

Использование MySQL в разработке Java-приложений

*Sveta Smirnova, Support Engineer
MySQL Bugs Group*

Connector/J: версии

- **Connector/J 5.1** – Type IV pure Java JDBC driver; полная совместимость с MySQL 4.1, 5.0, 5.1, 6.0 включая Falcon
- **Connector/J 5.0** – Connector/J 3.1 + распределенные транзакции (XA)
- **Connector/J 3.1** – работает с MySQL 4.1 и 5.0, кроме распределённых транзакций (XA)
- **Connector/J 3.0** – базовая версия; MySQL 3.x, 4.1, совместима с последующими версиями, но не имеет поддержки server-side prepared statements и не поддерживает функциональность, появившуюся после 4.1

Примеры использования: получение соединения

- `Connection c = DriverManager.getConnection("jdbc:mysql://localhost/test?user=xxx&password=xxx");`
- `Connection c = DriverManager.getConnection("jdbc:mysql:replication://master,slave1,slave2/test?user=xxx&password=xxx");` для `ReplicationConnection`
- `Connection c = DriverManager.getConnection("jdbc:mysql:loadbalance://master,slave1,slave2/test?user=xxx&password=xxx");` для `load-balanced connection`

Примеры использования

- Выполнение запросов

```
Statement s = c.createStatement();
```

```
ResultSet rs = s.executeQuery("SELECT ...");
```

```
int num_rows = s.executeUpdate("UPDATE | DELETE |  
INSERT ...");
```

```
bool select_query = s.execute("SELECT | UPDATE |  
DELETE | INSERT ...");
```

```
ResultSet rs = s.getResultSet();
```

```
int num_rows = s.getUpdateCount();
```

Примеры использования

Получение значения AUTO_INCREMENT столбца

```
CREATE TABLE t1(c1 INT AUTO_INCREMENT PRIMARY
  KEY, c2 VARCHAR(255));
```

- Statement.getGeneratedKeys() – JDBC API 3.0, portable

```
Statement s = c.createStatement(
  java.sql.ResultSet.TYPE_FORWARD_ONLY,
  java.sql.ResultSet.CONCUR_UPDATABLE);
s.executeUpdate(
  "INSERT INTO t1(c2) VALUES("hello, world"),
  Statement.RETURN_GENERATED_KEYS);
ResultSet rs = s.getGeneratedKeys();
rs.next();    // Error if false
int autoGenId = rs.getInt(1);
rs.close();
```

Примеры использования

Получение значения AUTO_INCREMENT столбца

```
CREATE TABLE t1(c1 INT AUTO_INCREMENT PRIMARY  
KEY, c2 VARCHAR(255));
```

- **SELECT LAST_INSERT_ID()** – MySQL specific
Statement s = c.createStatement();
s.executeUpdate(...);
ResultSet rs = s.executeQuery("SELECT
LAST_INSERT_ID()");
rs.next(); // Error if false
int autoGenId = rs.getInt(1);
rs.close();

Примеры использования

Получение значения AUTO_INCREMENT столбца

```
CREATE TABLE t1(c1 INT AUTO_INCREMENT PRIMARY  
KEY, c2 VARCHAR(255));
```

- Updatable result set – insertRow()

```
Statement s = c.createStatement(  
    java.sql.ResultSet.TYPE_FORWARD_ONLY,  
    java.sql.ResultSet.CONCUR_UPDATABLE);  
ResultSet rs = s.executeQuery("SELECT c1, c2 FROM  
t1");  
rs.moveToInsertRow();  
rs.updateInt("c2", 1234);  
rs.insertRow();  
rs.last();  
int autoGenId = rs.getInt("c1");  
rs.close();
```

Примеры использования

хранимые процедуры

- Использование хранимых процедур


```
CREATE PROCEDURE p1(v1 INT, INOUT v2 INT, OUT v3INT)
BEGIN ... END
```
- Подготовка callable statement


```
CallableStatement cs = c.prepareCall("{call p1(?, ?, ?)}");
```
- Регистрация INOUT / OUT параметров


```
cs.registerOutParameter(2, Types.INTEGER); // by Index
cs.registerOutParameter("v3", Types.INTEGER); // by Name
```
- Задание значений IN / INOUT параметров


```
cs.setInt(1, 1234); // by Index
cs.setInt("v2", 4321); // by Name
```

Примеры использования

хранимые процедуры

- Вызов процедуры


```
boolean has_results = cs.execute();
```
- Получение значений INOUT / OUT параметров


```
int v2 = cs.getInt(2);           // by Index
int v3 = cs.getInt("v3");       // by Name
```
- Получение result set


```
while (has_results)
{
    ResultSet rs = cs.getResultSet();
    ...
    has_results = cs.getMoreResults();
}
```

