使用してもパフォーマンスは向上しません。パフォーマンスが向上するのはローカル・アプリケーション (アクセラレーター・ライブラリーに直接リンクされたアプリケーション) だけです。

ただしリモート・アプリケーションでも、データベースに対して読み取りと書き込みを行うための低レベルの操作を可能にする SA API を使用することで、パフォーマンスが向上する場合があります。

リモート・アプリケーションを使用している場合は、solidDB SDKに含まれる以下のライブラリーにアプリケーション内でリンクする必要があります。

- 制御 API 関数呼び出しを使用しているアプリケーションをリモートで実行する場合は、solidDB 制御 API スタブ・ライブラリー (Windows プラットフォームでは solidctrlstub.lib) にリンクします (アプリケーションがリモート・アプリケーションではなくローカルである場合、つまりリンク・ライブラリー・アクセス・ライブラリーに直接リンクされている場合は、solidctrlstub.lib は不要です)。

制御 API スタブ・ライブラリー (solidctrlstub.lib) について詳しくは、 5 ページの『リンク・ライブラリー・アクセスでの二重モード・アプリケーションの使用』を参照してください。

- リモートの solidDB SA API アプリケーション (リンク・ライブラリー・アクセスを使用しないもの) を実行している場合は、solidDB SA API (Windows プラットフォームでは solidimpsa.lib) にリンクします。

ODBC、SA API、または JDBC をリモート・アプリケーションとしてのみ使用する場合 (制御 API 関数呼び出しを使用しない場合) は、solidctrlstub.lib にリンクする必要はありません。

サンプルの C アプリケーション

アクセラレーター制御 API の使用例 (C プログラミング言語) については、インストール・ディレクトリーにある samples/aclib、samples/aclib_smartflow、および samples/control_api を参照してください。これらの C サンプルには、ODBC API 関数を使用して solidDB サーバーに接続するリンクされたアプリケーションが示されています。

データ同期化の使用

solidDB のデータ同期化を初めて使用するユーザーのために、「*solidDB 拡張レプリケーション・ユーザー・ガイド*」にサンプル・スクリプトが用意されています。

サンプルの C アプリケーション acsnet.c (ディレクトリー samples/aclib_smartflow 内) を実行する前に、少なくとも以下のいずれかの作業を行って solidDB の機能に慣れておくことをお勧めします。

- solidDB (リンク・ライブラリー・アクセスなし) を使用して、「*solidDB 拡張レプリケーション・ユーザー・ガイド*」に収録されている SQL スクリプトを実行する。このスクリプトは、samples/smartflow にあります。
- solidDB リンク・ライブラリー・アクセスを使用して、SQL スクリプトをローカルで実行する。前提条件として、この章の指示に従って、サーバーを始動するようにアプリケーションをセットアップする必要があります。詳しくは、12 ページの『リンク・ライブラリー・アクセスへのアプリケーションのリンク』および 17 ページの『solidDB リンク・ライブラリー・アクセスの始動とシャットダウン』を参照してください。

注:

SA API を使用して同期化コマンドを実行することはできません。

- リンク・ライブラリー・アクセス・ライブラリーを伴うインプリメンテーション・サンプル・ファイル aclibstandalone.c を実行すると、標準のサーバーがエミュレートされます。このサンプル・ファイルは、ディレクトリー samples/aclib にあります。

上記のどの方法を使用した場合でも、「*solidDB 拡張レプリケーション・ユーザー・ガイド*」の『*データ同期化の概要*』という章に記載されているすべてのステップを solidDB SQL エディター (solsql) で実行できるようになります。

拡張レプリケーション・スクリプトによる ODBC アプリケーションのセットアップ

サンプルの C アプリケーション acsNet.c に似た ODBC アプリケーションをビルドして、同期環境をセットアップ、構成、および実行するために必要なすべてのステートメントを実行することができます。acsNet.c は、ディレクトリー samples/aclib_smartflow にあります。

ODBC クライアント・アプリケーションで使用するサンプル・データベースをセットアップするには、サンプル・スクリプト replica3.sql、replica4.sql、replica5.sql、および replica6.sql を実行します。このスクリプトはいずれも samples/smartflow/eval_setup ディレクトリーにあります。これらのサンプル・スクリプトには、新しいデータをレプリカに書き込み、同期メッセージの実行を制御する SQL ステートメントが含まれています。スクリプトは、solidDB SQL エディター (solsql) から単独で実行できます。

あるいは、SQL ステートメントを C/ODBC アプリケーションに埋め込んでコンパイルし、リンク・ライブラリー・アクセス・ライブラリーに直接リンクできます。サンプル・スクリプトをリンク・ライブラリー・アクセスにリンクすることで、リンク・ライブラリー・アクセスのアーキテクチャー特有のパフォーマンス上の利益を得られます。

samples/odbc ディレクトリーにあるサンプル・プログラム embed.c には、リンク・ライブラリー・アクセスを使用する ODBC クライアント・アプリケーションでデータベースをセットアップする方法が示されています。embed.c アプリケーションには、replica3.sql などのサンプル・スクリプトから SQL コマンドを挿入できます。

リンク・ライブラリー・アクセスへのアプリケーションのリンク

solidDB リンク・ライブラリー・アクセスは、ユーザー・アプリケーションにリンクする必要があるライブラリーです。アプリケーションが実行されている間は、solidDB データ管理サービスに対するローカル・アプリケーションとリモート・アプリケーションの要求が、このライブラリーを通じて行われます。

注:

solidDB 制御 API を使用するリモート・ユーザー・アプリケーションを作成する場合は、リモート・アプリケーションを solidDB 制御 API スタブ・ライブラリー (例えば Windows では solidctrlstub.lib) にリンクする必要があります。solidDB SA API をリモートで (リンク・ライブラリー・アクセスなしで) 使用する場合は、別の solidDB API ライブラリー (Windows では solidimpsa.lib) にリンクする必要があります。ODBC、SA API、または JDBC を制御 API を使わずにリモートで使用する場合、solidDB 制御 API スタブ・ライブラリーにリンクする必要はありません。

リンク・ライブラリー・アクセス・ライブラリーに直接リンクできるアプリケーションは、一度に 1 つだけです。ただし、リンクされたアプリケーションが稼働状態になり、サーバーが始動されると、あらゆるネットワーク・クライアントがそのサーバーのサポートするプロトコル (オペレーティング・システムによって異なります) をどれでも使用してサーバーに接続できるようになります。このプロトコルには、TCP/IP、共用メモリー、名前付きパイプなどがあります。リモート・クライアントは、直接関数呼び出しを使用できません。

リンク・ライブラリー・アクセスを使用する solidDB にアプリケーションをリンクする場合は、使用するオペレーティング・システムに合わせて以下のライブラリーのいずれかを使用します。詳しくは、オペレーティング・システムの文書を参照してください。

表 3. リンク・ライブラリー・アクセス・システム・ライブラリー

プラットフォーム	solidDB リンク・ライブラリー・アクセス・ライブラリー
Windows	solidimpac.lib (これはインポート・ライブラリー・ファイルで、実際のライブラリー・ファイルである ssolidacxx.dll へのアクセスを提供します)
Solaris	solidac.a
HP-UX	solidac.a
Linux®	solidac.a
VxWorks	solidac.a

リンク・ライブラリー・アクセスに対応するためのユーザー・アプリケーションの準備

リンク・ライブラリー・アクセスを使用する solidDB をアプリケーションで使用できるようにするには、以下の作業が必要です。

- ドライバー・ライブラリーではなくリンク・ライブラリー・アクセス・ライブラリーにリンクします。

  リモート・アプリケーションを使用している場合は、他のライブラリーへのリンクが必要となることがあります。詳しくは、10 ページの『リモート・アプリケーション用のライブラリー』を参照してください。

- 接続ストリングをローカルまたはリモートのサーバー名に変更します。詳しくは、16 ページの『リンク・ライブラリー・アクセスを使用する solidDB とのローカル接続またはリモート接続の確立』を参照してください。

- 必要に応じて、SSCStartServer および SSCStopServer の呼び出し、またはその他の制御 API 呼び出しを追加します。詳しくは、27 ページの『制御 API リファレンス』を参照してください。

シグナル・ハンドラー

シグナル・ハンドラーは、例外イベント (ゼロによる除算など) の発生をアプリケーションに報告するために使用されます。ユーザー・アプリケーションではシグナル・ハンドラーを設定しないようにする必要があります。これは、リンク・ライブラリー・アクセスによって設定されたシグナル・ハンドラーがオーバーライドされてしまうためです。例えば、ユーザー・アプリケーションで浮動小数点例外に対してシグナル・ハンドラーを設定すると、リンク・ライブラリー・アクセスによって設定されたハンドラーがオーバーライドされます。このため、サーバーは例えばゼロによる除算を catch できなくなります。

ダイナミック・リンク・ライブラリー

solidDB では、リンク・ライブラリー・アクセス・ライブラリーの「静的」バージョンと「動的」バージョンの両方を提供しています。一部の主要なプラットフォームで使用されるダイナミック・リンク・ライブラリー・ファイルの名前を以下に示します (静的ライブラリーの名前については、 12 ページの『リンク・ライブラリー・アクセスへのアプリケーションのリンク』を参照してください。

表 4. ライブラリー・ファイル

プラットフォーム	solidDB リンク・ライブラリー・アクセス・ライブラリー
Windows	ssolidacxx.dll
Solaris	ssolidacxx.so
HP-UX	ssolidacxx.sl
Linux	ssolidacxx.so

静的ライブラリー・ファイルと動的ライブラリー・ファイルの両方に、solidDB サーバーの完全なコピーがライブラリー形式で入っています。静的ライブラリー・ファイル (lib/solidac.a など) を使用する場合は、プログラムをそのファイルに直接リンクします。その結果、プログラム・コードとライブラリー・コードの両方が実行可能ファイルに書き込まれます。動的ライブラリー・ファイルにリンクする場合は、実行可能プログラムを含む出力ファイルにライブラリーのコードが書き込まれることはありません。代わりに、このコードは、プログラムの実行時にダイナミック・リンク・ライブラリーから別にロードされます。

静的ライブラリー・ファイルにリンクした場合と動的ライブラリー・ファイルにリンクした場合とでは、実行可能プログラムのサイズが変わる以外にほとんど違いはありません。メモリーに 1 回に読み込まれる全体的なコード量はほぼ同じです (コンピューター上でクライアントとサーバーを 1 つずつ実行している場合)。パフォーマンスもほぼ同じですが、動的ライブラリーを使用する場合は、わずかに余分なオーバーヘッドが生じます。

ダイナミック・リンク・ライブラリー・ファイルを使用する主な利点は、同じコンピューター上でサーバーのコピーを複数実行する場合にメモリーを節約できることです。例えば、1 台のコンピューターで開発作業を行うときに、拡張レプリケーションのマスターとレプリカの両方を同時にそのコンピューターに配置する場合や、HotStandby の 1 次サーバーと 2 次サーバーを同時に配置する場合は、動的ライブラリーを使用することで、リンク・ライブラリー・アクセス・ライブラリーの複数のコピーを同時にメモリーに格納する必要がなくなります。

Microsoft Windows では、solidDB リンク・ライブラリー・アクセスに lib/solidimpac.lib というファイルが追加されます。また、Microsoft Windows でダイナミック・リンク・ライブラリーを使用する場合は、ssolidacxx.dll ダイナミック・リンク・ライブラリー自体に直接リンクするのではなく、インポート・ライブラリーである solidimpac.lib にリンクします。これにより、少量のコードのみがクライアント実行可能プログラムにリンクされます。クライアント・プログラムが実際に実行されるときに、Microsoft Windows オペレーティング・システムによって

ssolidacxx.dll ファイルが自動的にロードされ、クライアントがその .dll ファイルにある通常のリンク・ライブラリー・アクセス関数を呼び出せるようになります。.dll ファイルを参照するプログラムを実行するときには、その .dll がロード・パスに配置されている必要があります。

注:

ダイナミック・リンク・ライブラリー・ファイルを使用することで、solidDB サーバーに複数の「ローカル」クライアントをリンクできるわけではありません。動的ライブラリーを使用する場合でも、ローカル・クライアントの数は 1 つに制限されます。その他のクライアントはすべてリモート・クライアントでなければなりません。つまり、ローカル・クライアントのように関数を直接呼び出す方法ではなく、TCP などのネットワーク・プロトコルを使用して solidDB サーバーと通信します。

Make ファイルの例

Windows および Vxworks でライブラリー名を指定する例を以下に示します。

Microsoft Windows ファイルの例

以下に示す Microsoft Windows での Make ファイルの例では、リンク・ライブラリー・アクセス用の solidDB ライブラリー名として solidimpac.lib が使用されています。

```
# コンパイラー
CC        = cl
# コンパイラー・フラグ
CFLAGS        = -I. -DSS_WINDOWS -DSS_WINNT
# リンカー・フラグとディレクティブ
SYSLIBS = libcmt.lib kernel32.lib advapi32.lib netapi32.lib wsock32.lib
user32.lib oldnames.lib gdi32.lib
LFLAGS  = ..¥solidimpac.lib
OUTFILE  = -Fe

# MyApp のビルド
all: myapp

myapp: myapp.c
  $(CC) $(CFLAGS)  $(OUTFILE)myapp myapp.c /link$(LFLAGS)
/NODEFAULTLIB:libc.lib
```

