Лекция 28. Работа с базами данных II.

- Выполнение операторов SQL
- Управление соединениями, командами и результирующими наборами
- Анализ SQL-исключений
- Чтение и запись больших объектов
- Транзакции
- Точки возврата
- Пакетные обновления

Выполнение операторов SQL

- Работа с базой данных происходит через объект типа Connection, который можно получить вызвав метод Connection connection = DriverManager.getConnection();
- Для выполнения оператора SQL нужно создать объект типа Statement: Statement st = connection.createStatement();
- В классе Statement определены два метода для выполнения операторов SQL:
  - executeUpdate(command) — возвращает количество строк, которые были изменены в результате выполнения команды; Используется с операторами INSERT, UPDATE, DELETE, CREATE TABLE, DROP TABLE и др. команды изменения данных;
  - executeQuery(query) — используется с оператором SELECT и возвращает объект типа ResultSet — результат запроса.

Интерфейс ResultSet

- Объект типа ResultSet можно использовать для просмотра результатов:
  ResultSet rs = stat.executeQuery(«SELECT * FROM Students»);
  while (rs.next()) { обработка строки результата запроса }
- С помощью соответствующих методов можно получить содержимое каждого столбца:
  String name = rs.getString(1); // нумерация столбцов нач-ся с 1
  Double rating = rs.getDouble(«rating»);
- Метод close() немедленно освобождает используемые ресурсы JDBC, аналогично Statement.close();
- Узнать закрыт ли набор можно вызвав метод isClosed();

Управление соединениями, командами и результирующими наборами

- Каждый объект типа Connection может создать один или несколько объектов типа Statement;
- Один и тот же объект типа Statement можно использовать для выполнения нескольких несвязанных между собой операторов SQL;
- Объект типа Statement не может иметь более одного открытого объекта типа ResultSet одновременно;
- Максимальное количество объектов типа Statement можно узнать вызвав метод DatabaseMetaData.getMaxStatements();
- Закончив работу с объектом типа Connection, Statement или ResultSet нужно вызвать метод close(), чтобы не ждать пока сборщик мусора доберется до массивных структур данных, создаваемых JDBC.

Пример

```
Connection conn = getConnection();
try
{
   Statement stat = conn.createStatement();
   ResultSet result =
         stat.executeQuery("SELECT * FROM Greetings");
   if (result.next())
      System.out.println(result.getString(1));
   result.close();
}
finally
{
   conn.close();
}
```

Анализ SQL-исключений

- Каждое SQL-исключение в дополнение к цепочке «cause» интерфейса Throwable имеет цепочку объектов типа SQLException, которые извлекаются методом getNextException;
- Начиная с Java 6 класс SQLException реализует интерфейс Iterable<Throwable>:
  - Метод iterator() возвращает объект типа Iterator<Throwable>, который осуществляет перебор в обеих цепочках: «cause» для каждого исключения цепочки SQLException;
- Кроме исключений драйвер СУБД может поддерживать предупреждения, которые могут возникнуть:
  - При соединении;
  - Выполнении команды;
  - Работы с результирующими наборами ResultSet;
- Предупреждения поддерживаются классом SQLWarning подклассом SQLException.

Примеры

- Проход всех исключений цепочек SQLException и «cause»:

```
} catch (SQLException se) {
  for(Throwable t : se){
     System.err.println(t.getMessage());
  }
}
```

- Проход предупреждений:

```
SQLWarning sw = stat.getWarnings();
while(sw!=null){
 System.err.println(sw.getLocalizedMessage());
 sw = sw.getNextWarning();
}
```

Чтение и запись больших объектов

- СУБД обычно поддерживает хранение *больших объектов* (LOB); Обычно это *изображения, звук, видео и пр.*
- В JDBC поддерживает две разновидности LOB:
  - Binary Large Object (BLOB);
  - Character Large Object (CLOB);

Чтение и запись больших объектов (продолжение)

- Чтобы прочитать LOB, нужно:
  - Выполнить команду SQL;
  - Вызвать метод getBlob() или getClob() в ResultSet;
  - Методы возвращают объекты классов Blob или Clob;
  - Для получения двоичных данных из Blob вызываем метод Blob.getBytes() или Blob.getInputStream();

Пример чтения BLOB

```
PreparedStatement pstat =
   conn.prepareStatement(
      "SELECT cd_cover FROM AudioLib WHERE Title=?");
pstat.setString(1, "Kalinka");
result = pstat.executeQuery();
if(result.next()){
   Blob blob = result.getBlob(1);
   Image img = ImageIO.read(blob.getBinaryStream());
 }
```

Предварительно подготовленные команды

- В предыдущем примере был создан запрос типа PreparedStatement, содержащий символ «?»;
- Такие запросы можно использовать многократно, подставляя вместо «?» различные значения;
- В предыдущем примере подстановка в запрос конкретного значения осуществлена так:

```
pstat.setString(1, "Kalinka");
```

- Первый аргумент указывает какой по счету параметр мы заполняем конкретным значением.

