WebOTX での
MySQL 利用方法

2006.7.26
初 版

NEC

前書き

【本書の位置付け】

本書は、MySQL を利用するための WebOTX での設定方法、ならびに MySQL の適用領域について説明したものです。MySQL を用いたシステム構築を検討している方や、今後、検討する可能性のある方を対象としています。
本書が対象とする WebOTX のバージョンは、6.1 以降です。

なお、WebOTX に関する情報は下記のサイトに掲載しておりますので、ご活用ください。

・WebOTX 製品紹介サイト
http://www.sw.nec.co.jp/WebOTX/

目次

1 はじめに

MySQL は、世界で最も人気のあるオープンソースのデータベース管理システムの一つです。性能を重視したデータベース管理システムとして知られ、大規模 Web システムでの活用事例もあります。

本書では MySQL の現在の提供機能の概要と WebOTX で MySQL を利用する方法を紹介します。適用領域についても考察を記載しています。

2 MySQL とその JDBC ドライバ

本章では、2006 年 4 月現在の、MySQL の機能や性能についての調査結果を説明します。

2.1 MySQL の概要

MySQL はスウェーデンの MySQL AB 社によって開発されている、人気の高いオープンソースのデータベース管理システムです。マルチストレージエンジン・アーキテクチャの採用により、用途や目的に合わせてデータの格納エンジンを変更できることが特徴の一つです。

デュアルライセンス方式を採用しており、利用形態に応じて、商用ライセンスと無償ライセンス(GPL)のいずれかを選択することができます。有償契約により正規のサポートを受けることができ、国内にも複数の代理店を持つなど、ビジネスとしての利用も広がっています。

他のデータベース管理システムと比較して、特に性能を重視した設計となっており、アクセス負荷の高い Web システムのバックエンドとして採用されるなどの利用実績も豊富です。反面、標準のストレージエンジンではトランザクション機能や外部キーをサポートしないなど、当初、関係データベースとしての機能に多くの制限がありました。

MySQL 3.23.34 および MySQL 4.0 以降、トランザクションなどをサポートしたストレージエンジンが使用できるようになり、2005 年 10 月にリリースされた MySQL 5.0 では、ストアドプロシージャやトリガー、ビューに対応するなど、機能面での充実が図られるようになりました。今後、高機能化に伴い、エンタープライズ分野での利用範囲が拡大していくものと思われます。

2.2 ストレージエンジン

MySQL は、内部データの格納に用いるストレージエンジンとして、次の型を選択することができます。

- 高速だがトランザクションなどをサポートしない、比較的低機能な MyISAM
- 若干低速ながらトランザクションなどをサポートする、高機能な InnoDB
- MERGE, MEMORY(HEAP), BDB(Berkeley DB), およびその他のストレージエンジン

MyISAM などのトランザクションをサポートしないストレージエンジンでは、自動コミットが行われるため、J2EE トランザクションのトランザクション制御を行うためには利用できません。アプリケーションから直接、java.sql.Connection クラスの commit や rollback メソッドを呼び出す場合も同様です。

コミットやロールバックといったトランザクションの制御を行うためには、トランザクションをサポートするInnoDBやBDBを利用する必要があります。特にInnoDBは、MySQL 5.0 から分散トランザクション機能に対応したというプレスリリースがあり、その実装を調査いたしました。詳しくは 2.6　をご覧ください。

今回は、WebOTX から InnoDB と MyISAM のストレージエンジンを利用して、評価を行いました。

2.3 JDBCドライバの概要

MySQL の JDBC ドライバは MySQL Connector/J という名称で、開発元の MySQL AB 社より提供されています。MySQL Connector/J は、Pure Java の Type4 のドライバであり、MySQL サーバと直接通信を行います。MySQL Connector/J は、JDBC 3.0 に対応しています。

2.4 ソフトウェア条件

MySQL Connector/J は Pure Java で実装されており、バージョン 1.2 以降の J2SE SDK が動作する OS をすべてサポートしています。これに対して、MySQL 本体にはサポートしていない OS があります。