VxWorks での Make ファイルの例

以下に示す VxWorks での Make ファイルの例では、リンク・ライブラリー・アクセス用の solidDB ライブラリー名として solidac.a が使用されています。例では円記号が使用されていることに注意してください。Make ファイル・プログラムがパス名での円記号をサポートしていない場合は、円記号をスラッシュに変更します。

```
CC      = ccppc
CFLAGS  = -DSS_UNIX -DSS_VXW -I. -I..¥..¥include -I$(WIND_BASE)
¥target¥h ¥
          -DCPU=PPC603 -DMV2600
LFLAGS  = -nostartfiles -s -r ..¥..¥lib¥solidac.a
OUTFILE = -o

# AcceleratorLib を使用する solidDB のサンプルのビルド

all:    acsNet acsrv

acsNet: acsNet.c
```

```
        $(CC) $(CFLAGS) $(OUTFILE)acsNet acsNet.c $(LFLAGS)

acsrv:  acsrv.c
        $(CC) $(CFLAGS) $(OUTFILE)acsrv acsrv.c $(LFLAGS)
```

リンク・ライブラリー・アクセスを使用する solidDB とのローカル接続またはリモート接続の確立

リンク・ライブラリー・アクセス・ライブラリーにリンクされたアプリケーションでは、ODBC API または SAAPI を使用して、ローカル・サーバーとのローカル接続またはリモート接続を直接確立できます。また、リンク・ライブラリー・アクセスを使用するサーバーも含めた他の solidDB サーバーとのリモート接続を確立することもできます。

ローカル接続の確立

ローカル接続を確立すると、サーバーに対するクライアントの呼び出しが、リンク・ライブラリー・アクセス・ライブラリーへの直接関数呼び出しとして行われます。つまり、呼び出しがネットワークを経由しません。

ODBC API でローカル・サーバー (つまりアプリケーションにリンクされたサーバー) との接続を確立するには、ユーザー・アプリケーションでリテラル・ストリング ″localserver″ を指定して SQLConnect 関数を呼び出します。ローカル・サーバー接続の場合は、空のソース名 ″″ を指定することもできます。ローカル・サーバー名を指定することもできますが、その場合はリンク・ライブラリー・アクセスで「リモート」接続が使用されます (リンク・ライブラリー・アクセス・ライブラリーへの直接関数呼び出しを使用するのではなくネットワーク経由で接続)。

以下の ODBC API コードの例では、ユーザー名 dba とパスワード dba を使用して、ローカルの solidDB サーバーに直接接続します。

```
rc = SQLConnect(hdbc, "localserver", (SWORD)SQL_NTS, "dba", 3, "dba", 3);
```

または

```
rc = SQLConnect(hdbc, "", (SWORD)SQL_NTS, "dba", 3, "dba", 3);
```

SA API で接続を確立するには、ユーザー・アプリケーションでリテラル・ストリング ″localserver″ (サーバー名ではなく) を指定して SaConnect 関数を呼び出します。ローカル・サーバー接続の場合は、空のソース名 ″″ を指定することもできます。ローカル・サーバー名を指定することもできますが、その場合はリンク・ライブラリー・アクセスで「リモート」接続が使用されます (リンク・ライブラリー・アクセス・ライブラリーへの直接関数呼び出しを使用するのではなくネットワーク経由で接続)。

以下の SA API コードの例では、ユーザー名 dba とパスワード dba を使用して solidDB サーバーに直接接続します。

```
SaConnectT* sc = SaConnect("localserver", "dba", "dba");
```

または

```
SaConnectT* sc = SaConnect("", "dba", "dba");
```

リモート接続の確立

リモート接続を確立すると、サーバーに対するクライアントの呼び出しが、リンク・ライブラリー・アクセス・ライブラリーへの直接関数呼び出しを使用するのではなく、ネットワークを経由して行われます。

ODBC API でリモート接続を確立するには、ユーザー・アプリケーションでリモート・サーバーの名前を指定して SQLConnect 関数を呼び出します。以下の ODBC API コードの例では、ユーザー名 dba とパスワード dba を使用して、リモートの solidDB サーバーに接続します。この例でクライアントとサーバーが使用するネットワーク・プロトコルは、「tcp」(TCP/IP) です。サーバーの名前は「remote_server1」、サーバーが listen するポートは 1313 です。

```
rc = SQLConnect(hdbc, "tcp remote_server1 1313",
(SWORD)SQL_NTS, "dba", 3, "dba", 3);
```

SA API でリモート接続を確立するには、ユーザー・アプリケーションでリモート・サーバーの名前を指定して SaConnect 関数を呼び出します。この例でクライアントとサーバーが使用するネットワーク・プロトコルは、「tcp」(TCP/IP) です。サーバーの名前は「remote_server1」、サーバーが listen するポートは 1313 です。

```
SaConnectT* sc = SaConnect("tcp remote_server1 1313", "dba", "dba");
```

solidDB リンク・ライブラリー・アクセスの始動とシャットダウン

solidDB サーバーを始動、再始動、およびシャットダウンするには、以下の API を使用します。

-

  明示的には、ローカルの (リンクされた) ユーザー・アプリケーションから制御 API 関数 SSCStartServer を使用して solidDB を始動し、SSCStopServer を使用してシャットダウンします。

  まだデータベースがない新しい solidDB サーバーを始動する場合は、solidDB で新しいデータベースを作成するように明示的に指定する必要があります。そのためには、関数 SSCStartServer() で以下のパラメーターを指定します。

  ```
  -Uusername
  -Ppassword
  -Ccatalogname (デフォルトのデータベース・カタログ名)
  ```

  詳しくは、 18 ページの『制御 API 関数 SSCStartServer による明示的な始動』を参照してください。

-

  暗黙的には、solidDB に初めてローカルで接続するときに、ODBC API 関数 SQLConnect または SA API 関数 SaConnect を使用します。この場合、solidDB との最後のローカル接続が関数 SQLDisconnect または SaDisconnect で切断されると、シャットダウンが行われます。

  アプリケーションから solidDB エンジン/サーバーが暗黙的に始動されるときに、solidDB ディレクトリーにデータベースが存在するかどうかが検査されます。データベース・ファイルが見つかった場合は、solidDB が自動的にそのデータベースを開きます。データベース・ファイルが見つからなかった場合は、solidDB か

らエラーが通知されます (暗黙的な始動では solidDB で新しいデータベースが作成されません。新しいデータベースを作成するには、SSCStartServer などの明示的な始動関数を使用し、適切なパラメーターを渡す必要があります)。

詳しくは、 21 ページの『ODBC API 関数呼び出し SQLConnect による暗黙始動』および 22 ページの『SA API 関数呼び出し SaConnect による暗黙始動』を参照してください。

注:

1. サーバー始動時に、アプリケーションに制御が戻る前に必要に応じてリカバリーが実行されます。したがって、サーバーが正常に始動された場合は、サーバーがアプリケーション要求を処理できる状態にあります。アプリケーション・プロセスが実行されている間は、サーバーを必要に応じて始動または停止できます。
2. ディスクレス・サーバーを始動する場合は、制御 API 関数 SSCStartDisklessServer でサーバーを始動する必要があります。

制御 API 関数 SSCStartServer による明示的な始動

solidDB を明示的に始動するには、ユーザー・アプリケーションで以下の制御 API 関数を呼び出します。

```
SSCStartServer (int argc, char* argv [ ],
SscServerT* h, SscStateT runflags)
```

ここで使用されているパラメーターを以下で説明します。

表 5. SSCStartServer のパラメーター

パラメーター	説明
argc	コマンド行引数の数。
argv	関数呼び出しで使用されるコマンド行引数の配列。引数 argv[0] は、ユーザー・アプリケーションのパスおよびファイル名用として予約されており、必ず指定する必要があります。有効なオプションについては、以下の SSCStartServer オプションを参照してください。
h	各サーバーには、そのサーバーを識別し、そのサーバーに関する情報の保管場所を示す「ハンドル」(データ構造へのポインター) があります。このハンドルは、他の制御 API 関数でサーバーを参照する際に必要となります。サーバーのハンドルは、SSCStartServer 関数を呼び出したときに提供されます。サーバーのハンドルを取得するには、「サーバー・ハンドルへのポインター」型の変数 (つまり、ハンドルへのポインター (本質的にはポインターへのポインター) である SSCServerT *) を作成し、SSCStartServer を呼び出すときにその変数を渡します。サーバーが正常に作成されると、SSCStartServer 関数によって新しいサーバーのハンドル (ポインター) が、アドレスを渡した変数に書き込まれます。

表 5. SSCStartServer のパラメーター (続き)

パラメーター	説明
runflags	このパラメーターのオプションは、SSC_STATE_OPEN (リモート接続を許可) および SSC_STATE_PREFETCH (サーバーが必要に応じてプリフェッチを実行) です。プリフェッチでは、表内容の先読み機能を備えたメモリーおよびディスク・キャッシュ、またはそのいずれかが参照されます。runflags パラメーターは以下のように入力します。 `runflags = SSC_STATE_OPEN | SSC_STATE_PREFETCH;`

サーバーを最初に始動するときに、データベース管理者のユーザー名、パスワード、およびデフォルト・データベース・カタログの名前を指定した場合に限り、solidDB によって新しいデータベースが作成されます。データベース・カタログについて詳しくは、「*solidDB SQL ガイド*」の『データベース管理のための solidDB SQL の使用』という章の『データベース・オブジェクトの管理』を参照してください。

例:

```
SscServerT h; char* argv[4];
argv[0] = "appname"; /* ユーザー・アプリケーションのパスとファイル名 */
argv[1] = "-UDBA"; /* ユーザー名 */
argv[2] = "-PDBA"; /* ユーザーのパスワード */
argv[3] = "-CDBA"; /* カタログ名 */
/* サーバーを始動 */
rc = SSCStartServer(argc, argv, &h, run_flags);
```

データベースが存在しない状態でデータベース・カタログ名を指定せずにサーバーを始動すると、データベースが見つからないことを通知するエラーが solidDB から返されます。

注:

データベースが既にある場合は、ユーザー名とパスワード、またはカタログ名を指定する必要はありません。

デフォルトでは、データベースが 1 つのファイルとして (デフォルト名の solid.db または solid.ini ファイルで指定した名前で) solidDB 作業ディレクトリー (現行作業ディレクトリーがあるディレクトリー) に作成されます。システム表とシステム・ビューのみで構成される空のデータベースは、約 850 KB のディスク・スペースを使用します。データベースの作成に要する時間は、使用しているハードウェア・プラットフォームによって異なります。

データベースが作成されると、solidDB がネットワークでリモート・クライアント接続要求の listen を開始します。

SSCStartServer argv パラメーターのオプション

以下に示すのは、argv パラメーターのコマンド行オプションです。いずれのオプションも大/小文字が区別されるので注意してください。

表 6. *SSCStartServer argv* のオプション

オプション	説明
-c *dir*	作業ディレクトリーを変更します。
-m	ユーザーのメッセージと SQL ステートメントをモニターします。
-n *name*	サーバー名を設定します。
-U *username*	作成されるデータベースの管理者のユーザー名を指定します。ユーザー名は大/小文字が区別されず、2 文字以上であることが必要です。ユーザー名の最大文字数は 80 です。ユーザー名は英字またはアンダースコアーで始まっていなければなりません。使用できる文字は、英小文字の a から z、英大文字の A から Z、アンダースコアー文字 (_)、および 0 から 9 の数字です。 **注意:** **ユーザー名を忘れると、solidDB に接続できません。デフォルトのユーザー名はありません。データベース作成時に入力するユーザー名が、その新しいデータベースに接続する際に使用できる唯一のユーザー名となります。**
-P *password*	作成されるデータベースの管理者のパスワードを指定します。パスワードは大/小文字が区別されず、3 文字以上であることが必要です。パスワードは、英字、アンダースコアー、または数字で始めることができます。使用できる文字は、英小文字の a から z、英大文字の A から Z、アンダースコアー文字 (_)、および 0 から 9 の数字です。
-C *catalogname*	データベースのデフォルト・カタログの名前を指定します。サーバーを初めて始動する場合はこの指定が必要です。カタログについて詳しくは、「*solidDB SQL ガイド*」の『データベース管理のための solidDB SQL の使用』という章の『データベース・オブジェクトの管理』を参照してください。
-xautoconvert	データベース・フォーマットを現行バージョンに変換し、サーバー・プロセスを開始します。
-xforcerecovery	ロールフォワード・リカバリーを強制的に実行します。
-xignoreerrors	索引エラーを無視します。
-xtestblocks	データベース・ブロックをテストします。
-xtestindex	データベースの索引をテストします。

SSCStartServer の開始

サーバー名、カタログ名、および管理者のユーザー名とパスワードを指定して SSCStartServer を開始します。

```
SscStateT runflags = SSC_STATE_OPEN; SscServerT h; char* argv[5];
argv[0] = "appname"; /* ユーザー・アプリケーションのパスとファイル名 */
argv[1] = "-nsolid1"; argv[2] = "-UDBA" argv[3] = "-PDBA";
argv[4] = "-CDBA"; /* サーバーの始動 */ rc =
SSCStartServer(argc, argv, &h, run_flags);
```