Примеры использования

- Пример программы
 - выполнение запросов
 - получение значения affected rows
 - получение значения
 - вызов процедуры
 - опция noDatetimeStringSync и расхождение в формате времени, используемом в MySQL и JDBC

Примеры использования

```
import testsuite.BaseTestCase;
import java.sql.ResultSet;
import java.sql.CallableStatement;
import java.sql.Types;
import java.sql.SQLException;
public class odessa1 extends BaseTestCase {
public odessa1(String name) {
super(name);
}
public static void main(String[] args) {
junit.textui.TestRunner.run(odessa1.class);
}
```

Примеры использования

```
public void testOdessa1() throws Exception {
    try {
        this.stmt.execute("drop table if exists odessa1");
        this.stmt.execute("create table odessa1(f1 time default
            NULL)");
        assertEquals(24, this.stmt.executeUpdate("insert into
            odessa1 values('03:08:22'), ('03:08:25'),
            ('03:00:18'), ('03:02:31'), ('01:39:07'), ('03:05:47'),
            ('03:10:29'), ('01:12:22'), ('02:20:23'), ('03:05:02'),
            ('02:39:00'), ('03:08:36'), ('01:13:19'), ('02:13:38'),
            ('02:10:11'), ('02:57:36'), ('03:06:06'), ('03:11:02'),
            ('03:12:29'), ('01:07:32'), ('03:17:18'), ('03:04:52'),
            ('03:20:12'), ('03:14:33')"));
    }
}
```

Примеры использования

```

if (this.stmt.execute("select
    SEC_TO_TIME(SUM(TIME_TO_SEC(f1))) as exercise
    from odessa1")) {
ResultSet rs = this.stmt.getResultSet();
try {
while(rs.next()) { assertEquals("64:49:10",
    rs.getString(1)); }
} catch (SQLException sqle) {
System.err.println("Option noDatetimeStringSync set to
    false");
}
rs.close();
}

```

Примеры использования

```
this.stmt.execute("drop procedure if exists p_odessa1");  
this.stmt.execute("create procedure p_odessa1(out  
    exercise time) begin select  
    SEC_TO_TIME(SUM(TIME_TO_SEC(f1))) into exercise  
    from odessa1; end");  
CallableStatement cs = this.conn.prepareCall("call  
    p_odessa1(?)");  
cs.registerOutParameter(1, Types.TIME);  
cs.execute();
```

Примеры использования

```
try {  
    assertEquals("64:49:10", cs.getTime(1).toString());  
} catch (SQLException sqle) {  
    assertEquals("64:49:10", cs.getString(1));  
}  
cs.close();  
} finally {  
    closeMemberJDBCResources();  
}  
}  
}
```

Примеры использования

- Пример программы
 - выполнение запросов
 - использование Prepared Statements
 - получение значения AUTO_INCREMENT
 - получение значения
 - опция zeroDateTimeBehavior и расхождение в формате времени, используемом в MySQL и JDBC

Примеры использования

```
import testsuite.BaseTestCase;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Date;

public class odessa2 extends BaseTestCase {

    public odessa2(String name) {
        super(name);
    }

    public static void main(String[] args) {
        junit.textui.TestRunner.run(odessa2.class);
    }
}
```

Примеры использования

```

public void testOdessa2() throws Exception {
try {
    this.stmt.execute("drop table if exists odessa2");
    this.stmt.execute("create table odessa2(id int not null
auto_increment primary key, f1 date not null default '000-00-00')");
    PreparedStatement psmt = this.conn.prepareStatement("insert
into odessa2 (f1) values (?)");
    psmt.setDate(1, Date.valueOf("2008-09-30"));
    psmt.addBatch();
    psmt.setInt(1, 0);
    psmt.addBatch();
    psmt.executeBatch();
    ResultSet rs = psmt.getGeneratedKeys();
    if (rs.next()) { assertEquals(2,rs.getInt(1)); }
    rs.close();
    psmt.close();
}
}

```

Примеры использования

```

if (this.stmt.execute("select id, f1 from odessa2")) {
    rs = this.stmt.getResultSet();
    try {
        while(rs.next()) {
            assertTrue(rs.getDate(2).toString().equals("2008-09-30") || rs.getDate(2).toString().equals("0001-01-01"));
        }
    } catch (SQLException sqle) {
        System.err.println("Option zeroDateTimeBehavior set to exception");
    }
    rs.close();
}
} finally { closeMemberJDBCResources(); }
}
}
}