2006 年 4 月現在の、WebOTX 6.31 がサポートする OS および J2SE SDK との対応表を次に示します。一番右の列に示している通り、MySQL は 64bit Windows OS をサポートしていません。WebOTX のその他のバージョンの対応表についてはマニュアルをご参照ください。

	bit	CPU	OS	SDK 1.4.2	SDK 5.0	MySQL サポート対象
Windows	32	x86	Windows 2000 Server Windows Server 2003	○	○	○
		EM64T	Windows Server 2003	○	○	○
	64	Itanium	Windows Server 2003	○	×	×
		EM64T	Windows Server 2003	×	○	×
HP-UX	32	PA-RISC	HP-UX 11.0 HP-UX 11i v1	○	○	○
	64	Itanium	HP-UX 11i v2	○	○	○
Solaris	32	SPARK	Solaris 8 Solaris 9	○	○	○
Linux	32	x86	RHEL 3.0 RHEL 4.0	○	○	○
	64	Itanium	RHEL 4.0	○	×	○

2.5 簡単な性能測定結果

WebOTX の JDBC データソース（コネクションプール機能）から MySQL Connector/J を使用した場合の、MySQL の性能を測定しました。測定環境と手順、および結果は次の通りです。

・ 測定環境

OS	Red Hat Enterprise Linux ES release 3
Application Server	WebOTX 6.31
DBMS	MySQL Server 5.0.19-0
Storage Engine	InnoDB / MyISAM
JDBC Driver	MySQL Connector/J 5.0 beta

・ 測定内容

MySQL が動作するマシンで、InnoDB ストレージエンジンと MyISAM ストレージエンジンを使用して次の処理を繰り返し行い、その実行時間を測定しました。JDBC データソースの defaultAutoCommit プロパティの値はデフォルトの true としたため、自動コミットが行われます。

- JTA 連携なしで、コネクションの取得、SELECT または UPDATE 文の発行、コネクションのクローズを行う

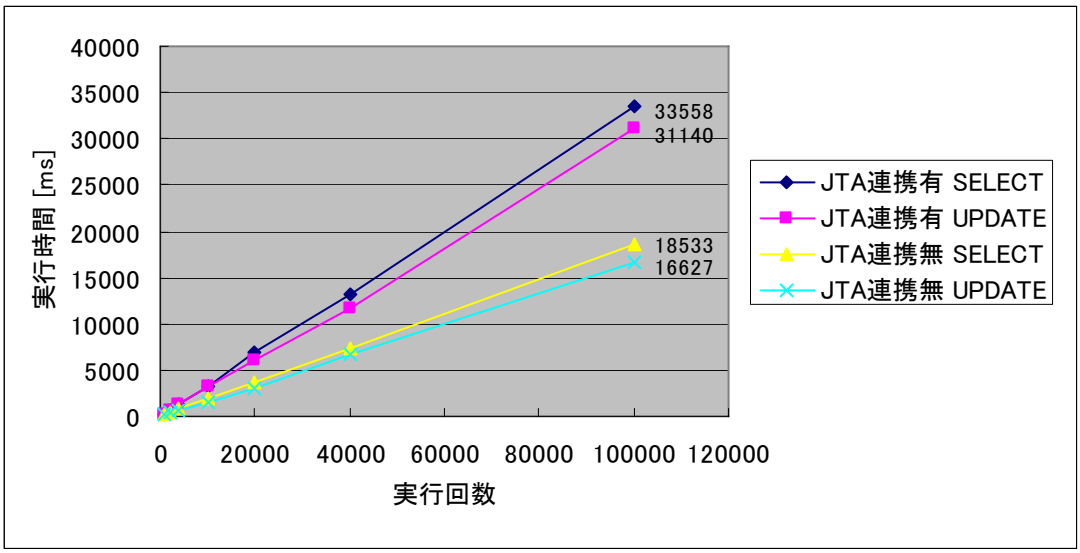
また、InnoDB ストレージエンジンでは、次の処理を繰り返した場合の実行時間も測定しました。

- JTA 連携ありで、トランザクションの開始、コネクションの取得、SELECT または UPDATE 文の発行、コネクションのクローズ、トランザクションのコミットを行う