注:

データベースが既にある場合は、ユーザー名とパスワード、またはカタログ名を指定する必要はありません。

SSCStopServer によるシャットダウン

サーバーを SSCStartServer で始動した場合は、組み込みアプリケーションで以下の関数呼び出しを使用してそのサーバーをシャットダウンする必要があります。

SSCStopServer()

例:

```
/* サーバーの停止 * /
SSCStopServer (h, TRUE);
```

ODBC API 関数呼び出し SQLConnect による暗黙始動

関数 SQLConnect が初めて呼び出されるときに、サーバーが暗黙的に始動されます。ユーザー・アプリケーションが開いている最後のローカル接続に対して関数 SQLDisconnect を呼び出すと、サーバーが暗黙的にシャットダウンされます。その場合、リモート接続が存在していてもサーバーがシャットダウンされるので注意してください。

注:

サーバーを初めて始動するときは、 solidDB データベースを作成する必要があります。そのためには、関数 SSCStartServer() を使用し、デフォルトのデータベース・カタログ、および管理者のユーザー名とパスワードを指定します。説明と例については、 18 ページの『制御 API 関数 SSCStartServer による明示的な始動』を参照してください。

SQLConnect および SQLDisconnect による暗黙的な始動とシャットダウンの例を以下に示します。

```
/* 接続 #1 */
rc = SQLConnect (hdbc1, "", SQL_NTS, "dba", SQL_NTS, "dba",
SQL_NTS); // ここでサーバーが始動
... ODBC 呼び出し

/* 切断 #1 */
SQLDisconnect (hdbc1); // ここでサーバーがシャットダウン

/* 接続 #2 */
rc = SQLConnect (hdbc2, "", SQL_NTS, "dba", SQL_NTS, "dba",
SQL_NTS); // ここでサーバーが始動
... ODBC 呼び出し

/* 切断 #2 * /
SQLDisconnect (hdbc2); // ここでサーバーがシャットダウン
```

または

```
/* 接続 #1*/
rc = SQLConnect (hdbc1, "", SQL_NTS, "dba",
SQL_NTS, "dba", SQL_NTS); // ここでサーバーが始動

/* 接続 #2*/
rc = SQLConnect (hdbc2, "", SQL_NTS, "dba", SQL_NTS, "dba", SQL_NTS);

... ODBC 呼び出し

/* 切断 #1 */
SQLDisconnect (hdbc1);
/* 切断 #2 * /
SQLDisconnect (hdbc2); // ここでサーバーがシャットダウン
```

注:

SSCStartServer 関数呼び出しでサーバーを始動した場合は、SQLDisconnect で暗黙シャットダウンが行われません。SSCStopServer、ADMIN COMMAND 'shutdown'、またはその他の明示的なシャットダウン・メソッドで、サーバーを明示的にシャットダウンする必要があります。

```
SscStateT runflags = SSC_STATE_OPEN;
SscServerT server;
SQLHDBC hdbc;
SQLHENV henv;
SQLHSTMT hstmt;

/* サーバーの始動 */
SSCStartServer (argc, argv, &server, runflags); // ここでサーバーが始動

/* 環境の割り振り */
rc = SQLAllocEnv (&henv);

/* データベースへの接続 */
rc = SQLAllocConnect (henv, &hdbc);
rc = SQLConnect (hdbc, "", SQL_NTS, "dba", SQL_NTS, "dba", SQL_NTS);

/* 表 foo からの全行削除 */
rc = SQLAllocStmt (hdbc, &hstmt):
rc = SQLExecDirect (hsmt, (SQLCHAR *) "DELETE FROM FOO", SQL_NTS);

/* コミット */
rc = SQLTransact (henv, hdbc, SQL_COMMIT);
rc = SQLFreeStmt (hstmt, SQL_DROP);

/* 切断 */
SQLDisconnect (hdbc);
SQLFreeConnect (hdbc);

/* 環境の解放 */
SQLFreeEnv(henv);

/* サーバーの停止 */
SSCStopServer (server, TRUE);  // ここでサーバーがシャットダウン
```

SA API 関数呼び出し SaConnect による暗黙始動

関数 SaConnect が初めて呼び出されるときに、サーバーが暗黙的に始動されます。ユーザー・アプリケーションが関数 SaDisconnect を呼び出したときにそれ以上の接続が残っていない場合は、サーバーが暗黙的にシャットダウンされます。

注:

サーバーを初めて始動するときは、 solidDB データベースを作成する必要があります。そのためには、関数 SSCStartServer() を使用し、デフォルトのデータベース・カタログ、およびユーザー名とパスワードを指定します。説明と例については、18 ページの『制御 API 関数 SSCStartServer による明示的な始動』を参照してください。

SaConnect および SaDisconnect による暗黙的な始動とシャットダウンの例を以下に示します。

```
/* 接続を開く */
SaConnect(...);

ここでサーバーが始動
... sa 呼び出し

/* 接続を閉じる */
SaDisconnect(...);

ここでサーバーがシャットダウン
```

注:

サーバーを SSCStartServer 関数呼び出しで始動した場合は、SSCStopServer 関数呼び出しでそのサーバーをシャットダウンする必要があります。

solidDB リンク・ライブラリー・アクセスのシャットダウン

SYS_ADMIN_ROLE 特権があれば、solidDB クライアント・インターフェースから、さらに別のリモート solidDB 接続から、solidDB サーバーをシャットダウンすることができます。

プログラムで、solidDB SQL エディター (solsql) や solidDB リモート制御などのアプリケーションからシャットダウンを実行できます。[1].

そのためには、以下のステップを実行します。

1. solidDB との新たな接続が作成されないように、以下のコマンドを入力してデータベースを閉じます。

   ADMIN COMMAND 'close'

2. 以下のコマンドを入力してすべての solidDB ユーザーを終了します。

   ADMIN COMMAND 'throwout all'

3. 以下のコマンドを入力して solidDB を停止します。

   ADMIN COMMAND 'shutdown'

1. ステップ 1 から 3 に solidDB リモート制御を使用する場合は、コマンド名を引用符なしで入力します (例えば `close`)。

シャットダウン・メカニズムはいずれも同じルーチンを開始します。このルーチンは、バッファー内の全データをデータベース・ファイルに書き込み、キャッシュ・メモリーを解放し、最後にサーバー・プログラムを終了します。サーバーでバッファー内の全データをメイン・メモリーからディスクに書き込む必要があるため、サーバーのシャットダウンには時間がかかることがあります。

注:

暗黙的な方法 (SQLConnect など) で始動されたサーバーは、明示的な方法 (SSCStopServer など) でシャットダウンできます。ただし、その逆は不可能です。例えば、SSCStartServer で始動されたサーバーを SQLDisconnect で停止することはできません。

暗黙始動の構成パラメーター

solidDB は、ローカル接続が確立されている場合に限り、サーバーを暗黙的に始動します。 solid.ini 構成ファイルの Accelerator セクションでは、パラメーター ImplicitStart がデフォルトで Yes に設定されます。このデフォルト設定では、ODBC 接続に必要な SQLConnect 関数を使用すると、自動的にサーバーが始動されます。関数 SaConnect も同じように動作します。この関数が初めて呼び出されるときに、サーバーが暗黙的に始動されます。

3 制御 API について

制御 API (SSC API とも呼ばれます) は、solidDB のタスク処理システムを簡単かつ効率よく制御する一連の関数です。

タスク情報の取得

すべてのアクティブ・タスクのリストを取得するには、SSCGetActiveTaskClass 関数を使用します。中断状態にある全タスクのリストを取得するには、SSCGetSuspendedTaskClass 関数を使用します。タスク・クラスの優先順位を取得するには、 SSCGetTaskClassPrio 関数を使用します。

特別なイベントの通知関数

リンク・ライブラリー・アクセスでは、優先タスクを微調整することができます。SSCSetNotifier() 関数を使用することで、特別なイベントが発生した場合に solidDB から指定のユーザー定義関数が呼び出されるように設定できます。この関数で検出される特別なイベントは以下のとおりです。

- solidDB サーバーのシャットダウン
- 索引からストレージ・ツリーへの Bonsai マージ
- Bonsai マージ間隔の最大値
- バックアップまたはチェックポイントの要求
- 活動停止状態のサーバー
- 1 次サーバーから受信したネットコピー要求 (1 次データベースのネットワーク・コピーを 2 次サーバーに送信する要求)
- ネットワーク・コピー (ネットコピー) を通じて受け取った新しいデータベースでサーバーを始動する際に発生したネットコピー要求の完了

solidDB の状況およびサーバー情報の取得

関数 SSCGetStatusNum を使用すると、solidDB データベース・サーバーの現在の状況情報が表示されます。表示される情報は以下のとおりです。

- Bonsai ツリーからストレージ・ツリーにマージされない行数

SSCGetServerHandle 関数は、solidDB サーバーが稼働している場合にそのサーバー・ハンドルを返します。

また、関数 SSCIsRunning を使用してサーバーが稼働しているかどうかを検証することや、関数 SSCIsThisLocalServer を使用して、アプリケーションのリンク先が、制御 API を使用するリモート・アプリケーションのテストに使用する、ダミーのサーバー・ライブラリー (例えば Windows プラットフォームでの solidctrlstub.lib) であるのか、またはローカルのリンク・ライブラリー・アクセス・サーバー・ライブラリー (例えば Windows プラットフォームでの ssolidacxx.dll) であるのかを検証することもできます。

制御 API 関数の要約

制御 API 関数の簡単な要約、およびその関数が制御 API 関数リファレンスのどこで説明されているかその参照先を以下に示します。

表 7. 制御 API 関数の要約

関数	説明	詳細の参照先
SSCSetCipher	アプリケーションが提供する暗号化ライブラリーを設定します。	33 ページの『SSCSetCipher』
SSCStartServer	solidDB リンク・ライブラリー・アクセス・サーバーを始動します。	43 ページの『SSCStartServer』
SSCStartDisklessServer	solidDB リンク・ライブラリー・アクセス・ディスクレス・サーバーを始動します。	41 ページの『SSCStartDisklessServer』
SSCSetState	solidDB サーバーの状態を設定します (例えば、SSC_STATE_OPEN は後続の接続を許可するかどうかを示します)。状態を ~SSC_STATE_OPEN に設定すると、ローカル接続とリモート接続がブロックされます。	39 ページの『SSCSetState』
SSCRegisterThread	リンク・ライブラリー・アクセス・アプリケーションのスレッドをサーバーに登録します。アクセラレーター API 関数を呼び出すには、事前にユーザー・アプリケーションのすべてのスレッドで登録を行う必要があります。	32 ページの『SSCRegisterThread』
SSCUnregisterThread	サーバーに対するリンク・ライブラリー・アクセス・アプリケーション・スレッドの登録を抹消します。登録抹消は、登録されているすべてのスレッドで終了前に行う必要があります。	49 ページの『SSCUnregisterThread』
SSCStopServer	solidDB サーバーを停止します。	47 ページの『SSCStopServer』
SSCSetNotifier	指定されたイベント (マージ、バックアップ、シャットダウンなど) で solidDB が呼び出すユーザー定義関数を指定します。	36 ページの『SSCSetNotifier』
SSCIsRunning	サーバーが稼働していない場合はゼロ以外の値を返します。	31 ページの『SSCIsRunning』

表 7. 制御 API 関数の要約 (続き)

関数	説明	詳細の参照先
SSCIsThisLocalServer	アプリケーションが、リンク・ライブラリー・アクセスを使用する solidDB サーバーにリンクされているかどうか、またはリンク・ライブラリー・アクセスの制御 API を使用する solidDB リモート・アプリケーションをテストするための「ダミー」ライブラリー (solidctrlstub) にリンクされているかどうかを示します。	32 ページの『SSCIsThisLocalServer』
SSCGetServerHandle	solidDB サーバーが稼働している場合は、そのサーバー・ハンドルを返します。	30 ページの『SSCGetServerHandle』
SSCGetStatusNum	solidDB の状況情報を取得します。	30 ページの『SSCGetStatusNum』

制御 API と対応する ADMIN COMMAND

制御 API 関数には、対応する solidDB SQL 拡張 ADMIN COMMAND があります。これらのコマンドは、リモート・サイトとローカル・サイトの両方から、solidDB リモート制御 (solcon) や solidDB SQL エディター (solsql) などの solidDB ツールを使用して実行できます。

制御 API に対応する ADMIN COMMAND について詳しくは、65 ページの『リンク・ライブラリー・アクセスのパラメーター』を参照してください。

制御 API リファレンス

以下のページでは、各制御 API 関数をアルファベット順に説明します。各説明では、目的、構文、パラメーター、戻り値、およびコメントが示されます。

関数の構文

関数の宣言構文は以下のとおりです。

```
ReturnType SSC_CALL function(modifier parameter[,...]);
```

ReturnType は変動しますが、通常は呼び出しが成功したか失敗したかを示す値です。戻り値について詳しくは、このセクションで後述します。

SSC_CALL は移植性を確保するために必要です。SSC_CALL によって、関数の呼び出し規則を指定します。SSC_CALL は、sscapi.h ファイルで各プラットフォームに合わせて定義されます。

