



MySQL

Principles of Database Design

Ehsan Edalat

Parham Alvani

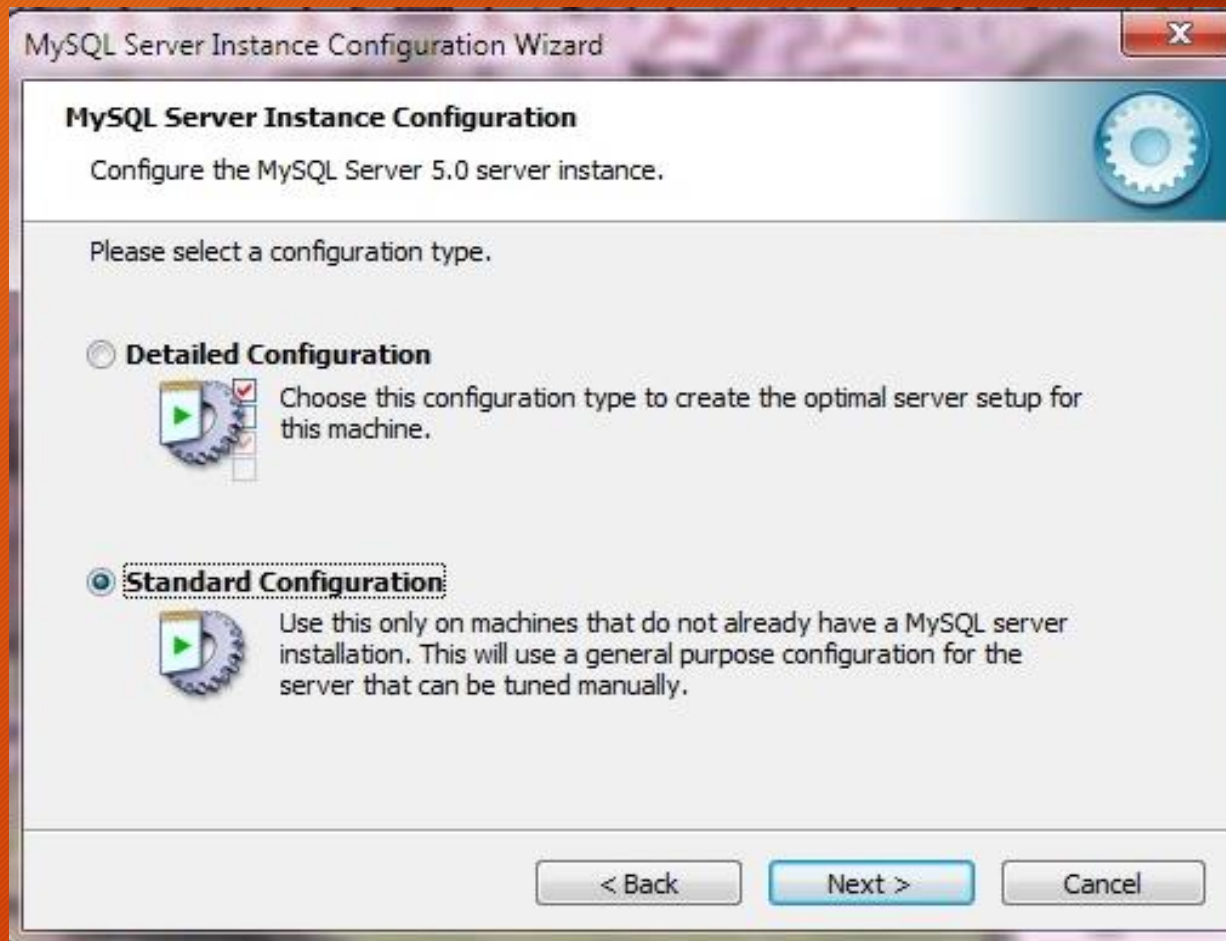
Outline

2

- Installing MySQL on Windows
- MySQL Workbench
- Some Useful Commands
- Joins
- MySQL in Java (JDBC)

Installing MySQL on Windows

3



Installing MySQL on Windows

4



Installing MySQL on Windows

5

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.

