SOEN 387

WEB-BASED ENTERPRISE APPLICATIONS DESIGN

TUTORIAL - 4

Using Databases

By

Vasu Ratanpara

Agenda

- ✓ What is Database & Database system ?
- ✓ Why Use a Database System?
- ✓ SQL : an example
- ✓ SQL: COMMIT and ROLLBACK
- ✓ BLOBS (Binary Large Objects)
- ✓ Insert files into a MySQL table with BLOB type
- ✓ JDBC (Java Database Connectivity)
- ✓ Query execution Best Practices
- ✓ SQL output parameters in stored procedure
- ✓ Exercise Query Execution using JDBC
- ✓ How to prevent SQL injection?

What is Database?

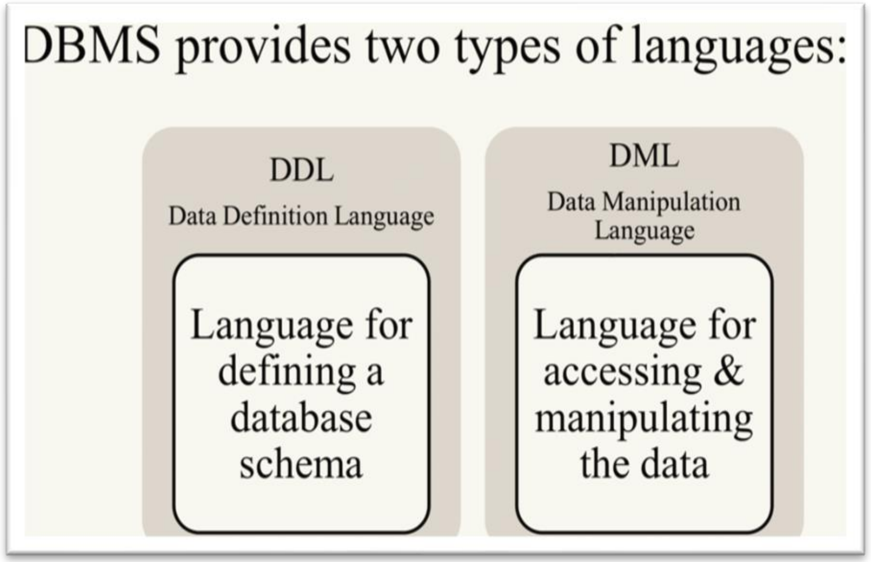
Database

- ✓ A large and persistent collection of (more-or-less similar) pieces of information organized in a way that facilitate efficient retrieval and modification
- ✓ The structure of the database is determined by the abstract data model that is used
- ✓ Examples:
  - List of names, addresses, and phone numbers of your friends
  - Information about employees, departments, salaries, managers, etc. in a COMPANY
  - Information about students, courses, grades, professors, etc. in a UNIVERSITY
  - Information about books, users, etc. in a LIBRARY

What is Database System?

Database System

Database Management System (DBMS) : Is a program (or set of programs) that manages details related to storage and access for a database.


DBMS provides two types of languages:
DDL
Data Definition Language
Language for defining a database schema
DML
Data Manipulation Language
Language for accessing & manipulating the data

Why Use a Database System?