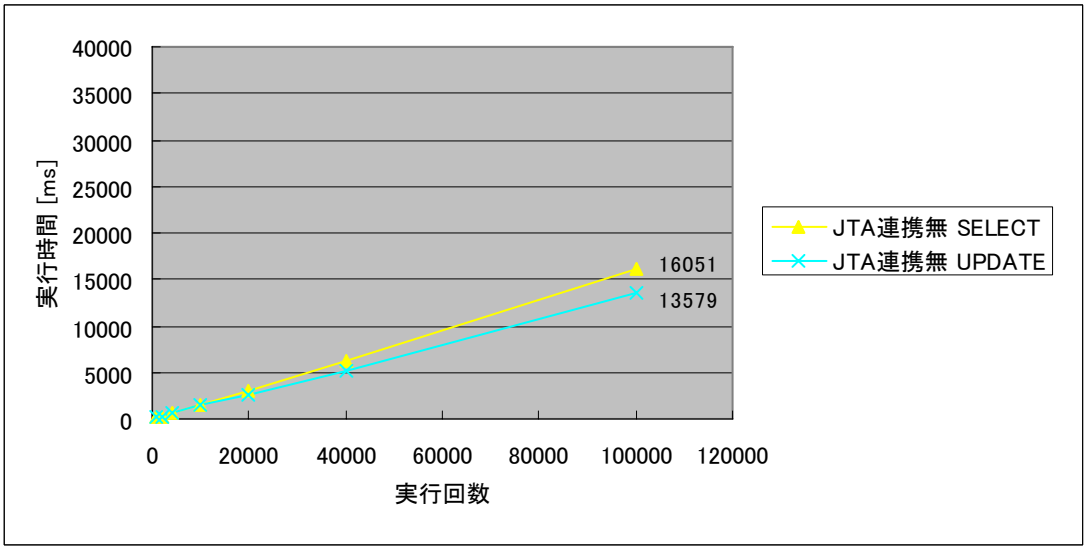
・ 測定結果

測定を 3 回行った結果の平均値は次のようになりました。InnoDB を利用した場合、一番速い、JTA 連携なしの UPDATE のケースでは、1 回当たり約 0.17 ミリ秒になりました。同様に、MyISAM を利用した場合、1 回当たり約 0.14 ミリ秒になりました。他のデータベースと比較しても、MySQL の性能は遜色ないレベルです。

◆ InnoDB ストレージエンジンの測定結果



◆ MyISAM ストレージエンジンの測定結果



2.6 JTA の実装の制限

MySQL Connector/J は、バージョン 5.0 において、JTA(分散トランザクションのための API)に対応しましたが、次の様な制限があります。

(a) 使用できるストレージエンジンは InnoDB だけです。
(b) 1 トランザクションあたり、1 コネクションしか使用できません。
(c) 同一コネクション内で、グローバルトランザクションとローカルトランザクションの切り替えを行うことができません。
(d) データベースサーバの再起動や通信障害が発生した場合、実行中のトランザクションはすべてロールバックされます。そのため、2 フェーズコミットでトランザクションが準備状態になったとしても、コミットの成功が保証されません。

これらの問題に対する、WebOTXでの対応について、3　章で説明します。

3 WebOTX での対応

本章では、MySQL に対する WebOTX の対応方針について説明しています。 また、提供機能について簡単に説明しています。

3.1 汎用のデータソースタイプ ″JDBC″

WebOTX の JDBC データソースでは、JDBC ドライバごとにデータソースタイプとして文字列を定義することで、簡単に JDBC ドライバやロードするクラスを変更することができます。しかし、現在のところ WebOTX の JDBC データソースは MySQL 専用のデータソースタイプを提供していません。そのため、MySQL に接続する場合、データソースタイプには汎用のデータソースタイプ"JDBC"を指定します。

WebOTXで専用のデータソースタイプを提供する目的の一つに、JDBCドライバのJTAの機能に対応することがあります。しかし、2.6　で説明したように、MySQLのJDBCドライバにはJTAの実装に制限があるため、現在のところ専用のデータソースタイプを提供していません。

