iAnywhere JDBC ドライバが最良の選択である理由

このマニュアルでは、iAnywhere JDBC ドライバの利点とアプリケーション開発者が利用を検討すべき理由について説明します。

SQL Anywhere は、Java アプリケーション向けに jConnect と iAnywhere JDBC ドライバの 2 種類のデータベース接続を提供しています。このマニュアルでは、iAnywhere JDBC ドライバの利点とアプリケーション開発者が利用を検討すべき理由について説明します。

背景

Sun は JDK 1.1 のときに JDBC を導入しました。JDBC は、Java クライアント・アプリケーションとデータベース間で通信を行うための API の仕様です。JDBC **ドライバ**は、データベース固有の通信に対する高水準の一般的な関数呼び出しを解釈する API のレイヤです。JDBC の仕様では、タイプ 1 の JDBC-ODBC ブリッジからタイプ 4 の pure Java ドライバまで、4 種類の JDBC ドライバが指定されています。
Sybase は、自社のデータベース・プラットフォーム用にタイプ 4 のドライバを提供した最初のベンダのうちの 1 社でした。jConnect 製品は、Sybase Adaptive Server Enterprise や SQL Anywhere などの Open Server 対応のサーバとの通信に TDS (表形式データ・ストリーム) プロトコルを使用します。

Java / JDBC 神話

Sun は、独自のタイプ 1 のドライバ Sun JDBC-ODBC ブリッジを JDBC 1.0 の一部として提供しました。これは、*sun.jdbc.odbc.JdbcOdbcDriver* クラスに実装されています。Java 開発コミュニティは、このドライバを **JDBC-ODBC ドライバ**と呼ぶことにしました。このドライバは、多くの ODBC ドライバとの間でパフォーマンス問題が発生したため、それ以来、すべてのタイプ 1 のドライバおよびブリッジ・ドライバの評判が悪くなりました。
多くの Java 開発者が、Java の「write once, run anywhere (一度書き込めば、どこでも動く)」という方針を維持するための解決策として、タイプ 4 のドライバである pure Java ドライバの使用に目を向けました。しかし、経験を積んだ多くの Java 開発者が証言するように、より良いパフォーマンス (または開発者のより良い生産性) を得るには、「どこでも動く」の方針はあきらめるべきです。たいていの場合、Java ソリューションは、1 つのプラットフォーム (アプリケーション・サーバ) またはもう 1 つのプラットフォーム (データ・ストア) に関連付けられているものです。

SQL Anywhere Studio における JDBC

SQL Anywhere Studio バージョン 7 では、2 つのメイン GUI ツール (Sybase Central および Interactive SQL) がデフォルトで jConnect 4.2 を使用して SQL Anywhere データベースに接続しました。バージョン 7 よりも前は、C/C++ ベースのツールが直接 ODBC を使用しました。SQL Anywhere Studio 7 は、クライアント・アプリケーションでの使用のために jConnect 5.2 もインストールしました。

SQL Anywhere Studio のコンポーネントの 1 つである Mobile Link 同期サーバは、ODBC を使用してさまざまな統合データベース (SQL Anywhere, Adaptive Server Enterprise, Oracle, SQL Server, and IBM DB2) との通信を行います。バージョン 8 では、Mobile Link の通信レイヤを簡素化するために、iAnywhere は iAnywhere *JDBC ドライバ* の ianywhere.ml.jdbcodbc.Idriver を開発しました。
Studio のメイン GUI ツールは、最初にリリースした iAnywhere JDBC ドライバを介した接続用に jConnect (バージョン 5.5) の使用を継続しました。SQL Anywhere Studio 8 は、クライアント・アプリケーションでの使用のために jConnect 4.5 もインストールしました。
SQL Anywhere Studio 9 では、GUI ツールがデフォルトで iAnywhere JDBC ドライバを使用します。クライアント・アプリケーションでの使用のために jConnect 5.5 がインストールされるため、jConnect を使用するためのオプションもまだ存在します。
SQL Anywhere Studio 10 では、iAnywhere *JDBC ドライバ* が JDBC 3.0 に準拠するようにアップグレードされましたが ("ianywhere.ml.jdbcodbc.jdbc3.IDriver")、下位互換性のために JDBC 2.0 準拠のドライバも含まれています。SQL Anywhere 10 には、JDBC 3.0 準拠の jConnect 6.0.5 も含まれています (jConnect 5.5 は JDBC 2.0 準拠)。また、バージョン 10 では、GUI ツールは JConnect を使用した SQL Anywhere への接続を許可しなくなりました。