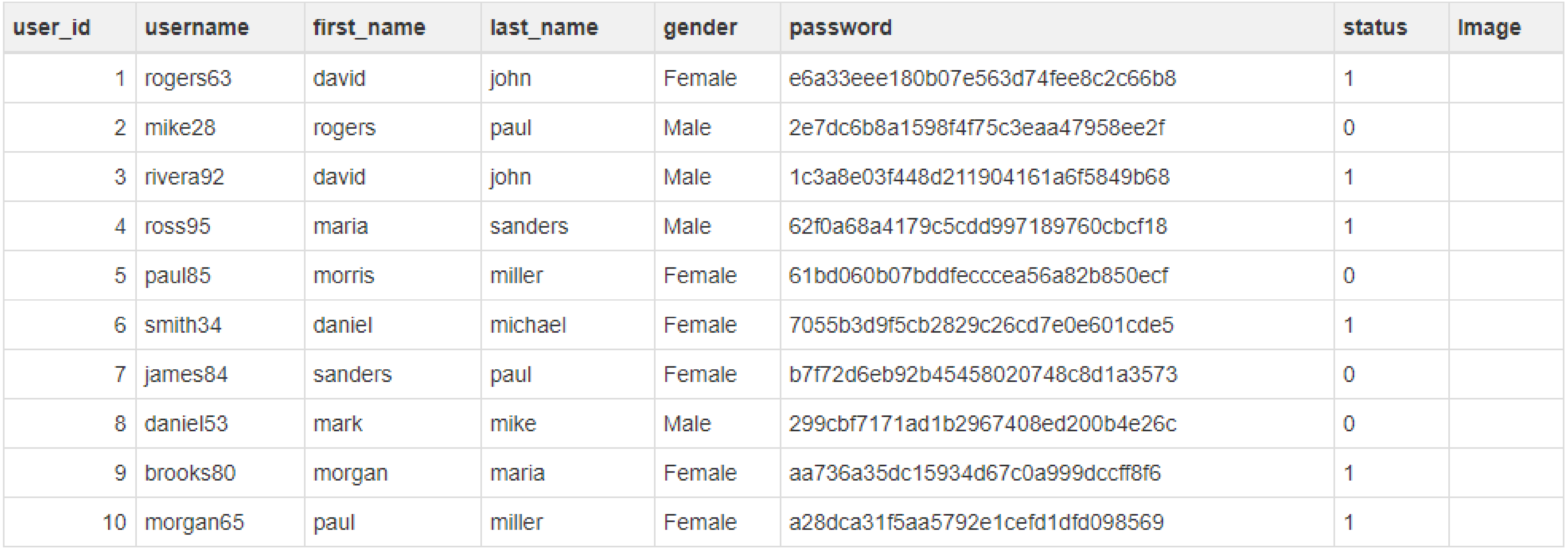
Database System

- ✓ Database systems have concentrated on providing solutions for all of these issues for scaling up Web applications in :
  - Performance
  - Scalability
  - Maintenance
  - Data Integrity
  - Transaction support
- ✓ While systems differ in their support, most offer some support for all of these.