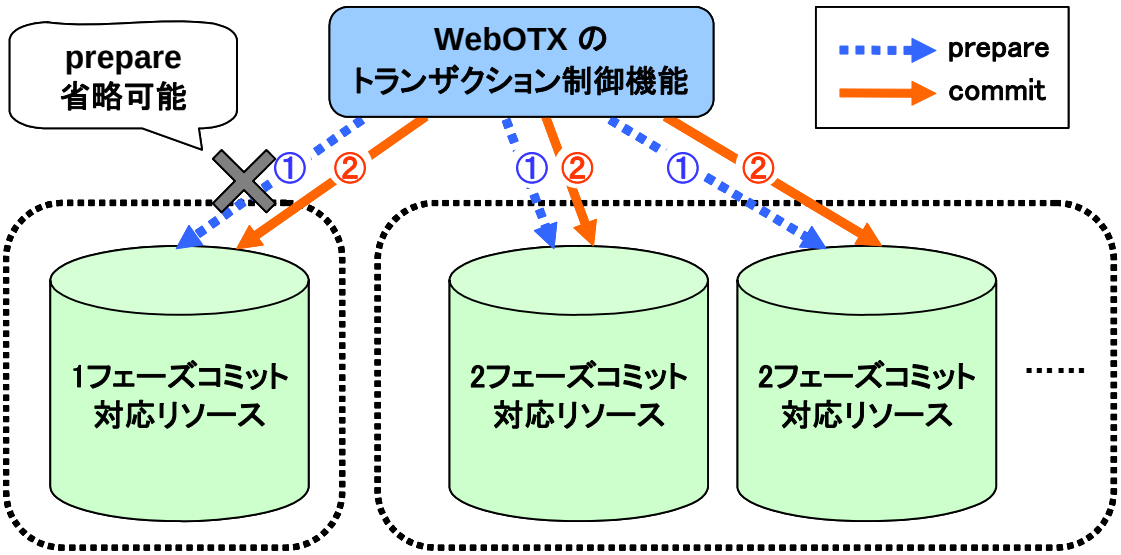
汎用のデータソースタイプ"JDBC"を利用することにより、2.6　で説明した制限のうち、(b)と(c)の制限を解除することができます。また、WebOTXの 1 フェーズコミット対応リソースとして扱われるため、2 フェーズコミットのトランザクションに参加することができます。

3.2 1 フェーズコミット対応リソースの機能

1 フェーズコミット対応リソースとは、トランザクション内で使用されるデータベースなどのリソースのうち、1 フェーズコミットの調整だけをサポートするものを指します。これに対して、2 フェーズコミットによるトランザクションの調整をサポートするものを 2 フェーズコミット対応リソースと呼びます。

1 フェーズコミット対応リソースでは、2 フェーズコミットメントの第 1 段階(プリペア)を実行することができません。このため、標準的なトランザクション機能では、1 フェーズコミット対応リソースを 2 フェーズコミットに参加させることはできません。

ですが、WebOTX のトランザクション制御機能では、1 フェーズコミット対応リソースを、2 フェーズコミットのトランザクションに参加させることができます。



3.3 1 フェーズコミット対応リソースの制限

1 フェーズコミット対応リソースの実装が完全ではないことから、トランザクションの同時更新における一貫性を保証する上で、次のような構成上の制限があります。

- 1 つのグローバルトランザクションに参加できる 1 フェーズコミット対応リソースは 1 つだけです。
- 伝播先トランザクションに 1 フェーズコミット対応リソースが参加している場合、伝播元(上位)のトランザクションに登録されているリソースが存在してはいけません。

また、コミットが失敗した場合、通常ロールバックされた状態になりますが、データベースサーバ側でのコミット処理完了後に通信障害が発生した場合など、稀にコミットされていることがあります。そのため、コミットの内容からトランザクションを復旧する方法を提供しておく必要があります。

これらの詳細については、マニュアルの「運用操作」-「Transaction サービスの運用操作」-「1 フェーズコミット対応リソースの 2 フェーズコミットへの参加」、および「注意制限事項」-「JDBC データソース」をご覧ください。

