

Бази данни



SAP GeekyCamp

Основи на базите от данни

- ❑ Йерархични
- ❑ Мрежови
- ❑ Обектно ориентирани
- ❑ Релационни
- ❑ No-SQL



ORACLE



PostgreSQL



Основи на базите от данни

Таблицы

Фак. №	Име	Фамилия	Стипен- дия	Редовно обучение	Дата на записване	Брой кредити
3123	Иван	Славов	250	1	10.09.2010	35
3124	Станислава	Петрова	300	1	NULL	20
1134	Ирена	Якимова	0	0	30.09.2011	25
3341	Петър	Петров	0	0	28.09.2012	5

Таблица 1. Students

Text data types:

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. Note: If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LOBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	<p>Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted.</p> <p>Note: The values are sorted in the order you enter them.</p> <p>You enter the possible values in this format: ENUM('X','Y','Z')</p>
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

Number data types:

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

Date data types:

Data type	Description
DATE()	<p>A date. Format: YYYY-MM-DD</p> <p>Note: The supported range is from '1000-01-01' to '9999-12-31'</p>
DATETIME()	<p>*A date and time combination. Format: YYYY-MM-DD HH:MI:SS</p> <p>Note: The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'</p>
TIMESTAMP()	<p>*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS</p> <p>Note: The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC</p>
TIME()	<p>A time. Format: HH:MI:SS</p> <p>Note: The supported range is from '-838:59:59' to '838:59:59'</p>
YEAR()	<p>A year in two-digit or four-digit format.</p> <p>Note: Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069</p>

*Even if DATETIME and TIMESTAMP return the same format, they work very differently. In an INSERT or UPDATE query, the TIMESTAMP automatically set itself to the current date and time. TIMESTAMP also accepts various formats, like YYYYMMDDHHMISS, YYMMDDHHMISS, YYYYMMDD, or YYMMDD.

Основи на базите от данни

Ключове, индекси и ограничения

Фак. №	Име	Идентификация	Редовно обучение	Дата на записване	Брой кредити	
3123	Иван	Славов	250	1	10.09.2010	35
3124	Станислава	Петрова	300	1	NULL	20
1134	Ирена	Якимова	0	0	30.09.2011	25
3341	Петър	Петров	0	0	28.09.2012	5

Таблица 1. Students

РК - идентифицира по уникален начин всеки запис
Индекс – увеличава скоростта за достъп
Ограничение – ограничава стойностите на атрибут(и)

Индекс (Index)

Първичен ключ
(Primary Key), РК

Ограничение,
(Constraint)

Основи на базите от данни

Релации

T3. Student_has_course

DEP_ID	Фак. №	Име	Фамилия	Брой кредити
1	3123	Иван	Славов	35
3	3124	Станислава	Петрова	20
3	1134	Ирена	Якимова	25

Таблица 1. Students

Фак.№	Курс_№
3123	1
3123	3
3124	3



№	Име	Номер стая	Тел. №
1	Автоматика	2350	9652612
2	ФКСУ	1441	9652054
3	Транспорт	9308	9652602

Таблица 2. Departments

№	Име	Стая	Брой кредити
1	Java	1350	5
2	C#	2441	5
3	SCRUM	8308	5

Таблица 3. Courses

Основи на базите от данни

Видове SQL

DML

DATA MANIPULATION LANGUAGE

- ОБРАБОТВАНЕ НА ДАННИ
- ПРИМЕРИ – SELECT, INSERT, UPDATE, DELETE

DDL

DATA DEFINITION LANGUAGE

- ДЕФИНИРАНЕ НА ДАННИ
- ПРИМЕРИ – CREATE, ALTER, DROP

SQL/PSM

SQL/PERSISTENT STORED MODULES

- ПРОЦЕДУРЕН ЕЗИК ЗА STORED PROCEDURES
- ПРИМЕРИ – PL/SQL, TRANSACT-SQL

Основи на базите от данни

DDL и DML(SQL) примери

```
CREATE TABLE STUDENTS (FN NUMBER,  
FIRST_NAME VARCHAR2(20), LAST_NAME VARCHAR2(20),  
SCHOLARSHIP NUMBER(10,2), REGULAR_FORM NUMBER(1),  
REGISTERED_DATE DATE, CREDITS NUMBER(3,0),  
CONSTRAINT PK PRIMARY KEY(FN))
```

DDL

FN	FIRST_ NAME	LAST_ NAME	Scholars hip	Regular form	Registered_ date	Credits
----	-------------	------------	--------------	--------------	------------------	---------

```
SELECT * FROM STUDENTS  
WHERE CREDITS > 20 AND REGULAR_FORM=0
```

DML(SQL)

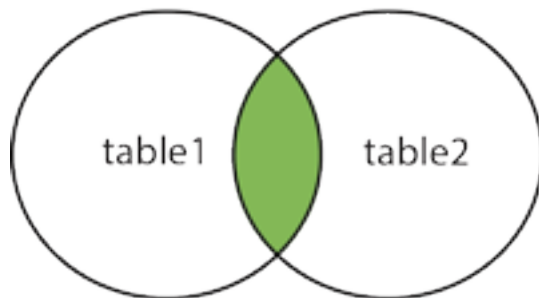
FN	FIRST_ NAME	LAST_ NAME	Scholars hip	Regular form	Registered_ date	Credits
1134	Ирена	Якимова	0	0	30.09.2011	25

