# Основы разработки приложений на Java

## Уральский производственный центр АйТи - Дататех

## УГАТУ

### Лабораторная работа №5

**Цель:** Освоение методов работы с базами данных в JAVA с помощью API JDBC.

# Задание.

**Написать простую J2SE-программу для работы с базой из двух связанных таблиц.**

Работа ведется в двумя таблицами – ITEM (элементы c ID, названием, номером группы) и ITEMGROUP (некие группы элементов)

Исключения перехватывать в МИНИМАЛЬНО ВОЗМОЖНОМ числе мест.

Можно использовать небольшие фрагменты примеров (в т.ч. интернет).

Например, <https://www.tutorialspoint.com/hsqldb/>

1. Выбрать СУБД для работы:

* Derby DB - она же JavaDB, встроена в GlassFish, устанавливается вместе с NetBeans
* HSQLDB: Найти jar-файл c jdbc-драйвером, добавить в библиотеки проекта (при запуске!).

<http://cetral.maven.org/maven2/org/hsqldb/hsqldb/2.4.0/hsqldb-2.4.0.jar>  
можно подключаться к файлу без сервера, см. примеры по ссылке

<http://hsqldb.org/doc/guide/running-chapt.html>

2. Создать БД с именем **groupdb** средствами Netbeans или другими.

Выбрать предметную область, поменять в скрипте **createTables.sql** начальное содержимое базы (группы-студенты, группы товаров-товары, дома-жильцы, и т.д.)

3. Подключиться к базе с помощью графического или другого интерфейса, открыть SQL-скрипт подключения **createTables.sql** и выполнить его в текущем подключении.

* generated always as identity – автогенерируемый ID

4. Создать главный класс (DBTester).

Добавить метод создания подключения к созданной БД **groupdb** с помощью **DriverManager: Connection connectToDB().**

5. Добавить рабочий метод **test()** - **connectToDB** (), **doWork(Connection), conn.close()** (закрытие подключения) с обработкой исключений SQL (**void doWork() throws SQLException**).

**метод doWork() пока пустой.**

Вызвать этот метод из функции **main(). Сделайте общий try/catch блок, чтобы не перехватывать исключения в каждом методе работы с SQL.**

Добиться успешного подключения к БД.

6. Добавить функции вывода списка групп и элементов на консоль

**void viewGroups()**

**void viewItems()**

(select … from itemgroup, select … from item).

Использовать простой **Statement** и цикл просмотра **ResultSet**.

**(\*) Усложнение** – создать универсальную функцию печати ResultSet с заголовками, взятыми из ResultSetMetaData.

7. Добавить функцию получения ID группы по её имени **int getGroupID(String key)** Использовать **PreparedStatement**  с одним параметром. (SELECT … WHERE NAME=?)

Проверить работу, вызвав эту функцию в doWork()

8. Добавить функцию вывода на консоль списка элементов в заданной группе

**void viewItemsInGroup (int groupid)**

**void viewItemsInGroup (String groupname)**

(использовать getGroupId и второй запрос, либо один запрос с JOIN)

Проверить работу, вызвав эту функцию в doWork()

9. Добавить функцию создания и начального заполнения таблиц, если они отсутствуют в базе.

**void createTablesIfNeeded()**

(если проверка-SELECT выявляет отсутствие таблиц, по очереди вызываются запросы, перенесенные из createTables.sql)

10. Добавить функции **addItemToGroup**(String itemName, String groupName) и

**removeItemFromGroup**(String itemName, String groupName)

(если происходит ошибка из-за отсутствующей группы или нарушения уникальности –

возвращать false)

11. Реализовать чтение редактирование списка групп: функцию чтения файла items.txt формата

фрукты+яблоко   
инструменты-баян

(+=добавить, -=удалить)

* Просто по очереди выполняем **addItemToGroup**/**removeItemFromGroup**
* Файл обрабатывается отдельной транзакцией – если добавить/удалить не получилось, все операции отменяются **conn.rollback()** (иначе **conn.commit()** )
* По умолчанию транзакция открывается для каждого запроса. Чтобы отключить это поведение, используйте **setAutoCommit(false)**

12. Реализовать массовое редактирование списка групп: функцию чтения файла groups.txt формата

+фрукты

+овоща

-стройматериалы

-овоща

+овощи

**Для тренировки обязательно использовать изменяемый ResultSet!**

Схема применения:

* Файл считывается построчно (BufferedReader), формируются множества (Set) добавляемых/удаляемых групп.
* Добавлять и удалять группы надо только если они отсутствуют. Чтобы не делать множество проверочных запросов, предлагается считать все группы запросом SELECT с изменяемым ResultSet.
* **Однократно** просматривается ResultSet, определяется, какие записи подлежат удалению (**rs.deleteRow()**) , а какие нужно добавить (**rs.moveToInsertRow(); rs.updateString(…) ; rs.insertRow()** )
* +/cуществующая группа/ и -/отсутствующая группа/ игнорируются
* Файл обрабатывается отдельной транзакцией – если добавить/удалить не получилось, все операции отменяются **conn.rollback()** (иначе **conn.commit()** )

Пример скрипта создания базы данных:

DROP TABLE ITEM;

DROP TABLE ITEMGROUP;

CREATE TABLE ITEMGROUP (ID INTEGER PRIMARY KEY generated always as identity,

TITLE VARCHAR(100) UNIQUE NOT NULL);

CREATE TABLE ITEM (ID INTEGER PRIMARY KEY generated always as identity,

TITLE VARCHAR(100) UNIQUE NOT NULL, GROUPID INTEGER,

FOREIGN KEY (GROUPID) REFERENCES ITEMGROUP(ID) );

INSERT INTO ITEMGROUP(TITLE) VALUES('КОМПЬЮТЕРЫ');

INSERT INTO ITEMGROUP(TITLE) VALUES('ТЕЛЕФОНЫ');

INSERT INTO ITEMGROUP(TITLE) VALUES('ТЕЛЕВИЗОРЫ');

INSERT INTO ITEM(TITLE,GROUPID) VALUES('Apple',1);

INSERT INTO ITEM(TITLE,GROUPID) VALUES('Dell',1);

INSERT INTO ITEM(TITLE,GROUPID) VALUES('HTC',2);

INSERT INTO ITEM(TITLE,GROUPID) VALUES('Nokia',2);

INSERT INTO ITEM(TITLE,GROUPID) VALUES('LG',3);