Урок 2

Базы данных

Реляционные базы данных, язык запросов SQL. Операторы SELECT, INSERT, UPDATE, DELETE. Подключение к базе через JDBC, отправка запросов и обработка результатов

Оглавление

Базы данных

Для работы с базами данных, нужно установить систему управления базой данных (СУБД). Примеры таких систем - MySQL, Oracle, MS SQL, SQLite. Рассматривать работу с базами данных будем на примере SQLite, она не требует установки и хранит все данные в одном файле.

Особенности SQLite**:**
• хранит всю базу в одном файле;
• не требует установки;
• не поддерживает тип данных Дата;

Поддерживаемые типы данных:
• NULL - NULL значение;
• INTEGER - целое знаковое;
• REAL - с плавающей точкой;
• TEXT - текст, строка (UTF-8);
• BLOB - бинарные данные.

Язык запросов SQL

Аббревиатура CRUD(Create/Read/Update/Delete) содержит в себе все операции, которые можно производить над данными в базе. Эти операции выполняются с помощью языка запросов SQL. Все команды языка регистронезависимы и могут иметь между собой любое количество пробелов и переносов строк.

CREATE

```
CREATE TABLE [имя таблицы] (
[имя колонки] [тип данных],
[имя колонки] [тип данных],
... );
```

Пример запроса:

```
CREATE TABLE IF NOT EXISTS Students
(
  StudID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
  Name TEXT NOT NULL,
  GroupName TEXT NOT NULL,
  Score INTEGER NOT NULL
);
```

Здесь создается таблица Students с полями ID, Name, GroupName, Score.

NOT NULL - означает, что поле всегда должно быть проинициализировано. СУБД следит за этим и в случае, если поле равно NULL выкидывает ошибку.
PRIMARY KEY - означает, что поле имеет уникальное значение в этой таблице (В примере это поле ID - мы хотим, чтобы у каждой записи был уникальный номер).
AUTOINCREMENT - означает, что при каждом добавлении записи в таблицу, ей автоматически будет присвоен ID на единицу больше предыдущего.

READ

Операция чтения данных из таблицы называется SELECT.

SELECT [список полей] FROM [имя таблицы] WHERE [условие];
Примеры запросов:
SELECT * FROM Students;
SELECT * FROM Students WHERE ID > 3
SELECT GroupName FROM Students WHERE ID = 2

Символ звездочка означает, что мы хотим получить все поля таблицы. Иначе мы можем через запятую перечислить необходимые поля. Выражение WHERE необязательно, но помогает извлекать только интересные для нас данные.

UPDATE

Операция изменения уже присутствующих в таблице данных либо добавления новых.

Добавление новых данных:
INSERT INTO [имя таблицы] ([список полей через запятую]) VALUES ([список значений через запятую]);

Изменение:
UPDATE [имя таблицы] SET [имя колонки]=[новое значение], [имя колонки]=[новое значение],.. WHERE [условие];

Примеры запросов:
INSERT INTO Students (Name, GroupName, Score) VALUES ("Bob", "Tbz11", 80);
UPDATE Students SET Score = 90 WHERE Name = "Bob";

DELETE

Удаление данных из таблицы.
DELETE FROM [имя таблицы] WHERE [условие];

Пример запроса:
DELETE FROM ACCOUNTS WHERE ID='0';

JDBC

Каждая СУБД разрабатывается отдельной компанией. Чтобы можно было работать с базой данных, производитель разрабатывает специальный драйвер – JDBC, который позволяет взаимодействовать с ней: устанавливать соединение, посылать запросы и изменять данные, обрабатывать результаты запросов.
Все основные сущности в JDBC API являются интерфейсами: Connection, Statement, PreparedStatement, CallableStatement, ResultSet, Driver, DatabaseMetaData. JDBC драйвер конкретной базы данных как раз и предоставляет реализации этих интерфейсов. DriverManager - синглтон, который содержит информацию о всех зарегистрированных драйверах. Метод getConnection на основании параметра URL находит java.sql.Driver соответствующей базы данных и вызывает у него метод connect.

Установка соединения

Драйвер jdbc обычно можно скачать с сайта производителя СУБД. Он распространяется в виде .jar – библиотеки, которую неободимо подключить к проекту. Прежде чем использовать драйвер, его нужно зарегистрировать - имя драйвера можно также найти на сайте разработчиков.

```
// Для SQLite регистрация выглядит следующим образом
Class.forName("org.sqlite.JDBC");
// Для H2 Database - org.h2.Driver
// Для MySQL - com.mysql.jdbc.Driver
```

Если посмотреть исходный код реализации любого драйвера он будет содержать статический блок инициализации такого вида:

```
static {
    try {
        java.sql.DriverManager.registerDriver(new Driver());
    } catch (SQLException e) {
        throw new RuntimeException("Can't register driver!");
    }
}
```

Вызов Class.forName() загружает класс и этим гарантирует выполнение статического блока инициализации, а значит и регистрацию драйвера в DriverManager. Чтобы указать, как найти базу данных используется URL - специальная строка формата [protocol]:[subprotocol]:[name]
protocol: jdbc
subprotocol: sqlite
name: **test**.db

```
Connection conn;
conn = DriverManager.getConnection("jdbc:sqlite:mydatabase.db");
// … Некоторые действия с БД …
conn.close();
```

Объект Connection предоставляет доступ к базе данных. Опционально в объект Connection можно передать имя пользователя и пароль(если они установлены). После окончания работы с базой, соединение необходимо закрыть методом close().