Пример записи BLOB

```
BufferedImage img = ImageIO.read(
      new URL("http://ya.ru/logo.png"));
Blob blob = conn.createBlob();
int offs = 0;
OutputStream out = blob.setBinaryStream(offs);
ImageIO.write(img, "PNG", out);
PreparedStatement pstat =
    conn.prepareStatement(
      "INSERT INTO AudioLib VALUES (?, ?)");
pstat.setString(1, "Kalinka");
pstat.setBlob(2, blob);
pstat.executeUpdate();
```

Чтобы получить все множественные наборы

1. Используйте метод execute для оператора SQL;
2. Получите первый результат или обновите подсчет;
3. Повторите вызов метода getMoreResults, для перехода к следующему результирующему набору.
4. Этот вызов автоматически закрывает предыдущий ResultSet;
5. Завершите процедуру, когда закончатся результирующие наборы или обновленные подсчеты;
6. Методы execute и getMoreResults возвращают true, если следующим элементом в соединении является ResultSet;
7. Метод getUpdateCount возвращает -1, если следующим элементом в соединении не является обновление подсчета.

Пример получения множественных результатов

```
Connection conn = getConnection();
Statement stmt = conn.createStatement();
String command = "SELECT ...";
boolean done = false;
boolean isResult = stmt.execute(command);
while(!done){
   if(isResult){
      ResultSet result = stmt.getResultSet();
      //do something
   } else {
      int updateCount = stmt.getUpdateCount();
      if(updateCount>=0){
         //do something
      } else {
         done = true;
      }
      isResult = stmt.getMoreResults();
   }
}
```

Транзакции

- В Java поддерживаются транзакции:
  - Отключите автоматическую фиксацию: conn.setAutoCommit(false);
  - Выполните необходимые действия с базой данных;
  - После успешного выполнения всех действий зафиксируйте транзакцию: conn.commit();
  - Если на каком-то этапе произошла ошибка (SQLException), все изменения можно отменить: conn.rollback();

Точки сохранения

- Процесс отката можно контролировать при помощи точек сохранения.
- Пример:

```
Connection conn =...//Открываем соединение.
PreparedStament ps = con.prepareStatement("Select...");
conn.setAutoCommit(false);
 Statement stmt = conn.createStatement();
 ...
 // Усианавливаем точку возврата
 Savepoint svpt = conn.setSavepoint("NewSavepoint");
 ...
 if(some condition) {
   conn.rollback(svpt);
 }
 conn.commit();
```

Пакетные обновления

- Несколько операторов доступа к базе данных можно выполнить в одном пакете;
- Это позволяет повысить производительность работы с СУБД;
- Пакетные обновления поддерживаются для операторов INSERT, UDATE, DELETE, CREATE TABLE, DROP TABLE;

Пример использования пакетных обновлений

```
conn.setAutoCommit(false);
  Statement stmt = conn.createStatement();
  stmt.addBatch( sql 1st insert stmt1 );
  stmt.addBatch( sql 2nd insert stmt1 );
  .....
  stmt.addBatch( sql nth insert stmt  );

  int rowsAfted = stmt.executeBatch( );
```

Верно ли что...

- Для работы с СУБД в Java используются объекты типов: *Connection, Statement, ResultSet*?
- Statement.ExecuteUpdate возвращает объект типа ResultSet?
- ResultSet — это интерфейс?
- Один и тот же объект Statement можно использоват ьдля выполнения нескольких SQL запросов?
- Connection, Statement, ResultSet необходимо закрывать перед повторным использованием?
- CLOB — это реализация BLOB в Java?
- По умолчанию автоматическая фиксация транзакций отключена?
- Пакетные обновления — это выполнение нескольких SQL запросов за одно обращение к СУБД?