Тема 10. Взаимодействие на Java с базите от данни (въведение)

Съдържание

- Дефиниране на модела на таблиците на базата;
- Създаване, четене, промяна и изтриване на данните в таблиците;
  - Команди за добавяне на нови данни;
  - Команди за четене на данни;
  - Команди за промяна на съществуващи данни;
  - Команди за изтриване на данни;
- Използване на Java Class за търсене на драйвер;
- Взаимодействие на езика с базата от данни;
- Настройка на връзката с базата от данни;
- Получаване на адреса на данновия източник (DSN)

Дефиниране на модела на таблиците на базата

Същност: Определя/променя структурата на таблиците в DB.

- Използва се езика за дефиниране на данни (**DDL-data definition language**)

Предназначение:

- Дефиниране;
- Промяна;
- Премахване на таблици в релационната база данни;

Дефиниране на модела на таблиците на базата

Програмно се създават SQL команди за създаване на таблици.

Съществуват и други команди на DDL:

- За дефиниране на тригери;
- Съхранени процедури;
- Индекси на таблици:
  - ускоряване на достъпа до данните;
- Изгледи-виртуални таблици.

Дефиниране на модела на таблиците на базата

Моделиране на данновия модел:

- Средства за проектиране на таблици. Фирмени:
  - MS Access;
  - MS SQL Server;
  - Oracle;
  - MySQL и др.
- Универсални-написани на Java.

Дефиниране на модела на таблиците на базата

Програмно създаване на таблица:

Пример:

```
CREATE TABLE StudentRegister (
  StudentFN CHAR(8) NOT NULL PRIMARY KEY,
  NAME CHAR(30) NOT NULL,
  PersonSex CHAR(3) NOT NULL,
  PersonTitle CHAR(10) NOT NULL  );
```

Дефиниране на модела на таблиците на базата

Програмна модификация:

```
ALTER TABLE StudentRegister
      ADD ,  UpdDate DATETIME   NULL;
```

Премахване:

```
DROP TABLE StudentRegister;
```

Създаване, четене, промяна и изтриване на данните в таблиците

Същност: За да се променят записаните данни в таблиците на релационната база се използват команди на SQL от езикът за промяна на данните (**DML- data manipulation language**).

Видове:

- Команди за добавяне на нови данни;
- Команди за четене на данни;
- Команди за промяна на съществуващи данни;
- Команди за изтриване на данни;

Команди за добавяне на нови данни

Пример: Команда за добавяне на данни (SQL ***INSERT)***

- Добавя нов запис в таблица на базата от данни:
  - Участва името на таблицата (**StudentRegister)** и списък от колони;
  - Списъкът от данни за съответните колони:
  - Символните данни се заграждат в апострофи;
  - Числовите полета са без апострофи;
  - Датата се представя по формат (на MS Access):

Команди за добавяне на нови данни

```
INSERT INTO StudentRegister ( StudentFN,
  NAME,
  PersonSex,
  PersonTitle,
 UpdDate)  VALUES ('17621610', 'Ivan', 'm', 'mr',
   '4/1/2017');
```

- Ако се изпълни заявка за добавяне на стойност към таблицата със същия ключ, командата за добавяне ще предизвика грешка.

Команди за четене на данни

Четене на данни от базата-командата на DML SQL SELECT.

Пример: четене на един или нула записи на таблицата:

- В условната част на командата се поставя условие за точно съвпадение на стойността на ключово поле на първичен ключ - колоната StudentFN да има стойност '17621610';
- Заявката може да намери (само един) или да не намери запис с такава ключова стойност;
- Не може да се получи повече от един записи ако ключовата стойност е уникална.

Команди за четене на данни

Пример - команда за четене:

- Четат се нула или повече записи:
  - В заявка за четене, в условната част на която, с помощта на клаузата WHERE, се изисква ключовата стойност да започва с низа '**17621**';
  - В SQL символът % в условието е указание за низ, съставен от символи на типа на полето;
    - '%**610**', '%62%' ...
  - Допълнително условие е сортирането на резултата от заявката по съдържанието на полето StudentFN;

Команди за четене на данни

Пример 1 за селектираща заявка:

```
SELECT * FROM StudentRegister
WHERE StudentFN = '17621610';
```

Пример 2 за селектираща заявка:

```
SELECT * FROM StudentTable
WHERE NAME = 'Ivan' AND StudentFN LIKE
   `17621%` ORDER BY      StudentFN;
```

Команди за промяна на съществуващи данни

Промяна на данни в базата от данни- команда на DML SQL ***UPDATE:***

- В примерната команда се специфицира таблицата и двойки поле = стойност които да се променят в нея;
- Допълнително може да има и условна част- клаузата WHERE, която е специфична и определя единствен (група) записи.

Команди за промяна на съществуващи данни

Промяна на уникален запис:

```
UPDATE StudentTable
  SET NAME = 'NewName',
    PersonSex = 'm'
WHERE StudentFN = '17621610';
```

Команди за изтриване на данни

За да се изтрият данни от релационна база данни се използва команда на DML SQL ***DELETE***.

- Пример за синтаксис на командата за изтриване на записи в таблицата StudentTable при условие, че те имат номер (полето StudentFN) '17621610' или същият е по-малък от `17621610`:

```
DELETE FROM StudentTable
WHERE       StudentFN = '17621610'
OR          StudentFN < `17621610'
```

Използване на Java Class за търсене на драйвер

Класовете в Java по време на изпълнение се представят от класа java.lang.Class;

- За всеки обект съществува обект от класа **Class, който може да се получи по време на изпълнение;**
- Обект от този клас позволява:
  - от референция да се определи класа;
  - да се получат неговите методи и други свойства;
  - да се определи структурата на обекта;
  - да се изпълнят неговите методи по време на изпълнение.

Процесът на получаване на информация по време на изпълнение се нарича "**рефлексия**"

Използване на Java Class за търсене на драйвер

Примерен клас за илюстрация на рефлексия:

```
import java.lang.reflect.Field;
public class Reflection {
    public int intField = 1;
    public String strField = "String value";
    public Reflection() {     }
```

(продължава)

Използване на Java Class за търсене на драйвер

```
public static void main(String[] args) {
// Търсене на клас по име. forName () е статичен метод,
// който връща обект от класа Class по дадено име като String
    try {
      Reflection objReflection = new Reflection();
      Class classReflection = objReflection.getClass();
      Class.forName("Reflection");
      // Class.forName("ReflectionBad");
      Field[] fields = classReflection.getFields( );
      for(int i=0; i<fields.length; i++) {
            System.out.println("field : "+fields[i].toString());
      }//field : public int Reflection.intField
    }
    catch ( ClassNotFoundException e ) {
// не може да намери класа
            System.out.println("No found class : "+e.toString());
// No found class : java.lang.ClassNotFoundException: ReflectionBad
        }
     }
}
```

(продължава)

Въпроси ?