MySQL

Principles of Database Design

Ehsan Edalat

Ahmad Asadi

Parham Alvani

Outline

- Installing MySQL on Windows [Hide]
- MySQL Workbench [Hide]
- Basis
- Joins
- MySQL in Java (JDBC)

Basis

- Show/create Databases
- Show/Create table
- Insertion
- Update
- Delete
- Query
- Aggregate Functions

Show/create Databases


```
show databases;
create database TA_DB character set
= utf8;
```

Show/Create Tables


```
show tables;
create table students (
      std_num char(7) primary key,
      name varchar(255) not null,
      family varchar(255),
      age int(2) unsigned default 20,
      gender bit(1) default 0,
      grade int(2) not null default 0,
      check (grade>=0 and grade <=20)
);
```

Data Types (STRING)

CHAR(*size*)	Maximum size of 255 characters.	Where *size* is the number of characters to store. Fixed-length strings. Space padded on right to equal *size* characters.
VARCHAR(*size*)	Maximum size of 255 characters.	Where ***size*** is the number of characters to store. Variable-length string.
TINYTEXT(*size*)	Maximum size of 255 characters.	Where ***size*** is the number of characters to store.
TEXT(*size*)	Maximum size of 65,535 characters.	Where ***size*** is the number of characters to store.
MEDIUMTEXT(*size*)	Maximum size of 16,777,215 characters.	Where ***size*** is the number of characters to store.
LONGTEXT(*size*)	Maximum size of 4GB or 4,294,967,295 characters.	Where ***size*** is the number of characters to store.

Data Types (NUMERIC)

Data Type Syntax	Maximum Size	Explanation
TINYINT(*m*)	Very small integer value. Signed values range from -128 to 127. Unsigned values range from 0 to 255.	
INT(*m*)	Standard integer value. Signed values range from -2147483648 to 2147483647. Unsigned values range from 0 to 4294967295.	
DECIMAL(*m*,*d*)	Unpacked fixed point number. ***m*** defaults to 10, if not specified. ***d*** defaults to 0, if not specified.	Where ***m*** is the total digits and ***d*** is the number of digits after the decimal.
BOOL	Synonym for TINYINT(1)	Treated as a boolean data type where a value of 0 is considered to be FALSE and any other value is considered to be TRUE.

Data Type (DATE/TIME)

Data Type Syntax	Maximum Size	Explanation
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'YYYY-MM-DD'.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIMESTAMP(*m*)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'.
YEAR[(2|4)]	Year value as 2 digits or 4 digits.	Default is 4 digits.

Constraints

- PRIMARY KEY

```
primary key (col_name);
```

- FOREIGN KEY

```
foreign key (col_name) references
        Table_name(col_name);
```

- Check

```
check (exp);
```

- Data type
  - [NOT NULL | NULL]
  - [DEFAULT ***default value***]
  - [AUTO_INCREMENT]
  - [UNIQUE [KEY]] | [PRIMARY KEY]

Insertion


```
insert into students values (
        '9031066',
        'ehsan',
        'edalat',
        23,0,0
);


insert into students (std_num, name, family, age, gender)
values(
        '9031062',
        'hamid',
        'ramezany',
        22,0,0
);
```

Update


```
update students set
      std_num = '9031806',
      name = 'seyed',
      family = 'ahmadpanah',
      age = 19, gender = 0,
      grade = 0
 where std_num = '9031066';
```

Delete

- Delete students with (age > 30):

```
delete from students where age > 30;
```

Query

- Select query structure:

select

$A_1, A_2, \ldots, A_n$

from

$r_1, r_2, \ldots, r_m$

where *P*;

- Example:

```
select name from students where age > 20;
```

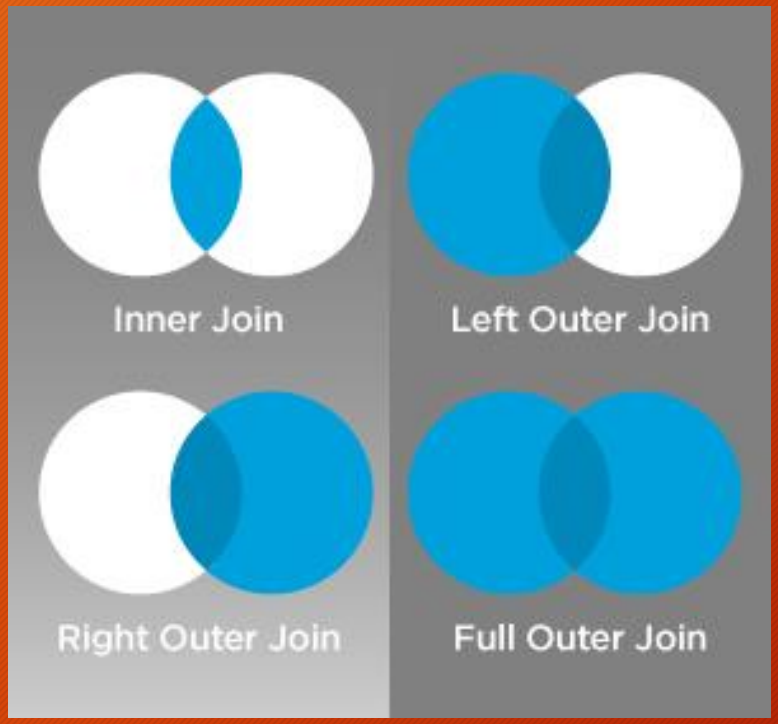
Aggregate Functions

- **avg:** average value
- **min:** minimum value
- **max:** maximum value
- **sum:** sum of values
- **count:** number of values