ドライバのタイプ

4 種類の JDBC ドライバの説明を見ると、iAnywhere JDBC ドライバはタイプ 1 かタイプ 2 のドライバであると言えます。これらの主な違いは、ODBC が実質的にリレーショナル・データベースへのネイティブ・インタフェースであるかどうかということのようです。ODBC は SQL Anywhere 用のネイティブ・インタフェースであるため、iAnywhere JDBC ドライバはタイプ 2 のドライバになります。[1] [2]

与えられた選択肢

SQL Anywhere Studio 10 のマニュアル『SQL Anywhere サーバ - プログラミング』には、「JDBC ドライバの選択」というページがあります。このページでは、jConnect と iAnywhere JDBC ドライバについての説明があり、さらにこれらのドライバの良い点と悪い点が比較されています。SQL Anywhere 製品マニュアル SQL Anywhere サーバ - プログラミング > SQL Anywhere JDBC API > JDBC の概要
100% の pure Java ドライバまたは TDS 通信が絶対要件である環境では、jConnect ドライバが適しています。
それ以外の環境では、iAnywhere JDBC ドライバを選択した方が良いです (*ほとんどの場合*)。

jConnect が使用する TDS プロトコルは SQL Anywhere のサポート対象ですが、一般的なプロトコルであり、元々は Adaptive Server Enterprise (ASE) 用に記述されたものであるため、SQL Anywhere 固有の機能をサポートしません。SQL Anywhere が使用するもう一方のネイティブ通信プロトコルは、SQL Anywhere で使用するために記述されているため、TDS よりもかなり効率的です。SQL Anywhere ODBC ドライバは、このプロトコルを使用します。このプロトコルには、SQL Anywhere 専用のいくつかの追加機能があります。

- TDS はローカル接続でも TCP/IP に限定されますが、iAnywhere JDBC ドライバは TCP/IP、SPX、HTTP(s)、および共有メモリを使用できます。
- RSA、RSA-FIPS、ECC 暗号化技術などの強力な暗号化のサポートを提供します。

- 結果セットのプリフェッチ・キャッシュはしますが、TDS はプリフェッチしません。
- サーバ側カーソルの完全サポートを提供します。jConnect は、サーバ側カーソルをサポートしません。jConnect は、クライアントが結果セットから少しの行数しか使用しない場合でも、ネットワークを介して結果セット全体を取得することによりクライアント側でカーソルを実装します。
- アプリケーション情報の完全サポートを提供します。jConnect は、アプリケーション情報の詳細を省略します。
- ローカル・データベース・サーバの自動起動／停止機能を提供します。TCP/IP 接続では、サーバの自動起動／停止をすることはできません。
- バッチ SQL 文が豊富です (ワイド・インサートおよびワイド・フェッチ)。jConnect はワイド・フェッチのみをサポートします。
- Windows プラットフォームでの統合ログインが可能です。

実証的証拠から、大量のデータベース・トラフィックが発生するアプリケーションでは、iAnywhere JDBC ドライバが jConnect よりもはるかに良いパフォーマンスを示すことがわかっています。多くの顧客が、パフォーマンスが 2 ～ 3 倍改善されたと報告しています。iAnywhere 内部のエンジニアリング・テストでは、最大 7 倍のパフォーマンス改善が見られたものもありました。
jConnect を使用して SQL Anywhere に接続する場合、多くの SQL Anywhere 開発者が好ましくないと考える Adaptive Server Enterprise 互換モードで接続が開始されます。たとえば、Adaptive Server Enterprise には空の文字列という概念がないため、jConnect を使用して空の文字列を選択すると、実際にはアプリケーションが 1 個のブランクで構成される文字列を取得します。最近まで、アプリケーションが空の文字列と 1 個のブランクで構成される文字列と NULL 文字列を区別できませんでした。比較的新しいバージョンの jConnect は、要求においてステータス・バイトを使用することで、NULL と空の文字列と空でない文字列を区別できます。当然、jConnect がこのステータス・バイトを使用できるようにするために、アプリケーション側で何らかの追加作業を行う必要があります。
もう 1 つは、DATETIME 値の精度の問題です。SQL Anywhere はマイクロ秒の精度をサポートしますが、Adaptive Server Enterprise は 1/300 秒の精度をサポートします。したがって、jConnect は 1/300 秒の精度でしか処理できません。

標準の JDBC ドライバ

SQL Anywhere 10 では、iAnywhere JDBC 2.0 ドライバは *ianywhere.ml.jdbcodbc.IDriver* クラスに実装されています。それに対し、JDBC 3.0 は *ianywhere.ml.jdbcodbc.jdbc3.IDriver* クラスに実装されています。このクラスは、SQL Anywhere Studio のインストール先の *java/jodbc.jar* ファイル内にあります。
このドライバを使用するには、通常の JDBC コーディング・パターンに従います。