Example of SQL

SQL : An Example

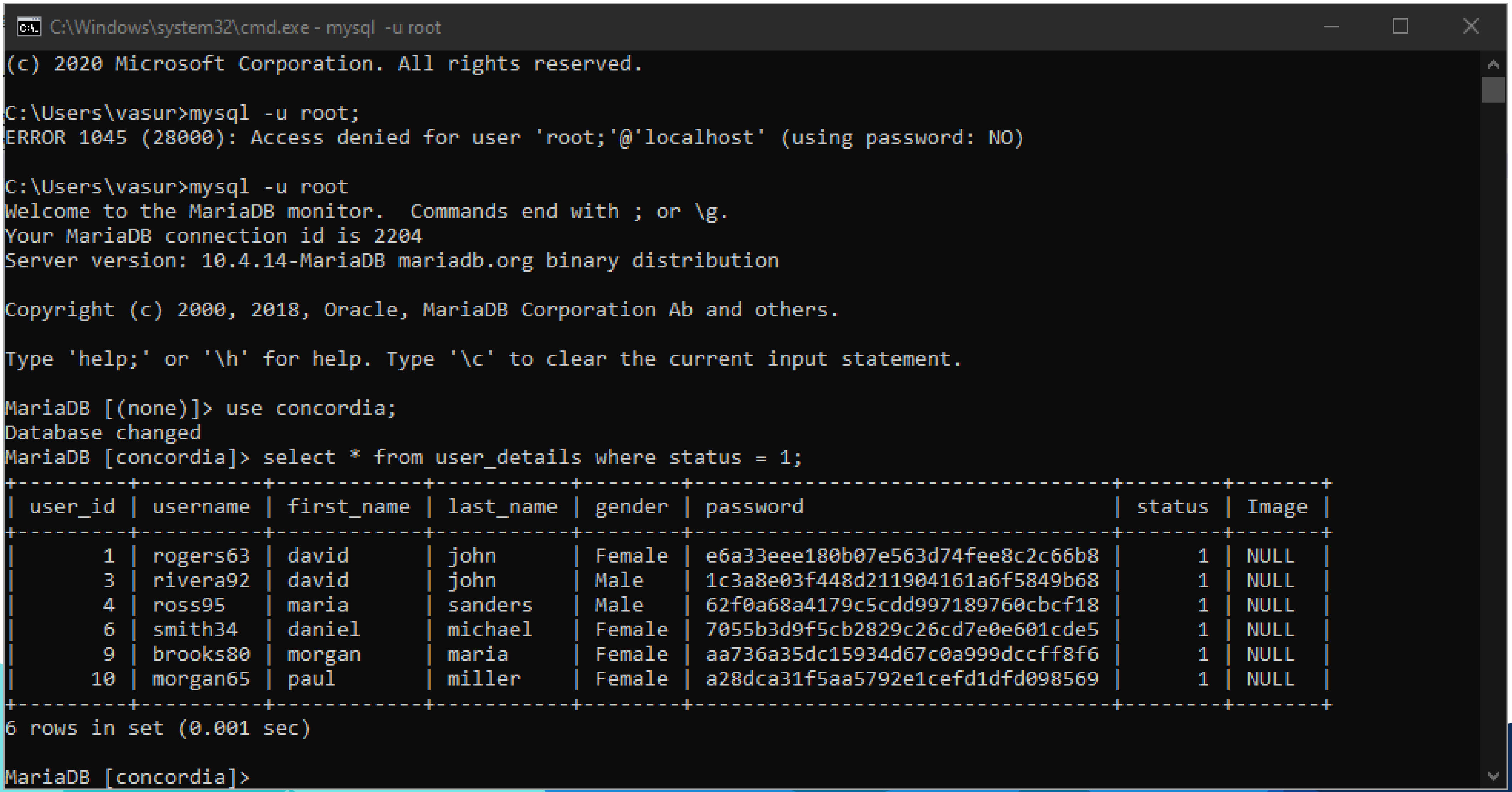
- ✓ Assume we have database of Concordia in which we have all the information about users like first name, last name, etc. We wish to know how many accounts are deactivated.



user_id	username	first_name	last_name	gender	password	status	Image
1	rogers63	david	john	Female	e6a33eee180b07e563d74fee8c2c66b8	1	
2	mike28	rogers	paul	Male	2e7dc6b8a1598f4f75c3eaa47958ee2f	0	
3	rivera92	david	john	Male	1c3a8e03f448d211904161a6f5849b68	1	
4	ross95	maria	sanders	Male	62f0a68a4179c5cdd997189760cbcf18	1	
5	paul85	morris	miller	Female	61bd060b07bddfecccea56a82b850ecf	0	
6	smith34	daniel	michael	Female	7055b3d9f5cb2829c26cd7e0e601cde5	1	
7	james84	sanders	paul	Female	b7f72d6eb92b45458020748c8d1a3573	0	
8	daniel53	mark	mike	Male	299cbf7171ad1b2967408ed200b4e26c	0	
9	brooks80	morgan	maria	Female	aa736a35dc15934d67c0a999dccff8f6	1	
10	morgan65	paul	miller	Female	a28dca31f5aa5792e1cefd1dfd098569	1	