パラメーターは斜体で示されます。パラメーターの説明については、以下を参照してください。

パラメーターの説明

各関数の説明では、パラメーターが表形式で説明されています。この表には、パラメーターの一般的な使用タイプ (以下で説明) および各関数でのパラメーター変数の用途が示されます。

パラメーターの使用タイプ

以下の表は、制御 API パラメーターの想定される使用タイプを示しています。パラメーターがポインターとして使用される場合は、呼び出し後のパラメーター変数の所属を指定する 2 番目の使用カテゴリーがパラメーターに追加されることに注意してください。

表 8. 制御 API パラメーターの使用タイプ

使用タイプ	**意味**
in	パラメーターが入力であることを示します。
output	パラメーターが出力であることを示します。
in out	パラメーターが入出力であることを示します。
use	ポインター・パラメーターにのみ適用されます。パラメーターが単に関数呼び出しで使用されることを意味します。呼び出し元は、関数呼び出し後にこのパラメーターを使用して任意の操作を実行できます。これは、パラメーター引き渡しのタイプとして最もよく使用されます。
take	ポインター・パラメーターにのみ適用されます。パラメーター値が関数で使用されることを意味します。呼び出し元は、関数呼び出し後にこのパラメーターを参照できません。パラメーターが不要となったときは、関数または関数で作成されたオブジェクトがそのパラメーターを解放します。

hold	ポインター・パラメーターにのみ適用されます。関数が関数呼び出し後もパラメーター値を保持することを意味します。呼び出し元は、関数呼び出し後もパラメーター値を参照でき、パラメーターの解放も担当します。
	
	重要:
	
	このパラメーターはユーザーとサーバーが共用するため、サーバーがパラメーターの処理を終えるまでユーザーはパラメーターを解放できません。一般に、保持されているオブジェクトを解放するには、保持しているオブジェクトを先に解放する必要があります。以下に例を示します。
	`conn = SaConnect("", "dba", "dba");`
	`/* 接続はカーソルが解放されるまで保持されます */`
	`scur = SaCursorCreate(conn, "mytable");`
	`...`
	`SaCursorFree(scur);`
	`/* カーソルを解放した後は、接続を安全に解放 */`
	`/* できます (あるいは、この例のように接続を */`
	`/* 解放するサーバー関数を呼び出します)。 */`
	`SaDisconnect(conn);`

戻り値

各関数の説明では、関数が値を返すかどうか、および返される値のタイプを示します。

SscTaskSetT

関数から SscTaskSetT タイプの値が返された場合、この定義はビット・マスクとして使用されます。SScTaskSetT は、sscapi.h で以下の値を使用して定義されます。

```
SSC_TASK_NONE
SSC_TASK_CHECKPOINT
SSC_TASK_BACKUP
SSC_TASK_MERGE
SSC_TASK_LOCALUSERS
SSC_TASK_REMOTEUSERS
SSC_TASK_SYNC_HISTCLEAN
SSC_TASK_SYNC_MESSAGE
SSC_TASK_HOTSTANDBY
SSC_TASK_HOTSTANDBY_CATCHUP
SSC_TASK_ALL (上記の全タスク)
```

HotStandby の「ネットコピー」操作と「コピー」操作は、タスク SSC_TASK_BACKUP によって実行されることに注意してください。SSC_TASK_NETCOPY というタスクは存在しません。

制御 API のエラー・コードとメッセージ

制御 API 関数から次のエラー・コードとメッセージが返される場合があります。

表 9. 制御 API 関数のエラー・コードとメッセージ

エラー・コード/メッセージ	説明
SSC_SUCCESS	操作が正常に終了しました。
SSC_ERROR	一般エラー。
SSC_ABORT	操作が異常終了しました。
SSC_FINISHED	すべてのタスクが実行されると、SSCAdvanceTasks からこのメッセージが返されます。
SSC_CONT	実行するタスクがまだ残っている場合に、SSCAdvanceTasks からこのメッセージが返されます。
SSC_CONNECTIONS_EXIST	開いている接続が存在します。
SSC_UNFINISHED_TASKS	未完了のタスクが存在します。
SSC_INFO_SERVER_RUNNING	サーバーは既に稼働しています。
SSC_INVALID_HANDLE	無効なローカル・サーバー・ハンドルが指定されました。このサーバーは、SSCStartServer で始動したサーバーと一致しません。
SSC_INVALID_LICENSE	ライセンスがないか、無効なライセンス・ファイルが見つかりました。
SSC_NODATABASEFILE	データベース・ファイルが見つかりません。

表 9. 制御 API 関数のエラー・コードとメッセージ (続き)

エラー・コード/メッセージ	説明
SSC_SERVER_NOTRUNNING	サーバーが稼働していません。
SSC_SERVER_INNETCOPYMODE	サーバーがネットコピー・モードです (高可用性/HotStandby を使用する場合のみ)。

上記の定数 (SSC_SUCCESS など) は、sscapi.h ファイルで定義されています。

SSCGetServerHandle

SSCGetServerHandle は、solidDB サーバーが稼働している場合に、そのサーバー・ハンドルを返します。

構文

```
SscServerT SSC_CALL SSCGetServerHandle(void)
```

コメント

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND がありません。

戻り値

- サーバーが稼働していない場合は NULL。
- サーバーが稼働している場合はサーバー・ハンドル。

SSCGetStatusNum

SSCGetStatusNum は、solidDB の状況情報を取得します。

構文

```
SscRetT SSC_CALL SSCGetStatusNum(SscServerT h, SscStatusT stat,
      long * num)
```

SSCGetStatusNum 関数で使用されるパラメーターは以下のとおりです。

表 10. SSCGetStatusNum のパラメーター

パラメーター	使用タイプ	説明
h	in、use	サーバーへのハンドル。
stat	in	取得の対象となる状況 ID を指定します。

表 10. SSCGetStatusNum のパラメーター (続き)

パラメーター	使用タイプ	説明
num	out	関数が正常に終了すると、関数が戻る際にこのパラメーターの値がマージされなかった書き込みの数またはサーバー・スレッドの数に設定されます。どちらに設定されるかは、要求された情報によって決まります。

コメント

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND がありません。

SSCGetStatusNum を呼び出す際に stat パラメーターに認識不能な値を指定すると、この関数から SSC_SUCCESS が返されます。

戻り値

- SSC_SUCCESS - 操作が正常に終了しました。この値は、stat パラメーターに無効な値を指定したときにも返されます。
- SSC_ERROR - 操作が失敗しました。
- SSC_SERVER_INNETCOPYMODE - サーバーはネットコピー・モードです (HotStandby のみ)。
- SSC_SERVER_NOTRUNNING - サーバーが稼働していません。

SSCIsRunning

SSCIsRunning は、サーバーが稼働している場合にゼロ以外の値を返します。

構文

```
int SSC_CALL SSCIsRunning(SscServerT h)
```

SSCIsRunning 関数で使用されるパラメーターは以下のとおりです。

表 11. SSCIsRunning のパラメーター

パラメーター	使用タイプ	説明
h	in、use	サーバーへのハンドル

戻り値

- 0 - サーバーが稼働していません。
- ゼロ以外 - サーバーは稼働しています。

コメント

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND がありません。

SSCIsThisLocalServer

SSCIsThisLocalServer は、アプリケーションが solidDB サーバーまたは「ダミー」ライブラリー (solidctrlstub) にリンクされているかどうかを示します。solidctrlstub ライブラリーを開発時に使用すると、制御 API を使用する solidDB リモート・アプリケーションを、リンク・ライブラリー・アクセス・ライブラリーにリンクすることなく、またソース・コードを変更することなくテストできます。

構文

```
int SSC_CALL SSCIsThisLocalServer(void)
```

戻り値

- 0 - アプリケーションは solidDB サーバーにリンクされていません。
- 1 - アプリケーションは solidDB サーバーにリンクされています。

コメント

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND がありません。

SSCRegisterThread

SSCRegisterThread は、solidDB アプリケーション・スレッドをサーバーに登録します。制御 API、ODBC API、または SA API を使用するスレッドは、すべて登録する必要があります。SSCRegisterThread 関数は、他のリンク・ライブラリー・アクセス API 関数を使用する前にスレッドで呼び出す必要があります。

アプリケーションにスレッドが 1 つしかない場合 (メイン・スレッド)、つまりアプリケーション自体がスレッドを作成しない場合、登録は必要ありません。

スレッドは、終了前に SSCUnregisterThread 関数を呼び出して自身の登録を抹消する必要があります。

構文

```
SscRetT SSC_CALL SSCRegisterThread(SscServerT h)
```

SCCRegisterThread 関数で使用されるパラメーターは以下のとおりです。

表 12. SCCRegisterThread のパラメーター

パラメーター	使用タイプ	説明
h	In、Use	サーバーへのハンドル。

戻り値

- SSC_SUCCESS
- SSC_INVALID_HANDLE

コメント

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND がありません。

関連項目

SSCUnregisterThread

SSCSetCipher

SSCSetCipher 関数は、アプリケーションが提供する暗号および暗号化/暗号化解除関数を設定します。提供された暗号は、SSCStartServer で関連するデータベース暗号化コマンド行引数が使用されるときに自動的に使用されます。

構文

```
void SSC_CALL SSCSetCipher(
    void* cipher,
    char* (SSC_CALL *encrypt)(void *cipher, int page_no, char *page,
        int n, size_t pagesize),
    int (SSC_CALL *decrypt)(void *cipher, int page_no, char *page,
        int n, size_t pagesize));
```

表 13. SSCSetCipher のパラメーター

パラメーター	使用タイプ	説明
cipher	in	アプリケーション固有の暗号オブジェクト (暗号化パスワードなど) へのポインター。solidDB サーバーがこのポインターを使用したり解釈したりすることはありません。アプリケーションが提供する暗号化/暗号化解除関数にこのポインターを渡すだけです。

表 13. *SSCSetCipher* のパラメーター *(続き)*

パラメーター	使用タイプ	説明
encrypt	in	アプリケーションが提供する暗号化関数へのポインター。この関数は、サーバーがデータベース・ファイルまたはログ・ファイルのページを暗号化する必要があるときに呼び出されます。この関数のパラメーターは以下のとおりです。 • cipher - アプリケーションが提供する暗号オブジェクトへのポインター。 • page_no - サーバーが提供するページ番号。暗号化アルゴリズムでは、この情報を無視するか、暗号鍵の一部として使用できます。 • n - 暗号化するページ数。 • pagesize - 暗号化するページのサイズ。 • page - 暗号化されるデータ・バッファーへのポインター。データ・バッファーのサイズは以下のとおりです。 *n*pagesize* この関数は、*n*pagesize* サイズのファイルに書き込まれる暗号化されたデータ・バッファーへのポインターを返す必要があります。 暗号化されたデータ・バッファーへのポインターをサーバーが解放することはありません。 関数に page パラメーターとして渡されたデータ・バッファーは、暗号化関数で自由に上書きまたは操作することができます。例えば、暗号化関数でデータを適切に暗号化し、page ポインターを返すことができます。

表 13. SSCSetCipher のパラメーター (続き)

パラメーター	使用タイプ	説明
decrypt	in	アプリケーションが提供する暗号化解除関数へのポインター。この関数は、サーバーが暗号化されたデータベースまたはログ・ファイルの一部を読み取って暗号化解除する必要があるときに呼び出されます。この関数のパラメーターは以下のとおりです。 • cipher - アプリケーションが提供する暗号オブジェクトへのポインター。 • page_no - サーバーが提供するページ番号。暗号化解除アルゴリズムでは、この情報を無視するか、暗号化解除鍵の一部として使用できます。 • n - 暗号化解除するページ数。 • pagesize - 暗号化解除するページのサイズ。 • page - 暗号化解除されるデータ・バッファーへのポインター。データ・バッファーのサイズは以下のとおりです。 *n*pagesize*

戻り値

SSCSetCipher 関数は値を返しません。この関数は、SSCStartServer 関数を使用してリンク・ライブラリー・アクセス・サーバーを始動する前に呼び出す必要があります。

コメント

暗号化解除関数は、ページの暗号化解除に成功した場合はゼロ以外の値を返し、何らかの理由で暗号化解除に失敗した場合は 0 値を返すようになっています。0 値が返された場合、サーバーは処理を続行できなくなるので緊急時シャットダウンを実行します。この関数は、'page' パラメーターで指定された同じバッファーに暗号化データを返すようになっています。

リンク・ライブラリー・アクセス暗号化 API の使用

以下のコードは、AcceleratorLib 暗号化 API の使用法を示しています。暗号化方式には、一般的な XOR 難読化が使用されます。

```
char* SS_CALLBACK encrypt(void *cipher, int page_no, char *page, int np,
size_t pagesize)
{
        size_t n = np*pagesize;
        int *key = cipher;
        size_t i;

        for (i=0; i<n; i++) {
            page[i] ^= (i+*key);
        }
        return page;
}


bool SS_CALLBACK decrypt(void *cipher, int page_no, char *page, int np,
size_t pagesize)
{
        size_t n = np*pagesize;
        int *key = cipher;
        size_t i;

        for (i=0; i<n; i++) {
            page[i] ^= (i+*key);
        }

        return TRUE;
}
...

int main(int argc, char** argv)
{
    int key = 17;
...
        SSCSetCipher(&key, encrypt, decrypt);
        sscret = SSCStartServer(argsc, args, &h, SSC_STATE_OPEN);
        ....
        SSCStopServer(h, FALSE);
...

}
```