4 WebOTX における MySQL の設定方法

本章では、WebOTX で MySQL を利用するための方法について説明します。WebOTX で MySQL を利用するためには、次の作業を行ってください。

- JDBC ドライバの準備
- JDBC データソースの設定
- JDBC データソースの接続テスト

それぞれの具体的な手順を次に説明します。

4.1 JDBC ドライバの準備

◆ MySQL Connector/J の入手

MySQL の JDBC ドライバである MySQL Connector/J を入手します。MySQL AB 社のウェブサイトから、mysql-connector-java-5.x.x.zip というファイルをダウンロードしてください。(5.x.x というバージョ

ン部分は、必要に応じて読み替えてください）

MySQL Connector/J の製品 URL は、2006 年 4 月現在、次の通りです。
http://www.mysql.com/products/connector/j/

ダウンロードが完了したら圧縮ファイルを展開してください。mysql-connector-java-5.x.x-beta-bin.jar が JDBC ドライバの Jar ファイルです。

◆ クラスパスの設定

WebOTX Ver.6 以降では、次のディレクトリに JDBC ドライバのファイルを格納することにより、自動的に JDBC ドライバへのクラスパスが追加されます。設定を反映させるためにはドメインの再起動が必要です。

<WebOTX インストールディレクトリ>/domains/<ドメイン名>/lib/ext

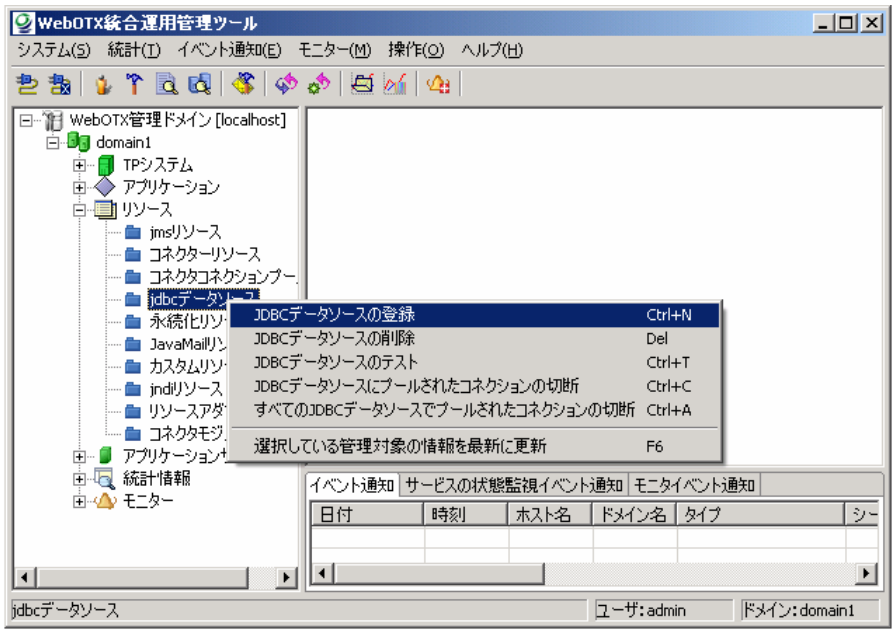
4.2 JDBC データソースの設定

本書では設定方法として、統合運用管理ツールから行う方法と、otxadmin コマンドから行う方法を紹介します。