```

Приемы эффективной работы

- Использовать последний драйвер
 - Собственно улучшается производительность драйвера
 - Драйвер лучше использует возможности сервера
- Использовать `ResultSet.getXXX(index)` вместо `getXXX(name)`
- Не запрашивать `Updatable (Scrollable) ResultSet` когда это не нужно
- Освобождать ресурсы ASAP (на практике самое критичное)
- Использовать `Statement.getGeneratedKeys()` для получения значений `AUTO_INCREMENT`-столбцов

Приемы эффективной работы

- Планирование архитектуры приложения
- Тюнинг запросов
- Тюнинг MySQL сервера
- Тюнинг Connector/J при помощи различных настроек

Приемы эффективной работы: отладка

- Штатные средства Connector/J
- Логи сервера
slow query log, error log, general query log
(Слишком много информации, перегружает сервер)
- Online logging
Начиная с версии 5.1
- MySQL Proxy
- Enterprise Monitor

Отладка

- `logger`
Имя класса, который реализует `com.mysql.jdbc.log.Log`.
По умолчанию: дамп в `STDERR`.
`Pluggable`, можно переопределять
- `dumpMetadataOnColumnNotFound` [false]
Позволяет вывести в лог информацию о столбцах
когда `ResultSet.findColumn()` возвращает ошибку.
- `dumpQueriesOnException` [false]
Позволяет получить запрос, который вызывал
исключение,
в сообщении `SQLException`.

Instrumentation

медленные запросы

- Выводит в лог медленные запросы
- Выдает агрегированные показатели производительности
- Выдает советы по лучшему использованию (usage advisor)

Instrumentation

медленные запросы

- `slowQueryThresholdMillis(Nanos)` [20000]
- `autoSlowLog` [true]
 Драйвер поддерживает статистику и дампит запросы
 превышающие 99%
- `explainSlowQueries` [false]
- `logSlowQueries` [false]

Wed Apr 14 05:38:02 PDT 2004 WARN:
 at testsuite.simple.ConnectionTest.testSavepoint
 (ConnectionTest.java:376) Slow query (exceeded
 50 ms):
 DROP TABLE IF EXISTS testSavepoints

Instrumentation

Performance Monitor

- gatherPerfMetrics [false]
- reportMetricsIntervalMillis [30000]
- profileSQL [false]

```
Wed Apr 14 05:26:02 PDT 2004 INFO: at
    testsuite.BaseTestCase.tearDown(BaseTestCase.java:186)
    ** Performance Metrics Report **
```

```
Longest reported query: 200 ms
Shortest reported query: 1 ms
Average query execution time: 38.333333333333336 ms
Number of queries executed: 12
Number of queries prepared: 0
Number of prepared statement executions: 0
```

Histogram:

```
Queries taking between 1 and 13: 6
Queries taking between 13 and 25: 1
....
Queries taking between 205 and 217: 1
```

Instrumentation

Usage Advisor

- Печатает в лог “советы по использованию”
 - забытые объекты
 - слишком много строк в result set
 - неиспользуемые столбцы в result set
 - запрос значений столбца с использованием неправильного типа данных
 - неполный обход возвращенного result set
- Properties
 - useUsageAdvisor [false]
 - resultSetSizeThreshold [100]

Instrumentation

Usage Advisor

Wed Apr 14 05:22:28 PDT 2004 WARN: Profiler Event:
[WARN] at
 testsuite.simple.MetadataTest.testForeignKeys
(MetadataTest.java:88)
duration: 0 ms, connection-id: 0, statement-id: 112,
 resultset-id: 138,
message: ResultSet implicitly closed by driver.

Online Logging (MySQL 5.1)

- Начиная в версии 5.1.12 возможно включить/выключить `general` и `slow query logs` без перезагрузки сервера
- Вы можете также выбрать куда писать логи: в таблицу или в файл

Online Logging (MySQL 5.1)

- Удобно для performance tuning:
- Включить
- `SET GLOBAL general_log = 'ON';`
- Найти проблемные запросы, что-то с ними сделать
- Выключить
- `SET GLOBAL general_log = 'OFF';`
- Таким образом overhead использования `general_log` сводится к минимуму

Log destination (MySQL 5.1)

- `--log-output=TABLE`
- `--log-output=FILE`
- `--log-output=NONE`
- `--log-output=TABLE,FILE` (for both)
- `log_output` – глобальная переменная: вы можете изменить расположение `general` и `slow query logs` во время работы сервера

Online Logging (MySQL 5.1)

- <http://dev.mysql.com/doc/refman/5.1/en/query-log.html>
- <http://dev.mysql.com/doc/refman/5.1/en/slow-query-log.html>
- <http://dev.mysql.com/doc/refman/5.1/en/log-tables.html>