Please set the security options.

☒ **Modify Security Settings**

 New root password: Enter the root password.

Confirm: Retype the password.

☐ Enable root access from remote machines

☐ Create An Anonymous Account

 This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

< Back Next > Cancel

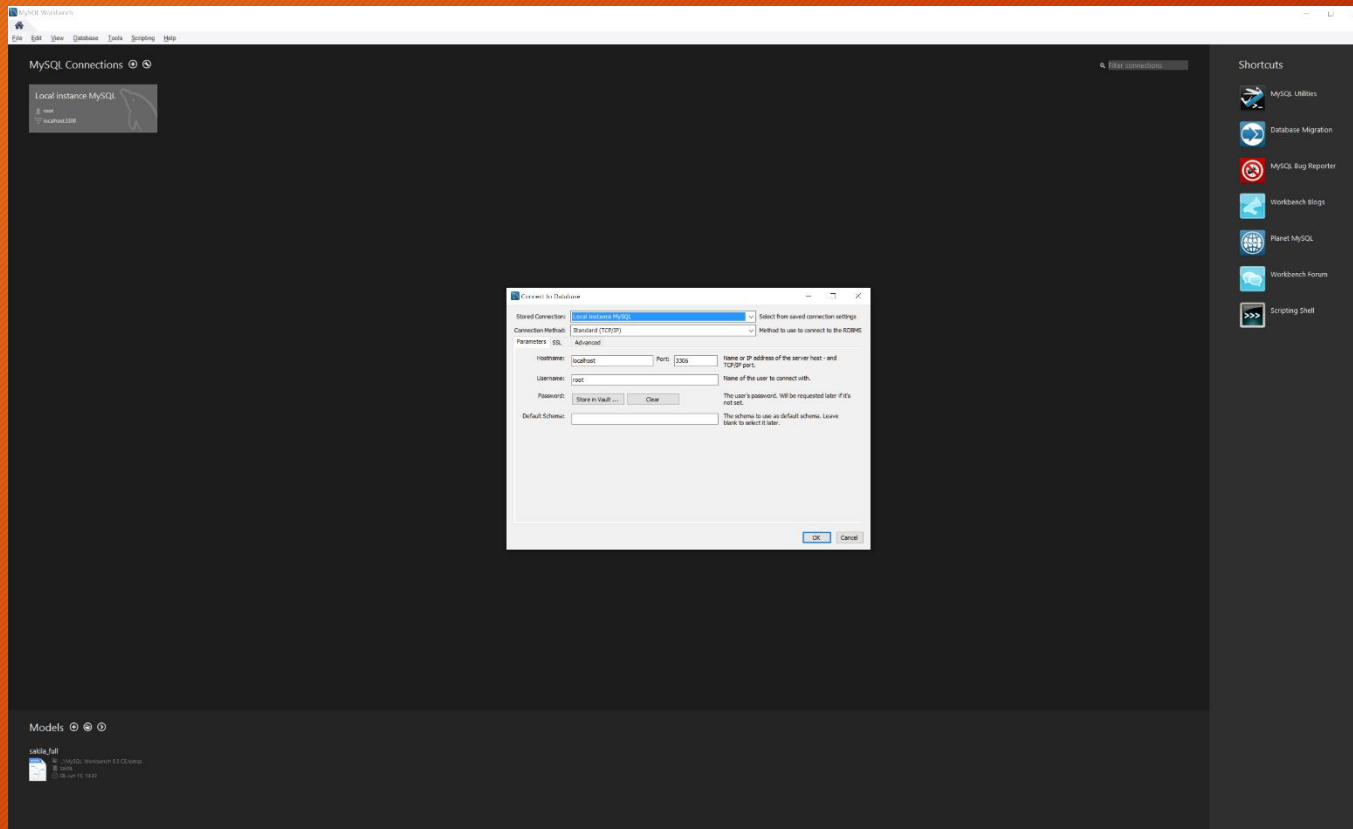
MySQL Workbench

6

- Design
- Develop
- Administer
- Database Migration
- Visual Performance Dashboard