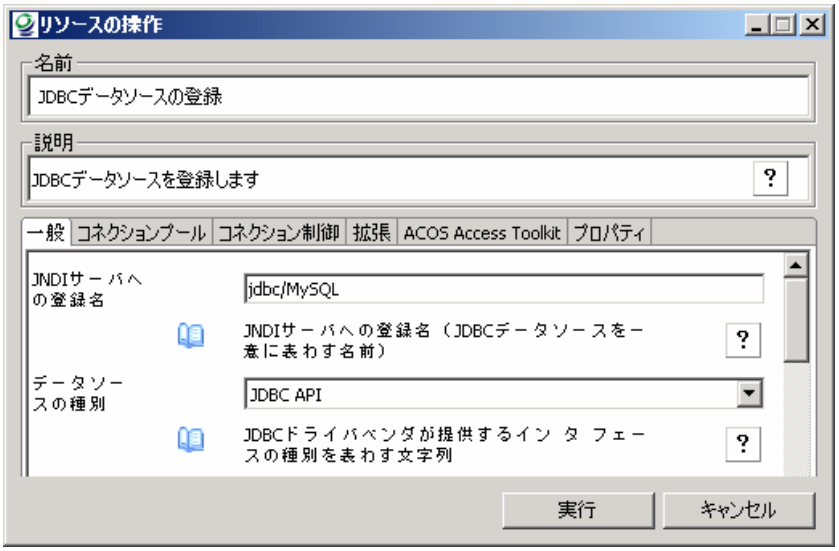
◆ 統合運用管理ツールから設定する方法

① 統合運用管理ツールを起動し、ドメインにログインします。

② 「WebOTX 管理ドメイン」-「(ドメイン名)」-「リソース」-「jdbc データソース」を選択し、右クリックメニューから「JDBC データソースの登録」を選択します。なお、WebOTX のバージョンにより、「リソース」は「resources」、「jdbc データソース」は「jdbc-datasource」の様に、英語で表示される場合があります。

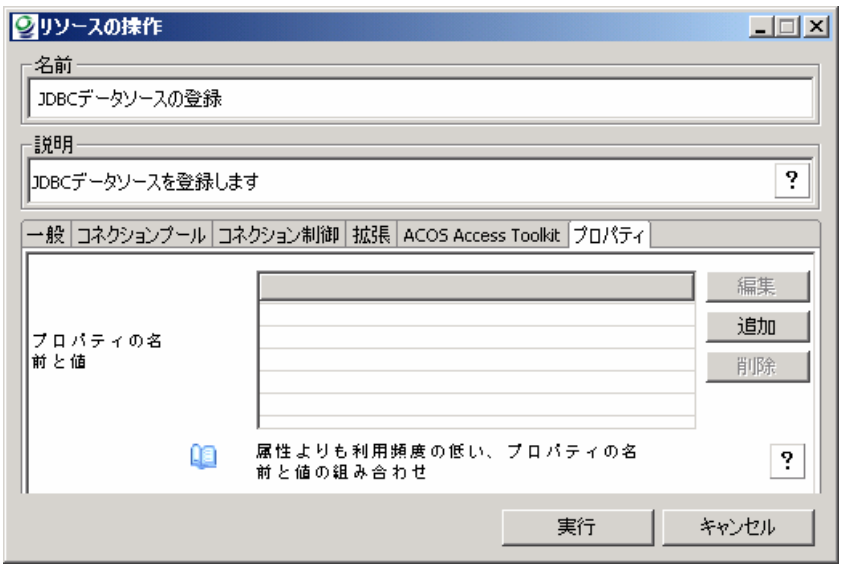


③ 「リソースの操作」ウィンドウの「一般」タブが表示されますので、環境に合わせて変更してください。



項目名	値
JNDI サーバへの登録名	登録する JDBC データソースの名前を設定します 設定例："jdbc/MySQL"
データソースの種別	"JDBC API"
JDBC 仕様のメジャーバージョン	"3"
JTA 連携有無	必要に応じてチェックします
JDBC データソースについての説明	任意の文字列を設定します
JDBC URL またはデータベース名、データソース名	jdbc:mysql://ホスト名:ポート番号/データベース名 設定例："jdbc:mysql://localhost:3306/dbname"
ユーザ名	データベースとの接続時に使用するユーザ名を指定します
パスワード	データベースとの接続時に使用するパスワードを指定します

④　ウィンドウ内の「プロパティ」タブを選択します。



⑤　「追加」ボタンを押し、表示されるダイアログボックスに次のように入力して「OK」ボタンを押します。

```
jdbcDriverName=com.mysql.jdbc.Driver
```