1. データ・ソースへの接続を示す JDBC URL を作成します。
2. Class.forName() を使用して、データベース・ドライバ・クラスを読み込みます。
3. URL を使用して、JDBC ドライバ・マネージャをデータ・ソースに接続します。

URL のフォーマット:

```
jdbc:ianywhere:driver=[ODBC_DRIVER];[CONNECTION_STRING]
```

[*ODBC_DRIVER*] は、ODBC ドライバのファイル名かまたはファイル・システムのフル・パスです (ドライバ名は、Microsoft ODBC Driver Manager などの ODBC ドライバ・マネージャがネーミング・ドライバを

サポートするプラットフォームでのみ使用可能)。
サポート対象の Linux/Unix プラットフォームでは、バージョン 10 以降の SQL Anywhere には独自の ODBC ドライバ・マネージャが付属しているため、Microsoft ODBC Driver Manager の場合と同様に接続文字列でドライバ名を使用できます。
[*CONNECTION_STRING*] は、セミコロンで区切られたフィールドを持つ、有効な SQL Anywhere 接続文字列です。詳細については、SQL Anywhere Studio 9 のオンライン・マニュアルの索引から「connection strings」を参照してください。

JDBC の URL プレフィックス

JDBC の URL プレフィックスが旧バージョンのものから変更されました。Java 6 JDK (1.6) を使用する場合、Class.forName() コマンドが iAnywhere JDBC ドライバを参照するにもかかわらず、"jdbc:odbc" プレフィックスが Sun JDBC-ODBC ブリッジのデフォルトになります。そうならないようにするために、Sybase iAnywhere は個別のプレフィックスを追加し、iAnywhere JDBC ドライバを正しく指定しました。現在のプレフィックスは "jdbc:ianywhere" です。
9.0.2.3441、10.0.0.2797、10.0.1.3415 以降のビルドおよび将来のソフトウェア・バージョンでは、新しい URL プレフィックスを使用してください。

SQL Anywhere 以外にも使用可能

iAnywhere JDBC ドライバは、SQL Anywhere データベース接続専用ではありません。このドライバは ODBC 全般を扱うので、URL で ODBC ドライバを正しく指定すれば、ほとんどすべての ODBC データ・ソースに接続できます。
Mobile Link 同期サーバは、このドライバを使用して、SQL Anywhere ODBC ドライバかまたは SQL Anywhere Studio に付属しているいずれかのサード・パーティ製ドライバを使用するサポート対象の統合データベースに接続します。
iAnywhere JDBC ドライバは、他の JDBC ドライバが特定の RDBMS に対して理想的なパフォーマンスを提供できず、ODBC ドライバの方が良い性能を発揮することで知られる SQL Anywhere 以外の環境でも使用されています。

コードの例

以下のクラスは、データベースに接続する際の標準的な JDBC の 3 段階プロセスです。このプログラムが iAnywhere JDBC ドライバを使用し、iAnywhere JDBC ドライバが次に SQL Anywhere ODBC ドライバを使用します。

```
//―――――― Start of TestIasJdbc.java ――――――――――――――――――
import java.sql.*;
import java.io.File;
public class TestIasJdbc {
  public static String USAGE = "¥n¥n" +
      "usage: java TestIasJdbc <DB_FILE> <ODBC_DRIVER>¥n" +
      " where <DB_FILE> is the full path to a SQL ¥n" +
     " Anywhere database
```

```
    " and <ODBC_DRIVER> is the name¥n" +
    " or path to the driver¥n"
public static void main(String args[]) {
    String driver, url, dbFile, odbcDriver;
    Connection conn;
    dbFile = checkArgs( args );
    odbcDriver = args[1];
    // Step 1: Create a JDBC URL
    //
    url = "jdbc:ianywhere:driver=" + odbcDriver +
          ";UID=DBA;PWD=SQL;DBF=" + dbFile;
        // Step 2: Load the database driver class
     //

            //For SQL Anywhere 10 (JDBC 3.0):
    driver="ianywhere.ml.jdbcodbc.jdbc3.IDriver";
     //For SQL Anywhere 10 (JDBC 2.0):
    driver="ianywhere.ml.jdbcodbc.IDriver";
     try {
        Class.forName( driver );
    } catch (Exception ex) {
        ex.printStackTrace();
        System.exit(3);
    }
    // Step 3: Have the JDBC driver manager connect to
    // the data source
    try {
        conn = DriverManager.getConnection( url );
        if ( conn == null ) {
            System.out.println("Connection failed!");
            System.exit(4);
        }
    } catch (Exception ex) {
        ex.printStackTrace();
        System.out.println("¥n¥nURL is [" + url + "]¥n");
        System.exit(5);
    }
    System.out.println("Successfully connected!");
    }
    public static String checkArgs( String args[] ) {
        if ( args.length != 2 ) {
            System.out.println(USAGE);
```

```
            System.exit(1);
        }
        if ( ! new File(args[0]).isFile() ) {
            System.out.println("¥nDB_FILE not a file: " + args[0]);
            System.out.println(USAGE);
            System.exit(2);
        }
        return args[0];
    }
}
//———————— End of TestIasJdbc.java ————————————————————
```