Тюнинг Connector/J

- Callable statements cache
 - cacheCallableStmts [false]
 - callableStmtCacheSize [100]
- Prepared statements
 - cachePrepStmts [false]
 - prepStmtCacheSize [25]
 - prepStmtCacheSqlLimit [256]
 - Server-side prepared statements – всегда проверяйте результат: большинство проблем возникает при их использовании

Тюнинг Connector/J

- Meta-data cache
 - `cacheResultSetMetadata` [false]
 - `metadataCacheSize` [50]
- Кроме того:
 - load balancer
 - query timeouts
 - memory / speed trade-offs

Приемы эффективной работы: hacking Connector/J

- connectionLifecycleInterceptors
 - Connection c =
 DriverManager.getConnection("jdbc:mysql://localhost/test?user=xxx&password=xxx&connectionLifecycleInterceptors=CustomLifecycleInterceptor1,CLifecycleInterceptor");
 - Можно указать более одного

Приемы эффективной работы: hacking Connector/J

- Пример кода: Implementation of ConnectionLifecycleInterceptor
 - Ничего полезного не делает
 - Печатает в STDERR сообщение при загрузке и при смене каталога

Приемы эффективной работы: hacking Connector/J

```
import com.mysql.jdbc.ConnectionLifecycleInterceptor;  
import com.mysql.jdbc.Connection;  
import java.sql.SQLException;  
import java.sql.Savepoint;  
import java.util.Properties;  
public class CI1 implements
```

```
ConnectionLifecycleInterceptor
```

```
{  
    public void close() throws SQLException {}  
    public boolean commit() throws SQLException  
        { return true; }  
}
```

Приемы эффективной работы: hacking Connector/J

```
public boolean rollback() throws SQLException
{ return true; }
public boolean rollback(Savepoint s) throws SQLException
{ return true; }
public boolean setAutoCommit(boolean flag) throws
    SQLException
{ return true; }
public boolean setCatalog(String catalog) throws SQLException
{
System.err.println("Using catalog: " + catalog);
return false;
}
```

Приемы эффективной работы: hacking Connector/J

```

public boolean transactionBegun() throws SQLException
{ return true; }
public boolean transactionCompleted() throws SQLException
{ return true; }
public void destroy() {
System.err.println("ConnectionLifecycleInterceptor CI1
    LoadedDestroyed");
}
public void init(Connection conn, Properties props) {
System.err.println("ConnectionLifecycleInterceptor CI1
    Loaded");
} }

```

Приемы эффективной работы: hacking Connector/J

Пример из практики работы Enterprise Team

- <http://www.mysql.com/products/enterprise/>
- Почему не использовать Enterprise Monitor для мониторинга Enterprise Monitor?
- Большинство запросов, «съедающих» ресурсы – SET AUTOCOMMIT=true; - 10 % от общего execution time
- 2-ое по величине execution time в приложении
- Для приложения, использующего только транзакции, выглядит странно

Приемы эффективной работы: hacking Connector/J

Пример из практики работы Enterprise Team

- Почему?
- DBCP содержит следующий код:

```
public void passivateObject(Object obj) throws Exception
{
    if(obj instanceof Connection) {
        Connection conn = (Connection)obj;
        if(!conn.getAutoCommit() && !
conn.isReadOnly()) {
            conn.rollback();
        }
    }
}
```

Приемы эффективной работы: hacking Connector/J

Пример из практики работы Enterprise Team

- Почему?
- DBCP содержит следующий код:

```
conn.clearWarnings();
conn.setAutoCommit(true);
}
if(obj instanceof DelegatingConnection) {
    ((DelegatingConnection)obj).passivate();
}
}
```

Приемы эффективной работы: hacking Connector/J

Пример из практики работы Enterprise Team

- Решение проблемы: ConnectionLifecycleInterceptor

```
public boolean setAutoCommit(boolean flag) throws
    SQLException {
    if (!flag) {
        return true;
    }

    return false;
}
```

Приемы эффективной работы: hacking Connector/J

Пример из практики работы Enterprise Team

- Исправлено при помощи 4 строк кода и конфигурационного параметра

Ресурсы

<http://developers.sun.com/learning/javaoneonline/j1sessn.jsp?sessn=TS-7813&yr=2008&track=javaee>

<http://belarusjug.org/images/stories/docs/Minsk-2008.ppt>

<http://blog.spinn3r.com/2008/04/slides-from-spi.html>

<http://dev.mysql.com/doc/refman/5.1/en/connector-j.html>

http://conferences.oreilynet.com/presentations/mysql07/matthews_JDBC.pdf

http://conferences.oreilynet.com/presentations/mysql07/matthews_Scale-out.pdf

<http://svn.mysql.com/svnpublic/connector-j/trunk/jdbc-4-0-examples/src/examples/cex2007/>

Thank You!

Sveta Smirnova
Support Engineer
sveta@mysql.com