⑥ 以上の操作が完了しましたら、ウィンドウ下部の「実行」ボタンを押します。設定が完了すると、「結果」ウィンドウに「操作の実行に成功しました」と出力されます。

統合運用管理ツールについての詳細は、WebOTX マニュアルの「運用編」-「運用管理ツール」をご覧ください。

◆ otxadmin コマンドから設定する方法

otxadmin コマンドを実行し、プロンプトに次の様に入力します。なお、太字の部分は、実際の環境に合わせて変更してください。

① ログインを行います。

```
otxadmin> login --user admin --password adminadmin --host localhost --port 6212
```

ログインが成功すると、"login successful."と出力されます。

② JDBC データソースの設定を行います。

ここでは例として、登録する名前を"jdbc/MySQL"、JTA 連携有りとしています。また、接続するホスト名は"localhost"、データベース名は"dbname"、接続するユーザ名は"dbuser"、そのパスワードは"dbpass"としています。いずれも環境に合わせて変更してください。

```
otxadmin> create-jdbc-datasource
            --dataSourceType JDBC --jdbcMajorVersion 3 --useJTA=true
            --dataSourceName jdbc:mysql://localhost:3306/dbname
            --jdbcUserName dbuser --jdbcPassword dbpass
            --property jdbcDriverName=com.mysql.jdbc.Driver   jdbc/MySQL
```

※ 実際には 1 行で入力します。

※ WebOTX Ver.6.1 の場合、上記の dataSourceName, jdbcUserName, jdbcPassword 属性を、次の通り、dataSourceName, userName, password プロパティとして設定してください。

(設定例)

```
otxadmin> create-jdbc-datasource
            --dataSourceType JDBC --jdbcMajorVersion 3 --useJTA=true
            --property userName=dbuser:password=dbpass:
                       dataSourceName=¥"jdbc:mysql://localhost:3306/dbname¥":
                       jdbcDriverName=com.mysql.jdbc.Driver
            jdbc/MySQL
```

③ 次のコマンドを実行し,現在の設定内容を確認します。

```
otxadmin> get server.resources.jdbc-datasource.jdbc/MySQL.*
```

これらのコマンドについての詳細は、WebOTX マニュアルの「運用管理コマンドリファレンスマニュアル」-「運用管理エージェント 運用管理コマンド（otxadmin）」をご覧ください。

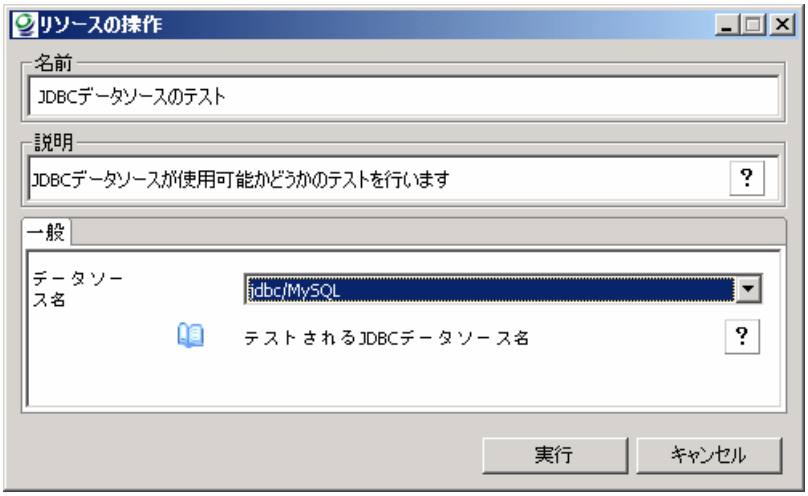
4.3 JDBC データソースの接続テスト

登録したデータソースの接続テストを行う方法として、統合運用管理ツールから行う方法と、otxadmin コマンドから行う方法を紹介します。

◆ 統合運用管理ツールから設定する方法

ドメインにログイン後、「WebOTX 管理ドメイン」-「(ドメイン名)」-「リソース」-「jdbc データソース」

を選択し、右クリックメニューから「JDBC データソースのテスト」を選択します。表示されるダイアログボックスの「データソース名」として JNDI サーバへの登録名を選択し、「実行」ボタンを押します。