SQL : An Example

mysql> select * from user_details where status = 1;



```
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\vasur>mysql -u root;
ERROR 1045 (28000): Access denied for user 'root;'@'localhost' (using password: NO)

C:\Users\vasur>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2204
Server version: 10.4.14-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use concordia;
Database changed
MariaDB [concordia]> select * from user_details where status = 1;
+---------+----------+------------+-----------+--------+----------------------------------+--------+-------+
| user_id | username | first_name | last_name | gender | password                         | status | Image |
+---------+----------+------------+-----------+--------+----------------------------------+--------+-------+
|       1 | rogers63 | david      | john      | Female | e6a33eee180b07e563d74fee8c2c66b8 |      1 | NULL  |
|       3 | rivera92 | david      | john      | Male   | 1c3a8e03f448d211904161a6f5849b68 |      1 | NULL  |
|       4 | ross95   | maria      | sanders   | Male   | 62f0a68a4179c5cdd997189760cbcf18 |      1 | NULL  |
|       6 | smith34  | daniel     | michael   | Female | 7055b3d9f5cb2829c26cd7e0e601cde5 |      1 | NULL  |
|       9 | brooks80 | morgan     | maria     | Female | aa736a35dc15934d67c0a999dccff8f6 |      1 | NULL  |
|      10 | morgan65 | paul       | miller    | Female | a28dca31f5aa5792e1cefd1dfd098569 |      1 | NULL  |
+---------+----------+------------+-----------+--------+----------------------------------+--------+-------+
6 rows in set (0.001 sec)

MariaDB [concordia]>
```

SQL:
COMMIT & ROLLBACK

SQL: COMMIT & ROLLBACK

The COMMIT command

- ✓ The transactional command used to save changes invoked by a transaction to the database.
- ✓ The syntax for the COMMIT command is as follows: COMMIT;

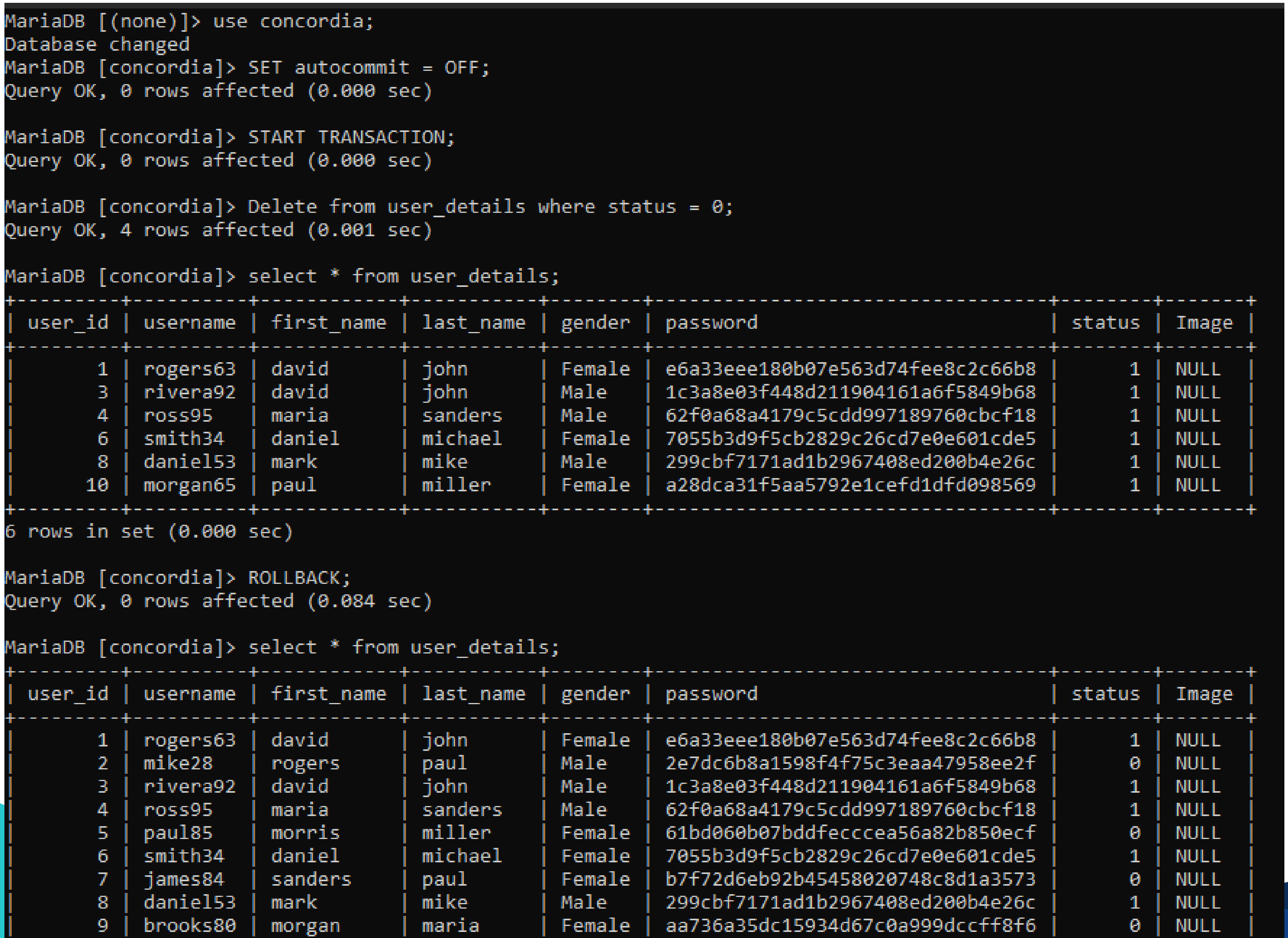
```
mysql> SET autocommit = OFF;
mysql> START TRANSACTION;
mysql> Delete from user_details where status = 0;
mysql> COMMIT;
```