MySQL Workbench

7



MySQL Workbench

8

The screenshot displays the MySQL Workbench interface with the following components:

- Navigator:** Shows the server status, client connections, and the current database schema.
- Query Editor:** Contains two SQL queries. The first query selects actor information from the 'sakila' database. The second query selects film information from the 'film' table.
- Result Set:** Displays the results of the second query, showing a list of films with columns: film_id, title, and description.
- Schema Browser:** Shows the 'film' table structure with columns: film_id, title, description, release_year, original_language_id, rental_duration, rental_rate, and length.
- SQL Snippets:** Provides a list of common SQL snippets for the SELECT statement.
- Output Panel:** Shows the execution results of the queries, including the time taken and the number of rows affected.

Query 1:

```
SELECT 'actor', 'actor_id',  
       'actor', 'first_name',  
       'actor', 'last_name',  
       'actor', 'last_update'  
FROM 'sakila', 'actor';
```

Query 2:

```
SELECT 'film', 'film_id',  
       'film', 'title',  
       'film', 'description',  
       'film', 'release_year',  
       'film', 'language_id',  
       'film', 'original_language_id',  
       'film', 'rental_duration',  
       'film', 'rental_rate',  
       'film', 'length',  
       'film', 'replacement_cost',  
       'film', 'rating',  
       'film', 'special_features',  
       'film', 'last_update'
```

Result Set:

film_id	title	description
1	ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies
2	ACE GOLDFINGER	A Epic Drama of a Feminist And a Mad Scientist who must Find a Car in Ancient China
3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberjack in A Balcony Factory
4	AFFAIR PREJUDICE	A Fanciful Documentary of a Friabee And a Lumberjack who must Chase a Monkey in A Shark Tank
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a Forensic Psychologist in
6	AGENT TRUMAN	A Intrepid Panorama of a Robot And a Boy who must Escape a Sumo Wrestler in Ancient China
7	AIRPLANE SIERRA	A Touching Saga of a Hunter And a Butler who must Discover a Butler in A Jet Boat
8	AIRPORT POLLOCK	A Epic Tale of a Moose And a Girl who must Confront a Monkey in Ancient India
9	ALABAMA DEVIL	A Thoughtful Panorama of a Database Administrator And a Mad Scientist who must Outgun a Mad Scie

Schema Browser:

Table: film

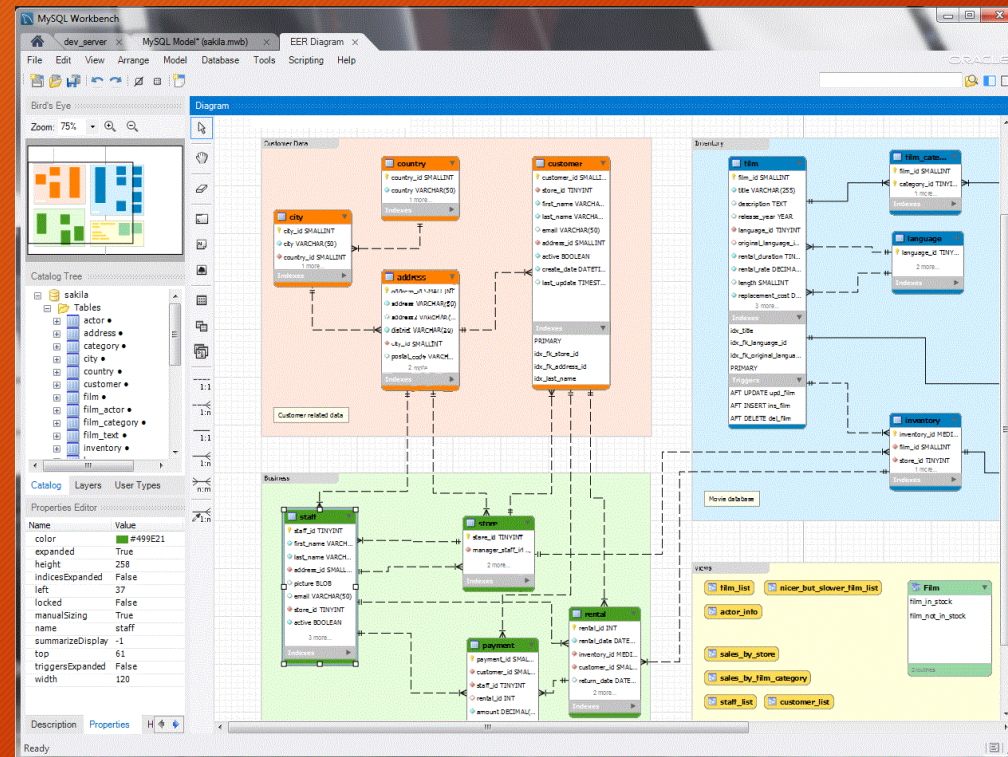
Column	Type
film_id	smallint(5) UNSIGNED
title	varchar(255)
description	text
release_year	year(4)
original_language_id	tinyint(3) UNSIGNED
rental_duration	tinyint(3) UNSIGNED
rental_rate	decimal(4,2)
length	smallint(5) UNSIGNED

Output Panel:

Time	Action	Message	Duration / Fetch
4 16:12:02	SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE=TRADITIONAL	0 row(s) affected	0.000 sec
5 16:12:02	CREATE TABLE actor (actor_id SMALLINT UNSIGNED NOT NULL AUTO...	Error Code: 1050. Table 'actor' already exists	0.000 sec
6 16:13:59	SELECT film 'film_id', film 'title', film 'description', film 'release_year...	1000 row(s) returned	0.031 sec / 0.000 sec

MySQL Workbench

9



Some Useful commands

10

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

Show/create Databases

11

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

Show/Create Tables

12

```
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)

13

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

Data Types (NUMERIC)

14

Data Type Syntax	Maximum Size	Explanation
TINYINT(<i>m</i>)	Very small integer value. Signed values range from -128 to 127. Unsigned values range from 0 to 255.	
INT(<i>m</i>)	Standard integer value. Signed values range from -2147483648 to 2147483647. Unsigned values range from 0 to 4294967295.	
DECIMAL(<i>m</i> , <i>d</i>)	Unpacked fixed point number. <i>m</i> defaults to 10, if not specified. <i>d</i> defaults to 0, if not specified.	Where <i>m</i> is the total digits and <i>d</i> 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)

15

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(<i>m</i>)	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

16

- 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

17

```
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

18

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

Delete

19

- Delete students with (age > 30):

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

Query

20

- Select query structure:

select

A_1, A_2, \dots, A_n

from

r_1, r_2, \dots, r_m

where P ;

- Example:

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


Aggregate Functions

21

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

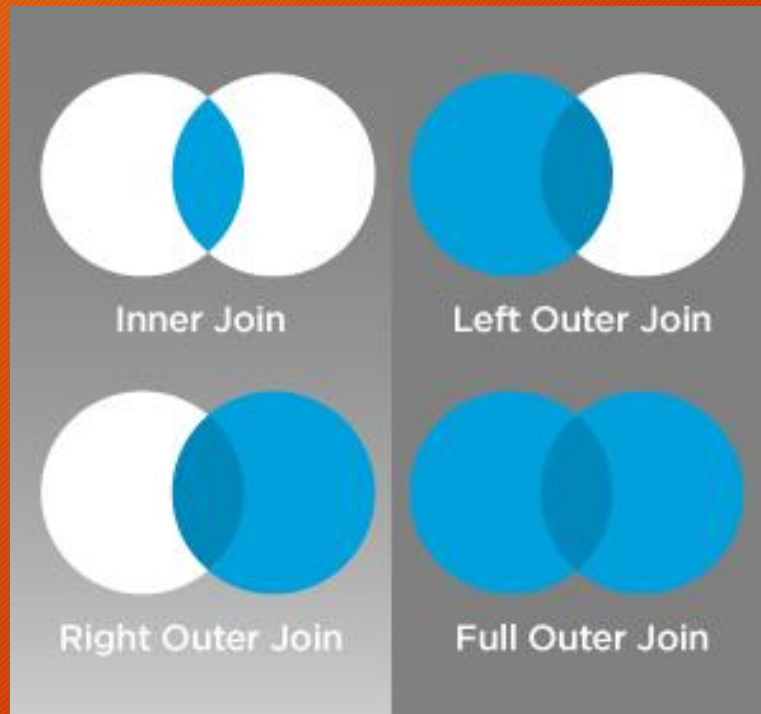
Joins

23

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

Joins

24



Joins

25

- **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)

26

- 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

27

- Left outer join

```
left [outer] join
```

- Right outer join

```
right [outer] join
```

- Full outer join

```
outer join
```

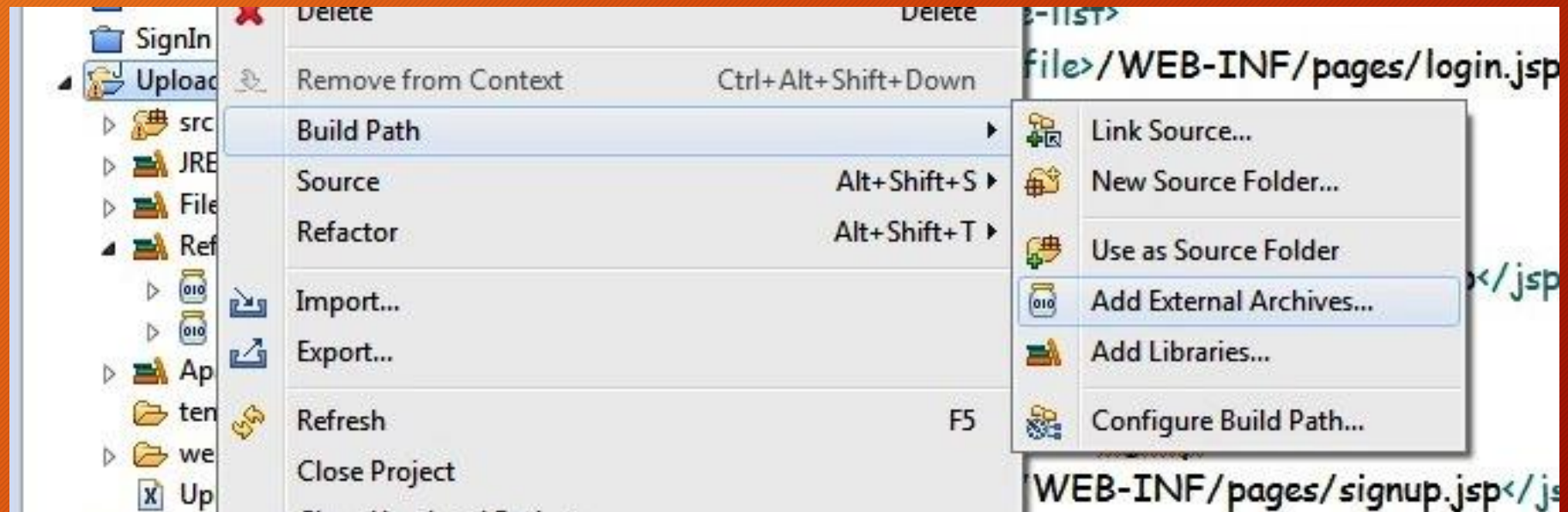
MySQL in Java

28

- 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)

29



JDBC

Connecting Phase

30

```
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

31

```
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)

32

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