SSCSetNotifier

SSCSetNotifier は、リンク・ライブラリー・アクセス・サーバーが始動または停止されるときに呼び出すコールバック関数を設定します。この関数には対応する ADMIN COMMAND がありません。

構文

```
SscRetT SSC_CALL SSCSetNotifier(SscServerT h, SscNotFunT what,
    notify_fun handler, void* userdata
)
```

SSCSetNotifier 関数で使用されるパラメーターは以下のとおりです。

表 14. SSCSetNotifier 関数のパラメーター

パラメーター	使用タイプ	説明
h	in	サーバーへのハンドル。

表 14. *SSCSetNotifier* 関数のパラメーター (続き)

パラメーター	使用タイプ	説明
what	in	通知の対象となるイベントを指定します。オプションは以下のとおりです。 • SSC_NOTIFY_EMERGENCY_EXIT サーバーが SSCStartServer() で活動化された後にクラッシュした場合に、この関数が呼び出されます。 SSCStartServer() の前に、通知関数の呼び出し SSCSetNotifier() を実行する必要があります。 • SSC_NOTIFY_SHUTDOWN シャットダウン時に関数が呼び出されます。 • SSC_NOTIFY_SHUTDOWN_REQUEST サーバーがシャットダウン要求を受信すると関数が呼び出され、ユーザー定義関数がその要求を受け付けた場合にシャットダウンを行います。通知を受けた関数から SSC_ABORT を返すことでシャットダウンを拒否できます。要求の処理に進む場合は SSC_CONT を返します。 • SSC_NOTIFY_ROWSTOMERGE Bonsai 索引ツリーにストレージ・サーバーにマージする必要があるデータがある場合に、関数が呼び出されます。 • SSC_NOTIFY_MERGE_REQUEST solid.ini 構成ファイルの MergeInterval パラメーターの設定を超過し、マージを開始する必要がある場合に、関数が呼び出されます。 • SSC_NOTIFY_BACKUP_REQUEST バックアップが要求されたときに関数が呼び出されます。バックアップを拒否するには、通知を受けた関数から SSC_ABORT を返します。 • SSC_NOTIFY_CHECKPOINT_REQUEST チェックポイントが要求されたときに関数が呼び出されます。チェックポイントを拒否するには、通知を受けた関数から SSC_ABORT を返します。 • SSC_NOTIFY_IDLE サーバーがアイドル状態に切り替えられたときに関数が呼び出されます。 • SSC_NOTIFY_NETCOPY_REQUEST このコールバック関数は HotStandby コンポーネントのみに適用されます。次サーバーからネットコピー要求 (1 次データベースのネットワーク・コピーを 2 次サーバーに送信する要求) を受信したときに、関数が呼び出されます。netcopy コマンドについて詳しくは、「*solidDB 高可用性ユーザー・ガイド*」を参照してください。 • SSC_NOTIFY_NETCOPY_FINISHED このコールバック関数は HotStandby コンポーネントのみに適用されます。ネットコピー要求が完了したときに関数が呼び出されます。完了すると、ネットワーク・コピー (ネットコピー) を通じて受け取った新しいデータベースでサーバーが始動され、SSC_NOTIFY_FINISHED が呼び出されてサーバーが再び使用可能になったことがアプリケーションに通知されます。
notify_fun_handler	in、hold	呼び出すユーザー関数。

表 14. SSCSetNotifier 関数のパラメーター (続き)

パラメーター	使用タイプ	説明
userdata	in、hold	通知関数に渡されるユーザー・データ。 27 ページの『パラメーターの説明』に記載されている、使用タイプ *hold* のパラメーターの解放に関する注意事項を必ず読んでおいてください。

戻り値

- SSC_SUCCESS - サーバーの要求が受け付けられました。

  HotStandby のみ:

  SSC_NOTIFY_NETCOPY_FINISHED が SSC_SUCCESS を返した場合は、他のアプリケーション接続がすべて終了し、サーバーが「ネットコピー listen モード」に設定されます。このモードでは、サーバーが 1 次サーバーからの接続を受け付けます。また、2 次サーバーでは、HotStandby の netcopy コマンドからデータを受け取ることだけが可能となります。「ネットコピー listen モード」について詳しくは、「*solidDB 高可用性ユーザー・ガイド*」を参照してください (以前は「ネットコピー listen モード」が「バックアップ listen モード」とも呼ばれていたことに注意してください)。

- SSC_ABORT - サーバーの要求が拒否されました。

  HotStandby のみ:

  SSC_NOTIFY_NETCOPY_REQUEST が SSC_ABORT を返した場合は、ネットコピーが開始されず、エラー・コード (SRV_ERR_OPERATIONREFUSED) が 1 次サーバーに返されます。

- SSC_INNETCOPYMODE - サーバーはネットコピー・モードです (HotStandby のみ)。

  SSC_SERVER_NOTRUNNING - サーバーが稼働していません。

コメント

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND がありません。

使用タイプ *hold* のパラメーターを保留解除する場合は注意が必要です。 27 ページの『パラメーターの説明』の hold に関する注意事項を参照してください。

ユーザー定義の通知関数では、SA、SSC、ODBC の各関数を呼び出さないようにする必要があります。

ユーザー定義の通知関数を作成するときは、以下のプロトタイプに準拠する必要があります。

```
int SSC_CALL mynotifyfun(SscServerT h, SscNotFunT what ,void* userdata);
```

SSC_CALL を使用してユーザー関数の規則を明示的に定義したら、SSCSetNotifier 関数を使用してその関数を登録して、その関数が指定のイベントで呼び出されるようにします。以下に例を示します。

```
SscRetT SSCSetNotifier(h, SSC_NOTIFY_IDLE, mynotifyfun, NULL);
```

例

シャットダウン時の関数の呼び出し

シャットダウンが要求されるたびに呼び出される user_own_shutdownrequest という関数をユーザーが作成したとします。

```
int SSC_CALL user_own_shutdownrequest(SscServerT h, SscNotFunT what, void
    *userdata);
{
  if (shutdown not needed) {
    return SSC_ABORT;
  }
  return SSC_CONT; /* シャットダウンに進む */
}
```

サーバーがシャットダウンされる前に user_own_shutdownrequest が呼び出されるように指定するには、SSCSetNotifier 関数を次のように呼び出します。

```
SSCSetNotifier(handle, SSC_NOTIFY_SHUTDOWN, user_own_shutdownrequest, NULL);
```

注:

関数 user_own_shutdownrequest から SSC_ABORT が返された場合はシャットダウンが許可されず、SSC_CONT が返された場合はシャットダウンに進むことができます。

SSCSetState

SSCSetState は、リンク・ライブラリー・アクセス・サーバーの状態を設定します。これにより、サーバーが後続の接続を受け付けるかどうか、またサーバーがプリフェッチを使用するかどうかを制御できます。

サーバーを「open」に設定すると、サーバーは接続を受け付けるようになります。サーバーを「closed」に設定すると、サーバーは以降の接続を受け付けなくなります (これにはローカル接続とリモート接続の両方が該当します)。ただし、既に作成されている接続は引き続き使用できます。

プリフェッチをオンにすると、サーバーはすぐに参照される可能性が高いデータを「先読み」によってフェッチします。プリフェッチには、追加のメモリーまたはディスク・キャッシュ・スペースが必要です。プリフェッチをオンにすると、一般にはパフォーマンスが向上します。プリフェッチをオフにすると、必要なメモリーが少なくなります。サーバーの大規模な順次スキャンを必要とする照会がある場合は、プリフェッチをオンにすると非常に役立ちます。例えば、レポートまたは集約関数を使用してデータベース全体 (またはその大部分) の値を取得する場合は、プリフェッチが役立ちます。すべての照会が 1 件または数件のレコードしか必要としないのであれば、通常プリフェッチは役立ちません。プリフェッチはメモリーを使い

果たすため、使用可能メモリーの少ないシステムでは、プリフェッチによって実際にパフォーマンスが低下することがあります。

プリフェッチをいつ使用すればよいかを判断するには、以下のガイドラインが役立ちます。

プリフェッチの使用が適切である場合: 使用可能メモリー (またはディスク・キャッシュ・スペース) が大量にあり、照会に大規模な順次スキャンが必要である。

プリフェッチの使用が適切でない場合: 使用可能メモリーが少なく、ほとんどの照会で 1 度に 1 回無関係なレコードが読み取られる。

構文

```
SscRetT SSC_CALL SSCSetState(SscServerT h,SscStateT runflags)
```

SSCSetState 関数で使用されるパラメーターは以下のとおりです。

表 15. SSCSetState 関数のパラメーター

パラメーター	使用タイプ	説明
h	in、use	サーバーへのハンドル。
runflags	in	SSC_STATE_OPEN フラグ (新たなリモート接続を許可) および SSC_STATE_PREFETCH フラグ (必要に応じてユーザーがサーバーにプリフェッチの実行を許可) の組み合わせを指定できます。指定可能な組み合わせの例を以下に示します。 • サーバーを open に設定: state = state | SSC_STATE_OPEN; • サーバーを closed に設定: state = state & ~SSC_STATE_OPEN; • プリフェッチを on に設定: state = state | SSC_STATE_PREFETCH; • プリフェッチを off に設定: state = state & ~SSC_STATE_PREFETCH;

戻り値

- SSC_SUCCESS - 操作が正常に終了しました。
- SSC_ERROR - 操作が失敗しました。
- SSC_SERVER_INNETCOPYMODE - サーバーはネットコピー・モードです (HotStandby のみ)。
-

SSC_SERVER_NOTRUNNING - サーバーが稼働していません。

コメント

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND があります。以下のコマンドです。

ADMIN COMMAND 'close';

SSCStartDisklessServer

SSCStartDisklessServer は、リンク・ライブラリー・アクセスを使用するディスクレス・サーバーを始動します。

構文

```
SscRetT SSC_CALL SSCStartDisklessServer (int argc, char* argv[ ],
    SscServerT * h, SscStateT runflags, char* lic_string, char* ini_string);
```

SSCStartDisklessServer 関数で使用されるパラメーターは以下のとおりです。

表 16. SSCStartDisklessServer のパラメーター

パラメーター	使用タイプ	説明
argc	in	コマンド行引数の数。
argv	in、use	関数呼び出しで使用されるコマンド行引数の配列。引数 argv[0] は、ユーザー・アプリケーションのパスとファイル名用として予約されており、必ず指定する必要があります。有効な引数のリストについては、下に示されている SSCStartDisklessServer のパラメーター・オプションを参照してください。
h	out	始動されたサーバーへのハンドルを返します。このハンドルは、他の制御 API 関数でサーバーを参照する際に必要となります。
runflags	in	このパラメーターには以下のオプションのみを指定できます。 SSC_STATE_OPEN - リモート接続が許可されます。 runflags = SSC_STATE_OPEN
lic_string	in	solidDB ライセンス・ファイルを含んだストリングを指定します。
ini_string	in	solidDB 構成ファイルを含んだストリングを指定します。

SSCStartDisklessServer パラメーターのオプション

以下に示すのは、*argv* パラメーターのコマンド行オプションです。

表 17. argv パラメーターのコマンド行オプション

オプション	説明
-h	ヘルプを表示します。
-n *name*	サーバー名を設定します。
-U *username*	データのユーザー名を指定します。ユーザー名では大/小文字が区別されません。ユーザー名には少なくとも 2 文字が必要です。ユーザー名の最大文字数は 80 です。ユーザー名は英字またはアンダースコアーで始まっていなければなりません。使用できる文字は、英小文字の a から z、英大文字の A から Z、アンダースコアー文字 (_)、および 0 から 9 の数字です。 **注:** ユーザー名を忘れると、solidDB に接続できません。デフォルトのユーザー名はありません。データベース作成時に入力するユーザー名が、その新しいデータベースに接続する際に使用できる唯一のユーザー名となります。
-P *password*	データに特定のパスワードを指定します。パスワードでは大/小文字が区別されません。パスワードには少なくとも 3 文字が必要です。パスワードは、英字、アンダースコアー、または数字で始めることができます。使用できる文字は、英小文字の a から z、英大文字の A から Z、アンダースコアー文字 (_)、および 0 から 9 の数字です。
-C *catalogname*	データのカタログ名を指定します。サーバーを初めて始動する場合に必要です。このパラメーターを指定するときは、必ず英大文字の C を使用してください。カタログについて詳しくは、「*solidDB SQL ガイド*」の『データベース管理のための solidDB SQL の使用』という章の『データベース・オブジェクトの管理』を参照してください。
-x ignoreerrors	索引エラーを無視します。

戻り値

- SSC_SUCCESS - サーバーが始動されました。
- SSC_ERROR - サーバーの始動に失敗しました。
- SSC_SERVER_INNETCOPYMODE - サーバーはネットコピー・モードです (HotStandby のみ)。

- SSC_INFO_SERVER_RUNNING - サーバーは既に稼働しています。
- SSC_INVALID_HANDLE - 無効なローカル・サーバー・ハンドルが指定されました。
- SSC_INVALID_LICENSE - ライセンスがないか、無効なライセンス・ファイルが見つかりました。