The ROLLBACK Command

- ✓ The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.
- ✓ This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.
- ✓ The syntax for a ROLLBACK command is as follows ROLLBACK;

```
mysql> SET autocommit = OFF;
mysql> START TRANSACTION;
mysql> Delete from user_details where status = 0;
mysql> ROLLBACK;
```

SQL: ROLLBACK



```
MariaDB [(none)]> use concordia;
Database changed
MariaDB [concordia]> SET autocommit = OFF;
Query OK, 0 rows affected (0.000 sec)

MariaDB [concordia]> START TRANSACTION;
Query OK, 0 rows affected (0.000 sec)

MariaDB [concordia]> Delete from user_details where status = 0;
Query OK, 4 rows affected (0.001 sec)

MariaDB [concordia]> select * from user_details;
+---------+----------+------------+-----------+--------+----------------------------------+--------+-------+
| user_id | username | first_name | last_name | gender | password                         | status | Image |
+---------+----------+------------+-----------+--------+----------------------------------+--------+-------+
|       1 | rogers63 | david      | john      | Female | e6a33eee180b07e563d74fee8c2c66b8 |      1 | NULL  |
|       3 | rivera92 | david      | john      | Male   | 1c3a8e03f448d211904161a6f5849b68 |      1 | NULL  |
|       4 | ross95   | maria      | sanders   | Male   | 62f0a68a4179c5cdd997189760cbcf18 |      1 | NULL  |
|       6 | smith34  | daniel     | michael   | Female | 7055b3d9f5cb2829c26cd7e0e601cde5 |      1 | NULL  |
|       8 | daniel53 | mark       | mike      | Male   | 299cbf7171ad1b2967408ed200b4e26c |      1 | NULL  |
|      10 | morgan65 | paul       | miller    | Female | a28dca31f5aa5792e1cefd1dfd098569 |      1 | NULL  |
+---------+----------+------------+-----------+--------+----------------------------------+--------+-------+
6 rows in set (0.000 sec)

MariaDB [concordia]> ROLLBACK;
Query OK, 0 rows affected (0.084 sec)

MariaDB [concordia]> select * from user_details;
+---------+----------+------------+-----------+--------+----------------------------------+--------+-------+
| user_id | username | first_name | last_name | gender | password                         | status | Image |
+---------+----------+------------+-----------+--------+----------------------------------+--------+-------+
|       1 | rogers63 | david      | john      | Female | e6a33eee180b07e563d74fee8c2c66b8 |      1 | NULL  |
|       2 | mike28   | rogers     | paul      | Male   | 2e7dc6b8a1598f4f75c3eaa47958ee2f |      0 | NULL  |
|       3 | rivera92 | david      | john      | Male   | 1c3a8e03f448d211904161a6f5849b68 |      1 | NULL  |
|       4 | ross95   | maria      | sanders   | Male   | 62f0a68a4179c5cdd997189760cbcf18 |      1 | NULL  |
|       5 | paul85   | morris     | miller    | Female | 61bd060b07bddfecccea56a82b850ecf |      0 | NULL  |
|       6 | smith34  | daniel     | michael   | Female | 7055b3d9f5cb2829c26cd7e0e601cde5 |      1 | NULL  |
|       7 | james84  | sanders    | paul      | Female | b7f72d6eb92b45458020748c8d1a3573 |      0 | NULL  |
|       8 | daniel53 | mark       | mike      | Male   | 299cbf7171ad1b2967408ed200b4e26c |      1 | NULL  |
|       9 | brooks80 | morgan     | maria     | Female | aa736a35dc15934d67c0a999dccff8f6 |      0 | NULL  |
```