上記のコードを *TestIasJdbc.java* というファイルに保存します。

コードを 32 ビット Windows シェルでコンパイルして実行するコマンドは、以下のとおりです。

```
javac -classpath "%SQLANY10%¥java¥jodbc.jar" TestIasJdbc.java
java -classpath ".;%SQLANY10%¥java¥jodbc.jar" TestIasJdbc
    %SQLANY10%¥demo.db "SQL Anywhere 10"
```

コードを UNIX/Linux/Mac OS X シェルでコンパイルして実行するコマンドは、以下のとおりです。

```
javac -classpath "$SQLANY10/java/jodbc.jar" TestIasJdbc.java
java -classpath ".:$SQLANY10/java/jodbc.jar" TestIasJdbc
    "$SQLANY10/demo.db" "$SQLANY10/lib/libdbodbc9_r.so"
```

この例に関する注意事項:

- 上記のコードは、ODBC データ・ソース名 [DSN] を使用しない DSN レス接続です。iAnywhere JDBC ドライバは、DSN および DSN レス接続文字列の両方をサポートします。
- 上記のコードは、SQL Anywhere ODBC ドライバを使用して DBF を指定するため、クライアントが稼働しているサーバを見つけられない場合、指定されたデータベース・ファイル用のサーバを自動的に起動 (autostart) します。
- SQL Anywhere 接続パラメータの **log=sa_connect.txt** を URL に追加することにより、接続の問題が発生したときに情報を収集できます。プログラムの実行後、*sa_connect.txt* ファイルにクライアント・ライブラリからの接続デバッグ情報が保存されます。このログ・ファイルへのパスは、クライアント・アプリケーションが稼働している場所からの相対パスかまたはフル・ファイル・システム・パス (こちらの方が良い) を使用します。

pure Java ではない

SQL Anywhere のニュースグループ (※1) では、JDBC 接続に関する質問がよく寄せられています。これまでに多くの Java 開発者が jConnect を選び、現在、動作やパフォーマンスの問題に直面しています。開発者は、以下のような理由で jConnect を選んでいます。

- jConnect の使用経験があること
- Sybase 製品および Web 上にマニュアルが豊富にあること
- SQL Anywhere Studio のマニュアル『JDBC の概要』で最初に取り上げられていること
- 名前が「J」で始まること

- タイプ 4 の pure Java ドライバであること

これらの意見において、技術的なメリットは後回しになっています。確かに、pure Java コード・ベースを継続する利点はありますが、iAnywhere JDBC ドライバを使用することによって得られるアプリケーションのパフォーマンスおよび柔軟性の改善は、iAnywhere JDBC ドライバの使用を検討する価値があります。
pure Java は、どのプラットフォームのどの JVM でもアプリケーションを実行できるという利点があります。iAnywhere JDBC ドライバは外部の ODBC ドライバが必要であり、そのほとんどが特定のプラットフォーム専用であるため、どこでも実行できるわけではありません。
SQL Anywhere の場合、ODBC ドライバ (dbodbc10) は C/C++ で記述されたライブラリ専用です。このドライバは、Windows (NT/XP/CE/Vista)、Solaris、HP-UX、Linux、AIX、および Mac OS X の各種バージョンを含む、非常に広範囲のプラットフォームで使用できます。ODBC クライアントのインストールに必要なファイルは良くできているため、事実上、iAnywhere JDBC ドライバは圧倒的多数の Java アプリケーションが実行しているすべてのプラットフォームで動きます。pure Java にこだわって、パフォーマンスや機能性を犠牲にする価値が本当にあるのでしょうか。

※1 これは英語のニュースグループを示しています。

結論

SQL Anywhere 接続を Java アプリケーションに追加する気になったら、まず最初に iAnywhere JDBC ドライバを検討してください。pure Java または TDS 通信を必要とする少数の状況を除くほとんどの状況で、iAnywhere JDBC ドライバがアプリケーションのスループットを改善し、開発者の生産性を向上させるでしょう。

1 JDBC テクノロジ・ドライバ (Sun Developer Network)
http://java.sun.com/products/jdbc/driverdesc.html
2 Java Database Connectivity (Wikipedia)
http://ja.wikipedia.org/wiki/JDBC