コメント

デフォルトでは、状態は SSC_STATE_OPEN に設定されます。

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND がありません。

例

SSCStartDisklessServer

```
SscStateT runflags = SSC_STATE_OPEN;
SscServerT h;
char* argv[4]; /* 4 つのパラメーター・ストリングへのポインター */
int argc = 4;
char* lic = get_lic(); /* ライセンスを取得 */
char* ini = get_ini(); /* solid.ini を取得 */
SscRetT rc;
argv[0] = "appname"; /* ユーザー・アプリケーションのパスとファイル名 */
argv[1] = "-Udba"; /* ユーザー名 */
argv[2] = "-Pdba"; /* ユーザーのパスワード */
argv[3] = "-Cdba"; /* カタログ名 */
/* ディスクレス・サーバーを始動 */
rc = SSCStartDisklessServer(argc, argv, &h, runflags, lic, ini);
```

注:

この例の get_ini() と get_lic() は、ユーザーが作成する関数です。それぞれの関数で、solid.ini ファイル・テキストまたは solid.lic ライセンス・ファイルを含んだストリングを返す必要があります。

カタログ名を指定しないと、solidDB からエラーが返されます。

関連項目

SSCStopServer

51 ページの『4 章 ディスクレス機能の使用』も参照してください。

SSCStartServer

SSCStartServer は、リンク・ライブラリー・アクセスを開始します。マルチスレッド環境では、サーバーがクライアントとは別のスレッドで稼働します。アプリケーションが実行されている間は、アプリケーションで必要に応じてサーバー・サブルーチンを開始または停止できます。

3 番目のパラメーターは「out」パラメーターであることに注意してください。サーバーが正常に始動されると、SSCStartServer ルーチンはそのサーバーのハンドルを指すようにこのパラメーターを設定します。

注:

ディスクレス・サーバーを始動する場合は、制御 API 関数 SSCStartDisklessServer を使用する必要があります。 41 ページの『SSCStartDisklessServer』を参照してください。

構文

```
SscRetT SSC_CALL SSCStartServer(int argc, char* argv[], SscServerT* h
    SscStateT runflags)
```

SSCStartServer 関数で使用されるパラメーターは以下のとおりです。

表 18. SSCStartServer のパラメーター

パラメーター	使用タイプ	説明
argc	in	コマンド行引数の数。
argv	in、use	コマンド行引数の配列。有効な引数のリストについては、 43 ページの『SSCStartServer』を参照してください。
h	out	始動されたサーバーへのハンドルを返します。このハンドルは、他の制御 API 関数でサーバーを参照する際に必要となります。
runflags	in	以下のいずれか、または両方を指定できます。 • SSC_STATE_OPEN - リモート接続が許可されます。 • SSC_STATE_PREFETCH - サーバーが必要に応じてプリフェッチを実行します。 例: `runflags = SSC_STATE_OPEN & SSC_STATE_PREFETCH`

戻り値

- SSC_SUCCESS - サーバーが始動されました。
- SSC_ERROR - サーバーの始動に失敗しました。

- SSC_ABORT
- SSC_BROKENNETCOPY - 不完全なネットコピーが原因でデータベースが破損しました。
- SSC_FINISHED
- SSC_CONT
- SSC_CONNECTIONS_EXIST
- SSC_UNFINISHED_TASKS
- SSC_INVALID_HANDLE - 無効なローカル・サーバー・ハンドルが指定されました。
- SSC_INVALID_LICENSE - ライセンスがないか、無効なライセンス・ファイルが見つかりました。
- SSC_NODATABASEFILE - データベース・ファイルが見つかりませんでした。
- SSC_SERVER_NOTRUNNING
- SSC_INFO_SERVER_RUNNING - サーバーは既に稼働しています。
- SSC_SERVER_INNETCOPYMODE - サーバーはネットコピー・モードです (HotStandby のみ)。
- SSC_DBOPENFAIL - データベースを開く操作に失敗しました。
- SSC_DBCONNFAIL - データベースへの接続に失敗しました。
- SSC_DBTESTFAIL - データベース・テストが失敗しました。

- SSC_DBFIXFAIL - データベースの修正に失敗しました。
- SSC_MUSTCONVERT - データベースを変換する必要があります。
- SSC_DBEXIST - データベースが存在します。
- SSC_DBNOTCREATED - データベースが作成されていません。
- SSC_DBCREATEFAIL - データベースの作成に失敗しました。
- SSC_COMINITFAIL - 通信の開始に失敗しました。
- SSC_COMLISTENFAIL - 通信の listen に失敗しました。
- SSC_SERVICEFAIL - サービスの操作に失敗しました。
- SSC_ILLARGUMENT - コマンド行引数が正しくありません。
- SSC_CHDIRFAIL - ディレクトリーの変更に失敗しました。
- SSC_INFILEOPENFAIL - 入力ファイルを開く操作に失敗しました。
- SSC_OUTFILEOPENFAIL - 出力ファイルを開く操作に失敗しました。
- SSC_SRVCONNFAIL - サーバーの接続に失敗しました。
- SSC_INITERROR - 操作の開始に失敗しました。
- SSC_CORRUPTED_DBFILE - アサートまたはその他の致命的エラーです。
- SSC_CORRUPTED_LOGFILE - アサートまたはその他の致命的エラーです。

コメント

デフォルトでは、状態は SSC_STATE_OPEN に設定されます。

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND がありません。

solidDB サーバーを新たに始動するときは、solidDB で新しいデータベースを作成するように明示的に指定する必要があります。そのためには、関数 SSCStartServer() を、-U *username* -P *password* -C *catalogname* (デフォルトのデータベース・カタログ名) の各パラメーターを指定して実行します。詳しくは、18 ページの『制御 API 関数 SSCStartServer による明示的な始動』を参照してください。

データベース・サーバーを再始動する場合 (つまりデータベースが既にディレクトリーに存在する場合) は、SSCStartServer で既存のデータベースが使用されます。

SSCStartServer 関数は、複数のスレッドを作成してサーバー・タスクを実行することがあります。サーバー・タスクには、ローカルおよびリモートのクライアント要求の処理だけでなく、チェックポイントやマージなどの各種バックグラウンド・タスクの実行も含まれます。

関連項目

SSCStopServer

SSCStopServer

SSCStopServer は、リンク・ライブラリー・アクセス・サーバーを停止します。

暗黙的な方法 (SQLConnect など) で始動されたサーバーは、明示的な方法 (SSCStopServer など) でシャットダウンできます。ただし、その逆は不可能です。例えば、SSCStartServer で始動されたサーバーを SQLDisconnect で停止することはできません。

アプリケーションが実行中にサーバーを開始および停止できる回数は 1 回に制限されていません。サーバーが停止された後に、アプリケーションで SSCStartServer を使用してサーバーを再始動することができます。

構文

```
SscRetT SSC_CALL SSCStopServer(SscServerT h, bool force)
```

SSCStopServer 関数で使用されるパラメーターは以下のとおりです。

表 19. SSCStopServer のパラメーター

パラメーター	使用タイプ	説明
h	in、use	サーバーへのハンドル。

表 19. SSCStopServer のパラメーター (続き)

パラメーター	使用タイプ	説明
force	in	オプションは以下のとおりです。 • TRUE - あらゆる場合においてサーバーを停止します。 • FALSE - 開いている接続がない場合にサーバーを停止します。それ以外の場合は停止が失敗します。

戻り値

- SSC_SUCCESS - サーバーが停止されました。
- SSC_CONNECTIONS_EXIT - 開いている接続があります。
- SSC_UNFINISHED_TASKS - 実行中のタスクがあります。
- SSC_ABORT
- SSC_ERROR

コメント

リモート・ユーザーは、ADMIN COMMAND 'shutdown' を使用することで solidDB を停止できます。詳しくは、 65 ページの『リンク・ライブラリー・アクセスのパラメーター』を参照してください。

FALSE オプションを指定すると、データベースまたは既存のユーザーとの接続が開いている場合に、シャットダウンが許可されなくなります。このオプションは、solidDB SQL 拡張 ADMIN COMMAND 'shutdown' に相当します。

SSCSetState() に &~SSC_STATE_OPEN オプションを指定すると、solidDB との新たな接続を作成できなくなります。

関連項目

SSCStartServer

SSCSetState

SSCUnregisterThread

SSCUnregisterThread は、サーバーに対する solidDB アプリケーション・スレッドの登録を抹消します。SSCUnregisterThread 関数は、 SSCRegisterThread 関数で自己を登録したすべてのスレッドで呼び出す必要があります。この関数は、スレッドが終了する前に呼び出されます。

構文

```
SscRetT SSC_CALL SSCUnregisterThread(SscServerT h)
```

SCCUnregisterThread 関数で使用されるパラメーターは以下のとおりです。

表 20. SCCUnregisterThread のパラメーター

パラメーター	使用タイプ	説明
h	in、use	サーバーへのハンドル

戻り値

- SSC_SUCCESS
- SSC_INVALID_HANDLE

コメント

SSC_CALL は、ユーザー関数の呼び出し規則を明示的に定義するために必要です。SSC_CALL は、sscapi.h ファイルで各プラットフォームに合わせて定義されます。

この関数には、対応する solidDB SQL 拡張 ADMIN COMMAND がありません。

関連項目

SSCRegisterThread

4 ディスクレス機能の使用

solidDB リンク・ライブラリー・アクセスを使用すると、ディスク・ストレージ・スペースなしで稼働するデータベース・エンジンを作成できます。この機能は、ネットワーク・ルーターやスイッチ内のライン・カードのような、ハード・ディスクを持たない組み込みシステムで役立ちます。

ディスクレス・サーバーは、主に 2 つの方法で実行されます。つまり、単独で実行されるか、拡張レプリケーション・システムのレプリカとして実行されます。いずれの場合も、リンク・ライブラリー・アクセスの関数呼び出し SSCStartDisklessServer() を使用してサーバーを始動します。

ディスクレス・サーバーを単独で使用する場合

ディスクレス・サーバーを単独で実行する場合は、始動時にデータを読み取ることも、シャットダウン時にデータを書き込むこともできません。つまり、サーバーは以前のデータがない状態で毎回始動されます。

さらに、サーバーはディスクにデータを書き込むことができないため、サーバーが正常にシャットダウンされなかった場合 (電源障害などにより)、サーバーのデータはすべて失われ、リカバリーできません。データ損失のリスクを軽減するには、solidDB の HotStandby コンポーネントを使用して、データのコピーを格納する「ホット・スタンバイ」マシンを作成します。このホット・スタンバイ機能について詳しくは、「*solidDB 高可用性ユーザー・ガイド*」を参照してください。

ディスクレス・サーバーを拡張レプリケーション・システムの一部として使用する場合

ディスクレス・サーバーは、拡張レプリケーション・システムのレプリカとして使用できます。この場合、レプリカはマスター・サーバーにデータを送信し、そのマスター・サーバーからデータをダウンロードできます。したがって、レプリカにディスク・ストレージやその他の専用永続ストレージがなくても、拡張レプリケーション・システム内でそのデータの一部または全部を永続化できます。

ディスクレス・サーバー用の構成パラメーター

このセクションでは、ディスクレス・サーバーのインプリメントと保守に関連するパラメーター設定について説明します。

ディスクレス・サーバーで使用されるパラメーター

以下に示す構成ファイルの各セクションには、ディスクレス・サーバー用の設定があるパラメーターが含まれています。

Index File セクション

索引ファイルに影響する構成パラメーターを以下に示します。

FileSpec_[1...N] パラメーター

FileSpec パラメーターでは、データベース・ファイルの名前と最大サイズを指定します。メイン・メモリー・エンジンに対して最大サイズをバイト単位で定義する場合は、FileSpec パラメーターで以下の引数を指定します。

- データベース・ファイル名 - ディスクレス・サーバーでは物理データベース・ファイルが作成されないのでこのパラメーターは使用されませんが、この引数にダミー値を指定する必要があります。
- 最大ファイル・サイズ - この設定は必要です。ディスクレス・サーバー内の全データを格納できるサイズをバイト単位で指定する必要があります。最大ファイル・サイズは、CacheSize パラメーターで設定されるキャッシュ・サイズよりも小さくする必要があることに注意してください。

FileSpec パラメーターのデフォルト値は solidr.db および 5000000 バイトです。以下に例を示します。

```
FileSpec_1=SOLIDR.db 5000000
```

注:

複数のファイルを指定する場合は、すべての FileSpec パラメーター設定の合計が、最大ファイル・サイズの設定となります。

ディスクレス・マシンには仮想メモリー用のスワップ・スペースとして使用するディスクがないため、最大サイズは使用可能な物理メモリーで制限されます。一部のプラットフォームでは、アプリケーションで使用可能な物理メモリーの量がマシンの物理メモリーの量より少ない場合があることに注意してください。例えば、32 ビット・システムで稼働する Linux の一部のバージョンでは、アプリケーションで使用可能なメモリー量が、理論上のアドレス・スペース (4 GB) の 2 分の 1 または 4 分の 1 に制限されます。これは、アドレスの最上位ビットの 1 つまたは 2 つが Linux 自体のメモリー管理用として確保されるためです。

メモリー内のデータが最大ファイル・サイズを超えると、エラー・メッセージ 11003 が表示されます。

```
File write failed, configuration exceeded
```