What is BLOBS (Binary Large Objects)?

BLOBS(Binary Large Objects)

- ✓ A BLOB is a binary large object that can hold a variable amount of data.
- ✓ It Stores any kind of data in binary format such as images, audio, and video.
- ✓ BLOB allocates spaces in Giga Bytes.
- ✓ Some projects require a large string or block of binary data to be stored in a database.
- ✓ For example, a digital file containing a picture, video, or a song can be stored in a database using a BLOB.

HOW to Insert files into MySQL table with BLOB type ?

Insert images into a MySQL table with BLOB type

- ✓ Let’s add an Image column in our user_details table.
- ✓ use alter command to add a column of blob type.

```
mysql> ALTER TABLE user_details ADD COLUMN image blob;
```

What is JDBC (Java Database Connectivity)?

Java Database Connectivity

- ✓ Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access any kind of tabular data, especially relational database.
- ✓ It is part of Java Standard Edition platform, from Oracle Corporation. It acts as a middle layer interface between java applications and database.
- ✓ The JDBC classes are contained in the Java Package **java.sql** and **javax.sql.**
- ✓ JDBC helps you to write Java applications that manage these three programming activities:
    1. Connect to a data source, like a database.
    2. Send queries and update statements to the database
    3. Retrieve and process the results received from the database in answer to your query

JDBC: Configuration

Step1 : Correct tools and configurations.

✓ What We Need to configure and create a demo JAVA JDBC project are as following :

1. IntelliJ IDEA Ultimate
2. JDK 8 (or >8)
3. MySQL 5 (or >5)
4. DBeaver or MySQL Workbench or PhpMyAdmin (Not required but recommended)

JDBC : Database creation

Step 2 : Review DataBase Tables

✓ Create a Database Demo with Employee table in it with some information.

✓ Syntax to create table and insert record in table.