Joins

- Natural Join (Inner Join)
- Full Outer Join
- Left Outer Join
- Right Outer Join

Joins



Joins

- **INNER JOIN**: Returns all rows when there is at least one match in BOTH tables
- **LEFT JOIN**: Return all rows from the left table, and the matched rows from the right table
- **RIGHT JOIN**: Return all rows from the right table, and the matched rows from the left table
- **FULL JOIN**: Return all rows when there is a match in ONE of the tables

SQL Natural Join (Inner Join)

- List the names of instructors along with the course ID of the courses that they taught.

```
select name,
course_id from
instructor, teaches
where instructor.ID
= teaches.ID;

select name,
course_id from
instructor natural
join teaches;
```

SQL Joins

- Left outer join

```
left [outer] join
```

- Right outer join

```
right [outer] join
```

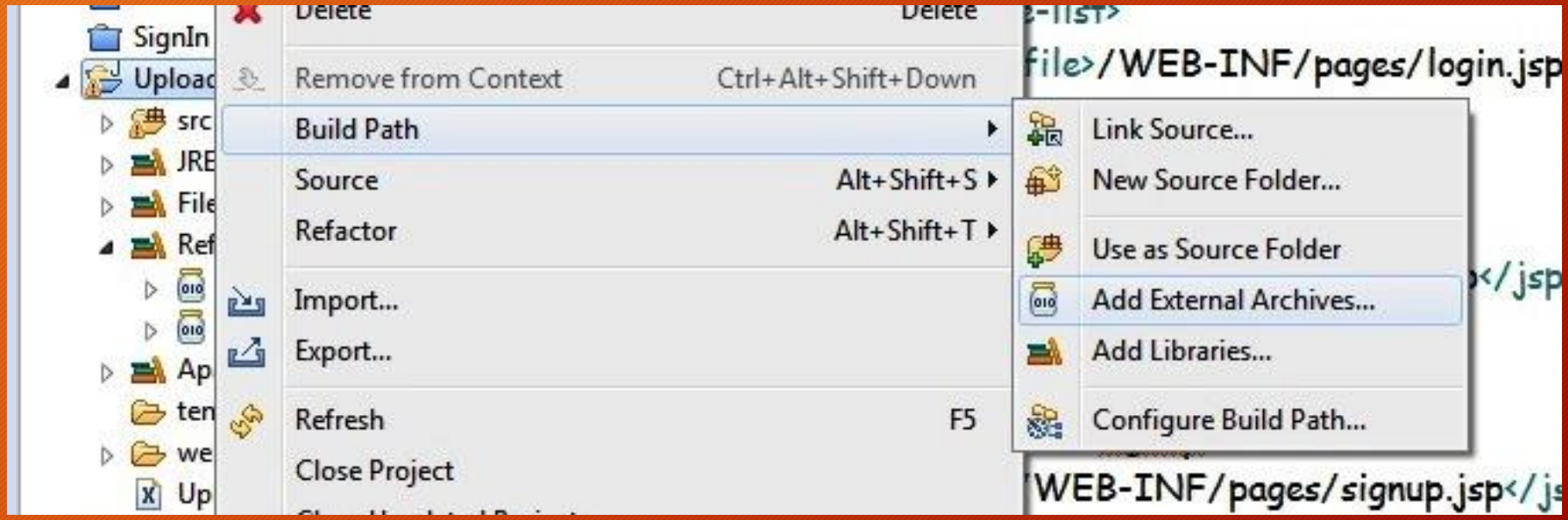
- Full outer join

```
outer join
```

MySQL in Java

- Client-Server Architecture
- Your Java Program: Client ☺
- MySQL: Server ☺
- JDBC: Your connection solution
- Download Link: https://www.mysql.com/products/connector/

Inserting JDBC to your Java Code (Manual)



JDBC Connecting Phase


```
try{
        Class.forName("com.mysql.jdbc.Driver");
} catch (ClassNotFoundException cnfe) {
        System.out.println("Error loading driver");
}
try {
        DriverManager.registerDriver(new com.mysql.jdbc.Driver());
        connection =
        DriverManager.getConnection("jdbc:mysql://localhost/TA_DB"+
"?useUnicode=true&characterEncoding=UTF-8", "ehsan", "*****");
} catch (SQLException e1) {
        e1.printStackTrace();
}
```

JDBC
Create Database/Table/Insert/Update/Delete


```
try {
      statement =
      connection.createStatement();
      statement.executeUpdate("insert into
students values
('9031066','ehsan','edalat',23,0,0)");
} catch (SQLException sqle) {
      System.out.println("Could not insert
tuple. " + sqle);
}
```

JDBC
(Query)


```
try {
      ResultSet rs =
      statement.executeQuery("select * from
students;");
      while (rs.next()) {
            System.out.println(rs.getString("name"));
      }
} catch (SQLException e) {
      e.printStackTrace();
}
```