CacheSize

CacheSize パラメーターでは、サーバーがバッファー・キャッシュに割り振るメイン・メモリーの量をバイト単位で定義します。以下に例を示します。

```
CacheSize=10000000
```

この値の設定は、ディスクレス・サーバーに関する以下の基準によって決まります。

- ディスク・ベース表の場合は、キャッシュ・サイズ (バイト単位) を FileSpec パラメーターで設定されている最大ファイル・サイズ (つまりデータの量) よりも少

なくとも 20% 大きくする必要があります。これは、このデータがバッファー・キャッシュに保持されるためです。この 20% のオーバーヘッドは見積もりであり、データベースの使用状況によって変わる可能性があります。以下に例を示します。

```
[IndexFile]
FileSpec_1=solid.db 10MB
CacheSize=12MB
```

- ディスク・ベース表が使用されない場合でも (インメモリー表を使用してデータベースが作成されている場合)、システム表を保持するためにキャッシュは必要です。その場合、最小キャッシュ・サイズは 1 から 2 MB です。システム表が占有するスペースは、データベース・オブジェクトの数と複雑度、および拡張レプリケーションが使用されているかどうかによって決まります。
- キャッシュ・サイズは、ディスクレス・サーバーの実行に使用できる物理メモリーよりも小さくする必要があります。

  ディスクレス・サーバーが使用する合計メモリー量は、以下のように試算できます (これらすべてを加算した合計は、使用可能な物理メモリー量の範囲内であることが必要です。つまり、キャッシュ・サイズはサーバーで使用可能な物理メモリー量よりも大幅に小さくしなければなりません)。

  ```
  CacheSize
  + 5 MB
  + (100 K * ユーザー数 * ユーザー 1 人あたりのアクティブなステートメント数)
  + インメモリー表のスペース
  + (2 次サーバーに送信される HSB 操作) [1][2]
  ```

  [1] 式のこの項は、HotStandby のユーザーにのみ適用されます。HSB 1 次サーバーは、2 次サーバーに送信される HotStandby 操作を保管するためにある程度のメモリーを必要とします。1 次サーバーと 2 次ディスクレス・サーバーの間で一時的なネットワーク障害が発生すると、1 次サーバーがアプリケーションからのトランザクションを継続して受け付けることがあります。サーバー間のネットワーク接続が復元すると、1 次サーバーから 2 次サーバーに更新情報が送信されます (HotStandby では、トランザクション・ログを使用してこれらの操作が保管されます。ディスクレス・サーバーではトランザクション・ログをディスクに書き込むことができないため、この情報をメモリーに保管する必要があります)。このメモリーはキャッシュとは別のものです。

  [2] 式のこの項については、現時点で 1 MB または 512 件の操作 (どちらか小さい方) が上限となっています。ディスク・ベースのサーバーとは異なり、使用可能なスペースを使い切るまでトランザクション・ログが拡大することは許可されていません。

  正確な必要量は、サーバーに対して実行される照会の性質をはじめとするその他の要因にも左右されます。物理メモリーはオペレーティング・システムなどによってある程度占有されるので、当然ながらサーバーで使用できるメモリー量は総物理メモリー量よりも少なくなります。

Com セクション

以下の構成パラメーターは、マスター・サーバーとディスクレス・レプリカ・サーバー (ディスクレス・サーバーを拡張レプリケーションのレプリカ・サーバーとして使用する場合) との間の通信に影響します。

Listen パラメーター [Com]

このパラメーターは、ディスクレス・サーバーがネットワークの listen を開始するときに使用するプロトコルと名前です。デフォルト値は、オペレーティング・システムによって異なります。「*solidDB 管理者ガイド*」の『ネットワーク接続の管理』を参照してください。

ディスクレス・エンジンに適用されない構成パラメーター

セクション別に分類された以下の構成ファイル・パラメーターは、ディスクレス・サーバーには無効であるか、または作用しません。これらのパラメーターは、ディスクレス・エンジンには適用されない動作に影響します。

表 21. ディスクレス・エンジンに適用されない構成パラメーター

パラメーター	説明
[General] セクション	
CheckpointInterval	ディスクレス・サーバーにはチェックポイントが適用されないため、このパラメーターは無効です。
[IndexFile] セクション	
ReadAhead	データベース・ファイルからの物理的な読み取りが行われないため、このパラメーターは作用しません。
PreFlushPercent	データベース・ファイルへの物理的な書き込みが行われないため、このパラメーターは作用しません。
[Logging] セクション	
LogEnabled	ディスクレス・サーバーではトランザクション・ロギングが常に無効となるため、このパラメーターは無効です。 **注:** ディスクレス・モードでは、トランザクションのロールバックのみがサポートされます。トランザクション・ロールバックは、一般に何らかの障害によって、途中まで完了したトランザクションが中断された場合に使用されます。ディスクレス・モードでは、ロールフォワード・リカバリーをサポートしません。

5 Java での solidDB リンク・ライブラリー・アクセスの使用

注:

この章は、これまでの章の内容について理解していることを前提としています。C/ODBC には関心がなく Java/JDBC だけに関心があるからといってこの章から読み始めると、十分な知識がないためにこの章全体を理解できないことになります。

solidDB JDBC アクセラレーター (SJA) の概要

Java/JDBC プログラムでは、C/ODBC プログラムと同様に、solidDB リンク・ライブラリー・アクセスを使用することでパフォーマンスを高め、サーバーを詳細に制御できます。SJA は、Java アプリケーションがローカルの solidDB サーバーを始動できるようにします。このサーバーは、ssolidacxx という動的ライブラリーから Java 仮想マシンのコンテキストにロードされるものです。これで Java アプリケーションは、solidDB サーバーに接続し、標準の JDBC API を通じて solidDB DBMS が提供するサービスを使用できるようになります。

クライアント・アプリケーション・プログラムのパフォーマンスが向上するのは、サーバー・ライブラリーに直接リンクされるのでサーバー関数を呼び出す際にネットワーク (RPC) 呼び出しのオーバーヘッドがかからないためです。また、アプリケーションでより詳細な制御が実現するのは、solidDB サーバー制御 (SSC) ライブラリーの関数 (メソッド) を呼び出して、一定のタイプのタスクに優先順位を割り当てることなどが可能となるためです。例えば、アプリケーションでそれ自体に高い優先順位を指定し、リモート・クライアント・アプリケーションに低い優先順位を指定することができます。

solidDB JDBC アクセラレーター (SJA) は、サーバーとクライアントがリンクされている場合にのみ使用できます。したがって、Java アプリケーションと solidDB サーバーが別々のホストで実行される場合は、SJA を使用できません。

当然ながら、「ローカル」クライアント (リンク・ライブラリー・アクセス・ライブラリーにリンクされているクライアント) だけが、ネットワークをバイパスし、リンク・ライブラリー・アクセスによるパフォーマンスの向上を実現できます。他のクライアント・プログラムもサーバーを使用できますが、ネットワークを介して接続する必要があります。また、solidDB サーバーと同じコンピューターで実行されていても「リモート」プログラムとして扱われます。ローカル・クライアントは 1 つしか作成できません。それ以外はリモートです。リモート・プログラムには C プログラムと Java プログラムが混在してかまいません。場合があります。

ローカル・クライアントを作成した言語によって、リモート・クライアントの作成に使用できる言語が制限されることはありません。例えば、JDBC アクセラレーターを使用する場合は、リモート・クライアント・プログラムに C、Java、またはその両方を使用できます。

アクセラレーターの動作

Java/JDBC プログラムでリンク・ライブラリー・アクセスを使用する場合は、C プログラムと同様に、プログラムを solidDB リンク・ライブラリー・アクセス・ライブラリー (ssolidacxx) にリンクする必要があります。このライブラリーには solidDB サーバー全体が、スタンドアロン実行可能プログラムではなく呼び出し可能なライブラリーの形式で格納されています。Java/JDBC で使用される ssolidacxx は、これまでの章で説明した ssolidacxx と同じです。つまり、Java クライアント用と C クライアント用に別々のバージョンが存在するわけではありません。ライブラリーにリンクすることで、クライアント・プログラムではネットワークを介した RPC (リモート・プロシージャー・コール) のオーバーヘッドを回避できます。

Java/JDBC でリンク・ライブラリー・アクセスを使用する場合は、以下の要素をリンクして 1 つの実行可能プロセスにします。

- solidDB リンク・ライブラリー・アクセス・ライブラリー
- Java 言語のクライアント・プログラム
- JVM

実行可能プロセスでのレイヤーは、上から順に以下のとおりです。

- ローカルの Java/JDBC クライアント・アプリケーション
- JVM (Java 仮想マシン)
- solidDB アクセラレーター・ライブラリー (ssolidacxx)

クライアントの Java コマンドは JVM によって実行されます。コマンドが JDBC 関数呼び出しである場合は、JVM によって ssolidacxx 内の適切な関数が呼び出されます。関数呼び出しは、ネットワーク (RPC) を介さずに直接実行されます。呼び出しには JNI (Java ネイティブ・インターフェース) が使用されます。これらの詳細に関して知る必要はありません。ユーザーは JNI コードを作成する必要はなく、リモート・クライアント・プログラムの場合と同じ JDBC 関数を呼び出すだけです。

Java アクセラレーターから solidDB データベースにアクセスする操作は RPC を介して solidDB データベースにアクセスする操作と同じですが、1 つ異なる点として、Java アクセラレーターを使用するアプリケーションでは、データベース・サービスにアクセスするためにまず solidDB リンク・ライブラリー・アクセス・サーバーを始動する必要があります。この処理には、SolidServerControl (SSC) と呼ばれるプロプラエタリー API を使用します。 SSC API 呼び出しは、solidDB DBMS を開始および停止するために使用されます。実際のデータベース接続は、標準の JDBC

API を使用して行われます。SolidServerControl API と solidDB の JDBC ドライバーは、どちらも SolidDriver2.0.jar という .jar ファイル内にあります。

ローカルの solidDB サーバーを始動すると、このサーバーが ssolidacxx という動的ライブラリーから Java 仮想マシンのコンテキストにロードされます。これで Java アプリケーションは、solidDB サーバーに接続し、標準の JDBC API を通じて solidDB DBMS が提供するサービスを使用できるようになります。

solidDB Java アクセラレーターを使用するローカル・クライアント・プログラムはいずれも、以下に示す基本的な 3 ステップのパターンをたどります。

1. SolidServerControl でアクセラレーター・サーバーを始動します。
2. 標準の JDBC API を使用してデータベースにアクセスします。
3. データベースの処理が完了すると、再び SolidServerControl でアクセラレーター・サーバーを停止します。

solidDB アクセラレーター・サーバーにアクセスするための SolidServerControl クラスは、solidDB JDBC ドライバー・ファイル内の solid.ssc パッケージに組み込まれています。solidDB JDBC ドライバーの jar ファイル (SolidDriver2.0.jar) には、以下のパッケージが含まれています。

- solid.jdbc.* solidDB JDBC ドライバー・クラス
- solid.ssc.* solidDB サーバー制御クラス (プロプラエタリー・インターフェース)

solidDB サーバー制御 (solid.ssc) パッケージには以下のクラスが含まれています。

- SolidServerControl (Java から solidDB サーバーを始動および停止する場合に使用)
- SolidServerControlInitializationError (エラーを報告する場合に使用)

SolidServerControl (SSC) クラス・インターフェースについて詳しくは、60 ページの『solidDB サーバー制御 (SSC) API』を参照してください。

solidDB サーバーを Java アプリケーションから始動するには、アプリケーションの冒頭で SolidServerControl クラスをインスタンス化し、適切なパラメーターを指定して startServer メソッドを呼び出す必要があります (以下の使用例を参照してください)。サーバーを始動した後は、JDBC でサーバーに接続できる状態になります。

システム要件

solidDB Java アクセラレーターを使用するには、以下が必要です。

- solidDB リンク・ライブラリー・アクセス・ライブラリー。これは ssolidacxx という名前のファイルです。ファイル名拡張子はプラットフォームによって異なります。よく使用される名前とプラットフォームの一部を以下に示します。
  - Microsoft Windows: ssolidacxx.dll および インポート・ライブラリー solidimpac.lib
  - Solaris および Linux: ssolidacxx.so
  - HP-UX: ssolidacxx.sl
- solidDB サーバーおよびリンク・ライブラリー・アクセスを使用するための有効なライセンス・ファイル
- solidDB JDBC2 ドライバー・ファイル (SolidDriver2.0.jar)
- プラットフォーム用の solidDB 通信ライブラリー (通常は solidDB Development Kit をインストールするとインストールされます)。
- Java Development Kit (JDK) 1.4.2 以降

基本的な使用法

インストール

Java Development Kit をインストールした場合は、それ以上のインストールは必要ありません。solidDB がインストールされていれば、solidDB Java アクセラレーターを使用する際に必要なライブラリーはそこに含まれています。