Запросы в базу

После того, как соединение с базой установлено, можно отправлять запросы. Для этого используется объект Statement, который умеет хранить SQL команды. В базу можно отправить запрос на получение данных, либо на изменение. В первом случае результатом будет объект ResultSet, который хранит результат. Во втором случае - количество строк таблицы, которые были изменены.

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM users");
Statement updateStmt = conn.createStatement();
int result = stmt.executeUpdate("INSERT INTO Students (Name, GroupName, Score) VALUES ("Bob", "Tbz11", 80);");
```

Подготовленный запрос и пакетное выполнение запросов

При необходимости выполнения множества похожих запросов разумным решением будет использование PreparedStatement, который представляет собой скомпилированную версию SQL-выражения, выполнение которого будет быстрее и эффективнее.
В запросах можно использовать параметры, то есть изменять его динамически в зависимости от входных данных. Параметр заменяется символом ?. Каждому параметру в запросе присваивается порядковый номер - индекс. Индексы начинаются с 1. У объекта PreparedStatement есть методы, которые позволяют установить параметры, вам нужно указать позицию и значение параметра.

```
PreparedStatement ps = conn.prepareStatement("SELECT * FROM Students WHERE ID = ?");
ps.setInt(1, 2);
ResultSet rs = ps.executeQuery();
```

PreparedStatement поддерживает пакетную (batch) отправку SQL запросов, что значительно уменьшает траффик между клиентом и базой данных. Небольшой пример:

```
PreparedStatement ps = conn.prepareStatement("INSERT INTO Students(Name, GroupName) VALUES(?, ?);");
statement.setString(1, “Alex”);
statement.setString(2, "Tbz11");
statement.addBatch();
statement.setInt(1, “Sergey);
statement.setString(2, "Tbz11");
statement.addBatch();
statement.executeBatch();
```

Обработка результатов

Результатом запроса SELECT в базу является таблица (набор строк), которая сохраняется в объекте ResultSet. По строкам можно перемещаться вперед и назад. Для получения значений из определенной колонки текущей строки можно воспользоваться методами get<Type>(<Param>),
где Type - тип извлекаемого значения, Param - либо номер колонки (int), либо имя колонки (String).

```
ResultSet rs = stmt.executeQuery();
while (rs.next()) { // пока есть строки
  String name = rs.getString(2); // или rs.getString("Name");
}
rs.first(); // перейти к первой строке
rs.last(); // перейти к последней
rs.next(); // перейти к следующей
rs.previous(); // перейти к предыдущей
```

Закрытие ресурсов

На каждое соединение СУБД выделяет определенные ресурсы и количество их ограничено, поэтому после окончания работы с объектами соединения их нужно закрывать.

Транзакции в JDBC

По умолчанию каждое SQL-выражение автоматически коммитится при выполнении statement.execute() и подобных методов. Для того, чтобы открыть транзакцию сначала необходимо установить флаг autoCommit у соединения в false. А затем пользоваться методами commit() и rollback().

```
conn.setAutoCommit(false);
Statement st = conn.createStatement();
try {
    st.execute("INSERT INTO user(name) values('kesha')");
    conn.commit();
```

```
} catch (SQLException e) {
    conn.rollback();
}
```

Домашнее задание

- Сформировать таблицу товаров (id, prodid, title, cost) запросом из Java приложения.
  id - порядковый номер записи, первичный ключ
  prodid - уникальный номер товара
  title - название товара
  cost - стоимость
- При запуске приложения очистить таблицу и заполнить 10.000 товаров вида:
  id_товара 1 товар1 10
  id_товара 2 товар2 20
  id_товара 3 товар3 30
  ...
  id_товара 10000 товар10000 100010
  т.е. просто тестовые данные
- Написать консольное приложение, которое позволяет узнать цену товара по его имени, либо если такого товара нет, то должно быть выведено сообщение "Такого товара нет". Пример консольной комманды для получения цены: "/цена товар545"
- В этом же приложении должна быть возможность изменения цены товара(указываем имя товара и новую цену). Пример: "/сменитьцену товар10 10000"
- Вывести товары в заданном ценовом диапазоне. Консольная комманда: "/товарыпоцене 100 600"

Дополнительные материалы

1 Кей С. Хорстманн, Гари Корнелл Java. Библиотека профессионала. Том 1. Основы // Пер. с англ. - М.: Вильямс, 2014. - 864 с.

2 Стив Макконнелл Совершенный код // Пер. с англ. – СПб.: Питер, 2007. – 896 с.

3 Брюс Эккель Философия Java // 4-е изд.: Пер. с англ. – СПб.: Питер, 2016. – 1168 с.

4 Г. Шилдт Java 8. Полное руководство // 9-е изд.: Пер. с англ. - М.: Вильямс, 2015. - 1376 с.