Основи на базите от данни

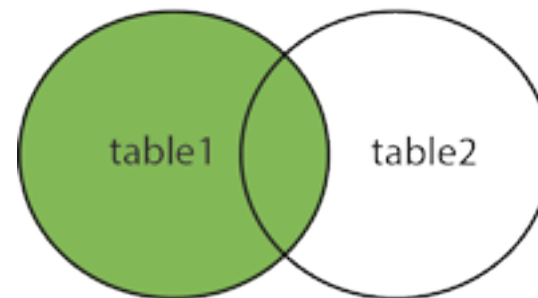
JOIN

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

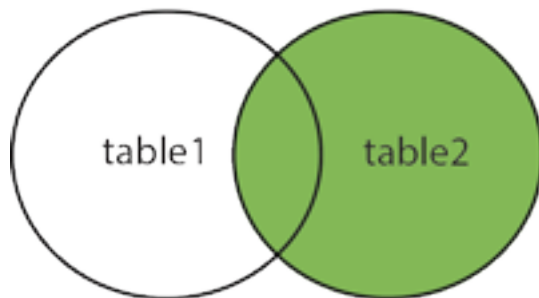
INNER JOIN



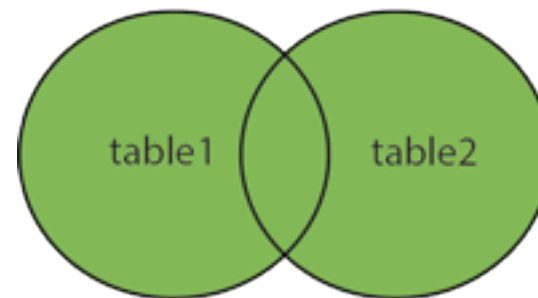
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



JDBC

Java Database Connectivity

- ❑ Java базирана технология за лесен достъп до БД
- ❑ Предоставя API за четене и писане на данни в БД
- ❑ Единен подход за работа с различни БД
- ❑ Без нужда от инсталация

JDBC

API – java.sql.Connection

1. Създаване на връзка (connection) към БД

```
try {  
    //Зареждане на MySQL jdbc драйвер, всяка  
    //БД има свой собствен драйвер  
    Class.forName("com.mysql.jdbc.Driver");  
    // Установяване на връзка с БД MySQL  
    Connection cn = DriverManager.getConnection(  
        "jdbc:mysql://localhost:port/db1?user=sqluser&password=sqluserpw");  
} catch (Exception e) {  
    ...//обработка изключение  
}
```

Connection String

Хост

име на БД

Потреб. име

парола

2. Затваряне на връзка

```
if (!cn.isClosed()) { cn.close(); }
```

Throws
SQLException

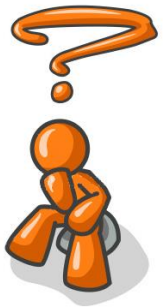
JDBC

API – java.sql.Statement, java.sql.ResultSet

```
Connection con = null; String userInputFN = "1325";
try {
    . . . //УСТАНОВЯВАНЕ НА ВРЪЗКА С БД
    Statement cmd = con.createStatement();
    ResultSet rs = cmd.executeQuery("SELECT s.first_name,
s.credits FROM students s WHERE s.fn=" + userInputFN);

while (rs.next()) { //ОБХОЖДАНЕ НА РЕЗУЛТАТИТЕ
    Student s = new Student();
    s.setFirstName(rs.getString("first_name"));
    s.setCredits(rs.getInt(2));
    System.out.println("Student " + s.getFirstName() + "added");
    . . . // ДРУГИ ОБРАБОТКИ
}
} finally {
    . . . // ЗАТВАРЯНЕ НА ВРЪЗКАТА КЪМ БД
}
```

Проблем?



JDBC

API – java.sql.Statement, java.sql.ResultSet – недостатъци



Непрекъснато сглобяване и компилиране на SQL заявките



Хакване чрез SQL Injection

```
cmd.executeQuery("SELECT s.first_name, s.credits FROM students  
s WHERE s.fn=" + userInputFN);
```

a) userInputFN = 1325 -> SELECT ... WHERE s.fn=1325 ✓

b) userInputFN = 1325 OR 1=1 -> SELECT ... WHERE s.fn=1325 OR 1=1 ✗

JDBC

API – java.sql.PreparedStatement

```
Connection con = null; String userInputFN = "1325";
try {
    . . . //УСТАНОВЯВАНЕ НА ВРЪЗКА С БД
    PreparedStatement cmd = con.prepareStatement("SELECT
s.first_name, s.credits FROM students s WHERE s.fn=?");
    ResultSet rs = cmd.executeQuery();
    rs.setString(1, userInputFN);

while (rs.next()) { //ОБХОЖДАНЕ НА РЕЗУЛТАТИТЕ
    Student s = new Student();
    s.setFirstName(rs.getString("first_name"));
    s.setCredits(rs.getInt(2));
    System.out.println("Student " + s.getFirstName() + " added");
    . . . // ДРУГИ ОБРАБОТКИ
}
} finally { . . . // ЗАТВАРЯНЕ НА ВРЪЗКАТА КЪМ БД }
```