注: Java コンパイラーにアクセスするには、PATH 環境変数と CLASSPATH 環境変数を適切な値に設定することが必要となる場合があります。

プログラムのコンパイルおよび実行

サーバーが正常に始動するには、少なくとも solidDB およびリンク・ライブラリー・アクセスを使用するための有効なライセンスが必要です。

システム検索パスに ssolidacxx ダイナミック・リンク・ライブラリーが含まれている必要があります。以下の手順に従ってください。

1. パスを設定します (Microsoft Windows コマンド・プロンプトでの例)。

   set PATH=<path to your ssolidacxx DLL>;%PATH%

   このパスには、solidDB 通信ライブラリーを収容しているディレクトリーも含めるようにしてください。
2. PATH 環境変数に、JDK の HOTSPOT ランタイム環境を組み込みます (SJA は HotSpot JRE でのみテストされています)。以下に例を示します。

   set PATH=<your JDK directory>¥jre¥bin¥hotspot;%PATH%
3. この章の最後に掲載されているサンプル・ファイルを SJASample.java というファイルに保存し、以下のコマンドでコンパイルします。

javac -classpath <IBM solidDB JDBC driver directory>/SolidDriver2.0.jar;. ¥
SJASample.java

4. 以下のようなコマンド行でアプリケーションを実行します。

   java -Djava.library.path=<path to ssolidacxx DLL> ¥ -classpath <IBM soliDB JDBC driver directory>/SolidDriver2.0.jar;. ¥ <your application name>

   例えば、Microsoft Windows でサーバーを C:¥soliddb にインストールした場合、SJASample プログラムを実行するには以下のコマンド行を使用します。

   java -Djava.library.path=C:¥soliddb¥bin -classpath
   C:¥soliddb¥jdbc¥SolidDriver2.0.jar;. SJASample

   Microsoft Windows では、 ssolidacxx.dll 動的ライブラリーが、solidDB ルート・インストール・ディレクトリーの bin サブディレクトリーにあります。)

   SJASample サンプル・クラスのように、SolidServerControl の startServer メソッドで、solidDB サーバーに少なくとも以下のパラメーターを渡す必要があります。

   ```
   -c<directory containing solidDB license file>
   -U<username>
   -P<password>
   -C<catalog>
   ```

   英大文字と英小文字の「C」があり、それぞれ別のものを指していることに注意してください。

   必要なすべてのファイル (ssolidacxx ライブラリー、通信ライブラリー、JDBC ドライバー、および solid.lic) が現行作業ディレクトリーに配置されていれば、以下のコマンド行で SJASample を開始できます。

   java -Djava.library.path=. -classpath SolidDriver2.0.jar;. <your application>

   すべての作業が正常に終了していれば、solidDB アクセラレーター・サーバーが稼働状態となるはずです。

JDBC 接続の作成

solidDBJava アクセラレーターは、ローカル・データベース接続と RPC ベースの接続の両方をサポートします。

ローカルの (RPC ベースでない) JDBC 接続を作成するには、ポート 0 で「localserver」で使用する JDBC ドライバーを指定する必要があります。したがって、例えば JDBC クラス DriverManager を使用してデータベース接続を作成する場合は、以下のステートメントを使用して接続します (この後の SJASample サンプル・コードにも示されています)。

```
DriverManager.getConnection("jdbc:solid://localserver:0", myLogin, myPwd);
```

このように、DriverManager は URL *"jdbc:solid://localserver:0"* を使用して、ローカル・サーバーとの接続を作成します。getConnection サブルーチンに別の URL を指定すると、ドライバーはおそらく RPC を使用して接続しようとします。

したがって、Java アクセラレーター接続を作成するときは、

```
jdbc:solid://localserver:0
```

という URL を忘れないようにしてください。

注:

Java アプリケーション内で solidDB リンク・ライブラリー・アクセス・サーバーにアクセスする複数のスレッド (`java.lang.Thread` オブジェクト) を使用する場合は、各スレッドを使用して JDBC 関連のアクティビティーを開始する前に、そのスレッドを個別に solidDB リンク・ライブラリー・アクセス・サーバーに登録する必要があります。スレッドを登録するには、SolidServerControl API の registerThread メソッドをスレッドのコンテキストで呼び出します。スレッドの登録は、ユーザー・スレッド (メイン・スレッドを除く) ごとに solidDB の JDBC ドライバーを使用して明示的に行う必要があります。

また、ユーザーは、solidDB リンク・ライブラリー・アクセス・サーバーに登録されている各スレッドを明示的に登録抹消する必要があります。スレッドの登録を抹消するには、SolidServerControl API の unregisterThread 関数を呼び出します。

制限事項

- solidDB の「管理コマンド (admin command)」は、Java アクセラレーター・コンテキストでは機能しません。
- VM コンテキストの外部 (例えばネイティブ・メソッド呼び出しの内部) で何らかの障害が発生すると、Java が確実に動作しなくなります。solidDB サーバーのネイティブ・コードでアサート (あるいはクラッシュ) が発生した場合、Java は予期しない例外を通知して終了するか、または完全にハングアップします。ハングアップした場合は、付随する Java プロセスを手動で強制終了する必要があります。
- メモリー消費量を最小にするために、割り振られているすべてのステートメントをユーザーが明示的にドロップすることをお勧めします。つまり、割り振られているすべての JDBC ステートメント・オブジェクトを明示的に解放するために、close() メソッドを呼び出す必要があります。
- 複数の Java スレッドから同じステートメント・オブジェクトにアクセスすると、サーバーがクラッシュする場合があります。JDBC を使用する必要があるスレッドごとに、別々の JDBC 接続 (およびステートメント) を開く必要があります。

solidDB サーバー制御 (SSC) API

以下に示すのは、SolidServerControl クラスの完全なパブリック・インターフェースです。このクラスの一部のメソッドを使用するプログラムの例については、samples/accelerator_java/SJASample.java というファイルを参照してください。

```
        /**
        * 以下の定数について詳しくは
        * 「IBM solidDB リンク・ライブラリー・アクセス・ユーザー・ガイド」
        を参照してください。
*/
       public final static int SSC_SUCCESS = 0;
```

```
        public final static int SSC_ERROR = 1;
        public final static int SSC_ABORT = 2;
        public final static int SSC_FINISHED = 3;
        public final static int SSC_CONT = 4;
        public final static int SSC_CONNECTIONS_EXIST = 5;
        public final static int SSC_UNFINISHED_TASKS = 6;
        public final static int SSC_INVALID_HANDLE = 7;
        public final static int SSC_INVALID_LICENSE = 8;
        public final static int SSC_NODATABASEFILE = 9;
        public final static int SSC_SERVER_NOTRUNNING = 10;
        public final static int SSC_INFO_SERVER_RUNNING = 11;
        public final static int SSC_SERVER_INNETCOPYMODE = 12;

        public final static int SSC_STATE_OPEN     = (1 << 0);
        public final static int SSC_STATE_PREFETCH = (1 << 1);


/**
 * SolidServerControl クラスを開始します。出力はどの
 * PrintStream にも送信されません。
*
 * @戻り値               SolidServerControl インスタンス
 */
public static  SolidServerControl instance()
     throws SolidServerInitializationError;


/**
 * SolidServerControl クラスを開始します。出力はパラメーター
 * os で指定された PrintStream オブジェクトに送信されます。
*
 * @パラメーター os      出力用の PrintStream
 * @戻り値               SolidServerControl インスタンス
 *
 */
public static SolidServerControl instance( PrintStream os )
     throws SolidServerInitializationError;

/**
 * setOutStream メソッドは、出力を指定の PrintStream に設定します。
 *
 * @パラメーター os      出力用の PrintStream
 */
public void setOutStream( PrintStream os );


/**
 * getOutStream は SolidServerControl クラスの出力に使用される
 * ストリームを返します。
*
 * @戻り値               このオブジェクトの出力ストリーム
 */
public PrintStream getOutStream();


/**
 * startServer は solidSB リンク・ライブラリー・アクセス・サーバーを始動します。
 *
 * @パラメーター argv    アクセラレーター・サーバーのパラメーター・ベクトル
 *                   (solidDB ライセンス・ファイルが置かれている
 *                    作業ディレクトリー (-c¥tmp など) を必ず先に指定し、
       *             その後に他のパラメーターを指定してください)。
 *                   リンク・ライブラリー・アクセス・サーバーに渡すことのできる

 *                   パラメーターについて詳しくは、「IBM solidDB リンク・
 *                   ライブラリー・アクセス・ユーザー・ガイド」を参照してください。
*
```

```
 * @パラメーター runflags このパラメーターのオプションは、SSC_STATE_OPEN
 *                        (リモート接続を許可) および SSC_STATE_PREFETCH
 *                        (必要に応じてサーバーがプリフェッチを実行) です。
 *                        プリフェッチでは、表内容の先読み機能を備えたメモリー
 *                        およびディスク・キャッシュ、またはその両方が参照
 *                        されます。runflags パラメーターは以下のように
 *                        入力します。
 *                        runflags |= SSC_STATE_OPEN & SSC_STATE_PREFETCH
 *
 * @戻り値    サーバーからの戻り値は以下のとおりです。
 *              SSC_SUCCESS
 *              SSC_ERROR
 *              SSC_INVALID_LICENSE - ライセンスがないか、無効なライセンス・
 *              ファイルが見つかった場合
 *              SSC_NODATABASEFILE  - データベース・ファイルが見つからない場合
 */
public long startServer( String[] argv, long runflags );


/**
* stopServer は、solidDB リンク・ライブラリー・アクセス・サーバーを停止します。
 *
 * @パラメーター runflags   solidDB リンク・ライブラリー・アクセス・サーバーを
 *                  停止するための runflags。
       *                  詳しくは、「IBM solidDB リンク・ライブラリー・
 *                  アクセス・ユーザー・ガイド」を参照してください。
*
 * @戻り値    サーバーからの戻り値は以下のとおりです。
 *              SSC_SUCCESS サーバーが停止した場合
 *              SSC_CONNECTIONS_EXIT - 開いている接続がある場合
 *              SSC_UNFINISHED_TASKS - 実行するタスクが残っている場合
 *              SSC_SERVER_NOTRUNNING - サーバーが稼働していない場合
 */
public long stopServer( int runflags );


/**
 * サーバーの状態、つまりサーバーが稼働しているかどうかを返します。
 *
 * @戻り値 SSC_STATE_OPEN - サーバーが稼働状態にある場合
 */
public int getState();




/**
* registerThread は、このユーザー・スレッドを solidDB リンク・ライブラリー・
* アクセス・サーバーに登録します。
*
*
* @戻り値           サーバーからの戻り値は以下のとおりです。
*                    SSC_SUCCESS            登録が成功した場合
*                    SSC_ERROR              登録が失敗した場合
*                    SSC_INVALID_HANDLE     無効なローカル・サーバー・ハンドルが
                                            指定された場合
*                    SSC_SERVER_NOTRUNNING  サーバーが稼働していない場合
*/
public long registerThread();



/**
* unregisterThread は、solidDB リンク・ライブラリー・アクセス・サーバーから
* このユーザー・スレッドの登録を抹消します。
```

```
*
*
* @戻り値            サーバーからの戻り値は以下のとおりです。
*                     SSC_SUCCESS             登録が成功した場合
*                     SSC_ERROR               登録が失敗した場合
*                     SSC_INVALID_HANDLE      無効なローカル・サーバー・ハンドルが
                                              指定された場合
*                     SSC_SERVER_NOTRUNNING   サーバーが稼働していない場合
*/
public long unregisterThread();
```

付録. リンク・ライブラリー・アクセスのパラメーター

リンク・ライブラリー・アクセスのパラメーター

この付録では、リンク・ライブラリー・アクセスの全パラメーターのリストを示します。リンク・ライブラリー・アクセスのパラメーターは、solidDB 構成ファイル (solid.ini) の [Accelerator] セクションに記述されます。

それ以外のすべての solidDB パラメーターについては、「*IBM solidDB 管理者ガイド*」の該当する付録を参照してください。

solidDB のパラメーターは、以下の方法で変更できます。

- solidDB solsql で ADMIN COMMAND 'parameter' コマンドを入力する。
- solid.ini 構成ファイルを手動で編集する。

上記の方法で solid.ini ファイルに加えた変更は、次にサーバーを始動するときまで反映されないことに注意してください。

Accelerator セクション

表 22. Accelerator のパラメーター

[Accelerator]	説明	ファクトリー値
ImplicitStart	このパラメーターを yes に設定すると、ユーザー・アプリケーションで ODBC API 関数 SQLConnect が呼び出された時点で solidDB が自動的に始動されます。no に設定した場合は、制御 API 関数 SSCStartServer を呼び出して solidDB を明示的に始動する必要があります。	yes

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

C

F

I

L

M

O

S

特記事項

Solid Information Technology Ltd. または 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、および 7266702 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。

〒106-8711
東京都港区六本木 3-2-12
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。

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。

商標

IBM、IBM ロゴ、ibm.com®、Solid®、solidDB、InfoSphere™、DB2®、Informix®、および WebSphere® は、International Business Machines Corporation の米国およびその他の国における商標です。これらおよび他の IBM 商標に、この情報の最初に現れる個所で商標表示 (® または ™) が付されている場合、これらの表示は、この情報が公開された時点で、米国において、IBM が所有する登録商標またはコモン・ロー上の商標であることを示しています。このような商標は、その他の国においても登録商標またはコモン・ロー上の商標である可能性があります。現時点での IBM の商標リストについては、「Copyright and trademark information」(www.ibm.com/legal/copytrade.shtml) をご覧下さい。

Java およびすべての Java 関連の商標およびロゴは Sun Microsystems, Inc.の米国およびその他の国における商標です。

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。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

Printed in Japan

SC88-5819-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12