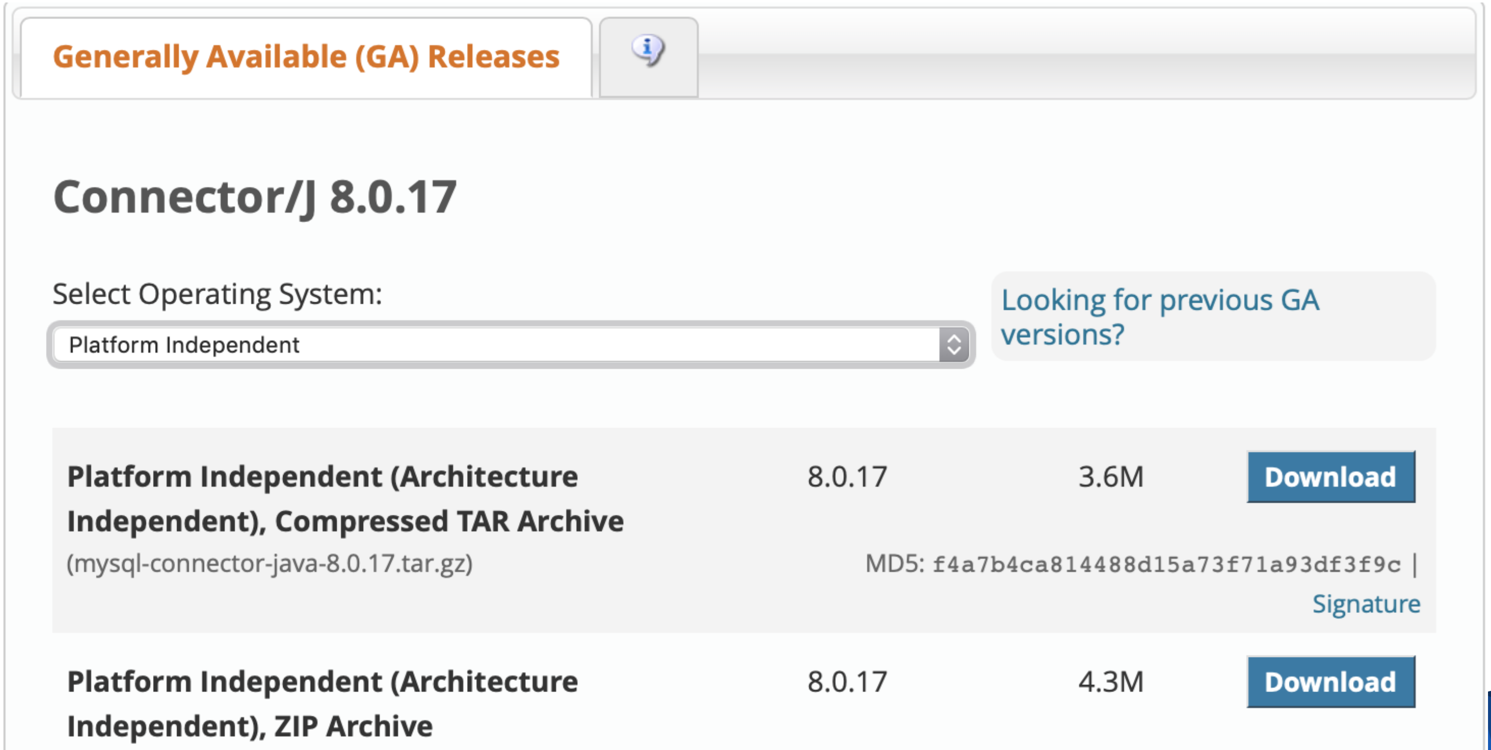
```
CREATE TABLE `user_details` (
  `user_id` int(11) NOT NULL,
  `username` varchar(255) DEFAULT NULL,
  `first_name` varchar(50) DEFAULT NULL,
  `last_name` varchar(50) DEFAULT NULL,
  `gender` varchar(10) DEFAULT NULL,
  `password` varchar(50) DEFAULT NULL,
  `status` tinyint(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `user_details` (`user_id`,
`username`, `first_name`, `last_name`, `gender`,
`password`, `status`) VALUES (1, 'rogers63',
'david', 'john', 'Female',
'e6a33eee180b07e563d74fee8c2c66b8', 1);
```

JDBC : Connectors

Step 3 : Download the JDBC connectors

- ✓ Link : https://dev.mysql.com/downloads/connector/j/
- ✓ In our case we download the platform independent version.



Generally Available (GA) Releases

Connector/J 8.0.17

Select Operating System: Platform Independent

Looking for previous GA versions?

Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-java-8.0.17.tar.gz)	8.0.17	3.6M	Download MD5: `f4a7b4ca814488d15a73f71a93df3f9c` | Signature
Platform Independent (Architecture Independent), ZIP Archive	8.0.17	4.3M	Download

JDBC : Example

Query Execution Best Practices

Query Execution Best Practices

- ✓ Avoid hardcoding server or host address. (Hint: See Configuration)
- ✓ Try with resource statement: The try-with-resources statement is a try statement that declares one or more resources.
- ✓ A resource is an object that must be closed after the program is finished with it. (See here)
- ✓ Avoid using SELECT * always because:
  1. you don't need all the columns
  2. Columns can change
  3. Columns can be added/removed.
- ✓ Protect JDBC application against SQL Injection (See article)

What is SQL output parameters in stored procedure ?

Stored procedure

What is Exercise Query Execution using JDBC ??

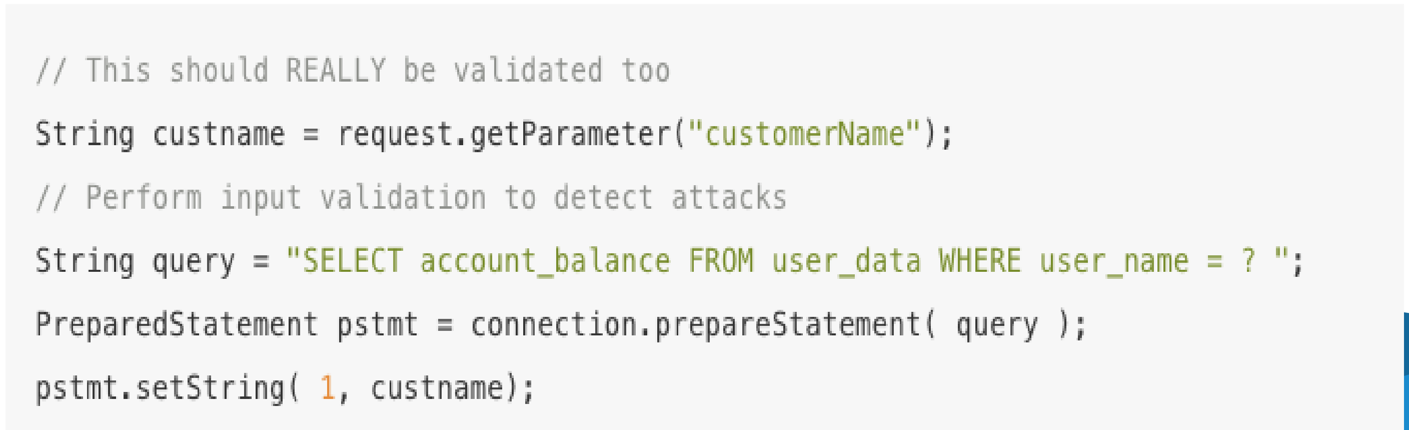
How to prevent SQL injection?

How to prevent SQL injection?

✓ Primary Defenses:

- Use of Prepared Statements (with Parameterized Queries)
- Use of Stored Procedures
- Escaping All User Supplied Input

✓ The following code example uses a PreparedStatement, Java's implementation of a parameterized query, to execute the same database query.



```
// This should REALLY be validated too
String custname = request.getParameter("customerName");
// Perform input validation to detect attacks
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
```

```
String custname = request.getParameter(“customerName”);

String query = “SELECT account_balance FROM user_data WHERE user_name = ?”;

PreparedStatement pstmt = connecton.prepareStatement ( query );

Pstmt.setString( 1 , custname );

ResultSet results  = pstmt.executeQuery ( );
```

References

- ✓ Install MySQL on your system
  https://overiq.com/installing-mysql-windows-linux-and-mac
- ✓ MySQL Tutorials
  https://www.javatpoint.com/mysql-tutorial
- ✓ JDBC Tutorials
  https://www.tutorialspoint.com/jdbc/index.htm
- ✓ Java Servlet with JDBC Example
  https://www.geeksforgeeks.org/java-servlet-and-jdbc-example-insert-data-in-mysql
- ✓ DBeaver Community Universal Database Tool
  https://dbeaver.io
- ✓ Download Database which is used in this tutorial
  https://gist.github.com/vasuratanpara/d0e49b410337868516388fffec968341

Thanks

Any questions?