JDBC

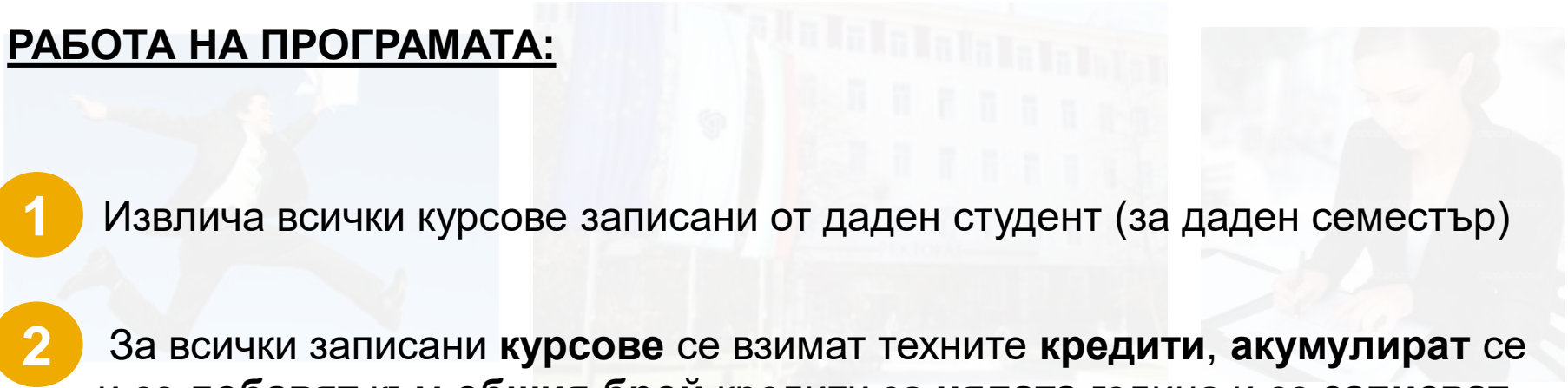
API – Транзакции



JDBC

API – Транзакции. Проблемът.

РАБОТА НА ПРОГРАМАТА:

- 
- 1 Извлича всички курсове записани от даден студент (за даден семестър)
 - 2 За всички записани **курсове** се взимат техните **кредити**, **акумулират** се и се **добавят** към **общия брой** кредити за **цялата** година и се **записват в БД**
 - 3 При взети 30 или повече кредита, отбелязва учебната година за студента като успешна.

Какво би станало ако стъпка 3 се провали?

Транзакционност

- ❑ Транзакцията е блок от код, който съдържа една или повече SQL заявки.
- ❑ Транзакцията е атомарна операция и всички заявки в нея или се изпълняват или нито една от тях не се изпълнява.

JDBC

Решението.

```
try {
    int studentCredits = getCurrentCredits(studentFn); //за семестър
    dbConnection.setAutoCommit(false); // стартираме транзакцията
    1 ResultSet stCourses = sqlGetStudCourses.executeQuery();
    2 while (stCourses.next()) {
        studentCredits+=stCourses.getInt("credits");
    }
    preparedStmtUpdateCredits.setInt(creditsIdx, studentCredits);
    preparedStmtUpdateCredits.executeUpdate();
    3 if (studentCredits >= 30) {
        preparedStmtUpdateCredits.setInt("year_completed", 1);
        preparedStmtYearCompleted.executeUpdate();
    }
    dbConnection.commit(); // приключваме транзакцията
} catch (Exception e) {
    try {
        System.err.print("Transaction is being rolled back");
        dbConnection.rollback(); //при проблем транзакцията се "отменя"
    } catch (SQLException exc) { exc.printStackTrace(); } }
```

Задача

Моделирайте база данни на приложение за университети, което трябва да може да добавя, изтрива и извежда списък от студенти.

- ❑ Всеки студент има факултетен номер, първо име, фамилия.
- ❑ *(Бонус 1)* Всеки студент може да записва много курсове. Всеки курс има име, описание и кредити, които носи. Добавете възможност за добавяне, изтриване и извеждане на списък с курсове (както отделно така и за всеки студент).
- ❑ *(Бонус 2)* Всеки студент е записан точно в един факултет. Един факултет може да има много студенти. Добавете възможност за добавяне, изтриване и извеждане на списък с факултети, както и списък на студентите, записани в даден факултет.
- ❑ *(Бонус 3)* Направете така, че когато изтриете даден факултет всички студенти записани в него също да бъдат изтрети.
- ❑ *(Бонус 4)* Използвайте JDBC за да манипулирате данните в таблиците.