接続テストが成功すると、「操作の実行に成功しました」と出力されます。

◆ otxadmin コマンドから設定する方法

ドメインにログイン後、次のコマンドを実行してください。太字の部分は、環境に合わせて変更してください。

```
otxadmin> ping-jdbc-datasource jdbc/MySQL
```

接続テストが成功すると、"Command ping-jdbc-datasource executed successfully."と出力されます。

5 JDBC アプリケーションの記述例

JDBC データソースを使用したアプリケーションのコード例を紹介します。

```
import java.sql.*;
import javax.sql.DataSource;
import javax.naming.*;

public class Sample {

    InitialContext ic;
    DataSource ds;

    public static void main(String[] args) {
        try {
            Sample s = new Sample();
            s.method_init();
            s.method_1();
        } catch (NamingException ex) {
            ex.printStackTrace();
        }
    }

    public void method_init() throws NamingException {
        try {
            // 1. JDBC データソースの取得
            ic = new InitialContext();
            ds = (DataSource)ic.lookup("jdbc/MySQL");
        } catch (NamingException ex) {
            ex.printStackTrace();
```

```
            throw ex;
        }
    }

    public void method_1() {
        Connection conn = null;
        try {
            // 2. JDBC コネクションの取得
            conn = ds.getConnection();
            // SQL 命令の実行
            // ...
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            if (conn != null) {
                try {
                    // 3. JDBC コネクションのクローズ
                    conn.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

1. JDBC データソースの取得

予め、JNDI サーバに登録済みの JDBC データソースを取得します。JDBC データソースの取得は、時間がかかる処理ですので、プロセスで一度だけ行うようにしてください。

2. JDBC コネクションの取得

JDBC データソースから JDBC コネクションを取得します。アプリケーションは、接続先データベース固有の情報を意識する必要がありません。

3. JDBC コネクションのクローズ

取得した JDBC コネクションは、必ずクローズします。

これらについての詳細は、WebOTX マニュアルの「アプリケーション開発ガイド」-「プログラミング・開発（J2EE 編その 2)」-「JDBC アプリケーション」をご覧ください。

6 MySQL の適用領域

本章では、MySQL と、各ストレージエンジンの適用領域について考察した結果を説明します。

6.1 InnoDB ストレージエンジンの適用領域

ストレージエンジンに InnoDB を使用する場合、汎用のデータソースタイプ"JDBC"を指定することで、1 フェーズコミット対応リソースの機能により 2 フェーズコミットを実現することができます。しかし、この機能にはいくつかの制限があるため、高い信頼性を求められるミッションクリティカルなシステムでの利用には適していないといえます。2 フェーズコミットを行う必要がある場合は、Oracle などの、２フェーズコミットをサポートしていて実際に運用実績のある RDBMS を利用するべきです。そのほか、十分な評価や検証ができていないという理由もありますが、現段階では、次のようなシステムで利用すること

が妥当ではないかと考えられます。

・ ミッションクリティカルでない、1 フェーズコミットだけを行うシステム

6.2 MyISAM ストレージエンジンの適用領域

ストレージエンジンに MyISAM を使用する場合、トランザクション制御を行うことができないという制限があります。そのため、J2EE のトランザクション制御を行う場合には利用できません。その代わり、実行性能では InnoDB を上回っており、データ更新の取り消しが必要のない次の様な用途で利用することが考えられます。

・ トランザクションデータやログの格納域
・ キャッシュなどの一時的なデータの管理

6.3 MySQL の適用領域

6.1 、6.2 より、MySQLは比較的単純な、または、高速に処理を行う必要のあるデータを扱う場合に、その利点が生かされるといえます。また、導入コストが低く抑えられるだけでなく、オープンソースソフトウェアでありながらサポートも充実しているなど、業務での利用に適した面があります。反面、分散トランザクション環境下のシステムでの利用に適していないなど、注意が必要な面もあります。更に商用での利用実績において、他のデータベース管理システムに及ばないという課題もあります。

MySQL は高速なデータベース管理システムであり、人気の高いオープンソースソフトウェアです。そこで今回接続検証を行い、その結果をご紹介いたしました。今後、MySQL の利用を検討されることがあれば、本書をご参考にしてください。