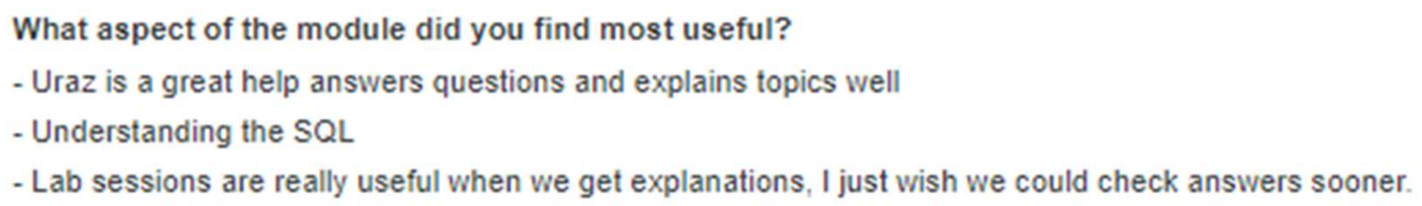
SCC201: DATABASES

Week 6: JDBC and Revision

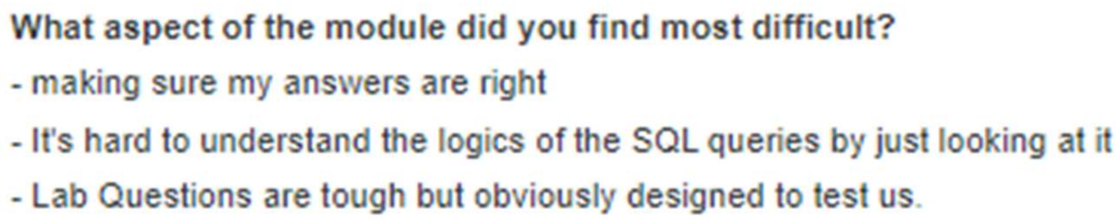
Week 1	Lecture 1	Introduction to the module, Why do we need Databases? Entity Relationship Model
	Lecture 2	Entity Relationship Model (ERM)
	Lab	**NO LAB**
Week2	Lecture 1	Relational Model (RM)
	Lecture 2	ER to RM
	Lab	**ER diagrams.**
Week 3	Lecture 1	Functional Dependencies
	Lecture 2	1st, 2nd, 3rd and Boyce-Cott Normal Forms
	Lab	**ER to Relational Model.**
Week 4	Lecture 1	Relational Algebra
	Lecture 2	Relational Algebra
	Lab	**Functional dependencies and Normal forms**
Week 5	Lecture 1	SQL Scripts
	Lecture 2	Advanced SQL Scripts
	Lab	**Relational algebra**
Week 6	Lecture 1	JDBC
	Lecture 2	Review
	Lab	**Advanced SQL Scripting**
Week 7	Lecture 1	Physical Storage - record files
	Lecture 2	Storage - secondary files
	Lab	Project
Week 8	Lecture 1	Record Search - B-Trees
	Lecture 2	Search - Hashing
	Lab	Project
Week 9	Lecture 1	Concurrency - Transaction Processing (cont)
	Lecture 2	Durability of Transactions and Crash Recovery
	Lab	Project
Week 10	Lecture 1	Advanced SQL - schemas, views, access control
	Lecture 2	Review
	Lab	Project

Lancaster University

From you

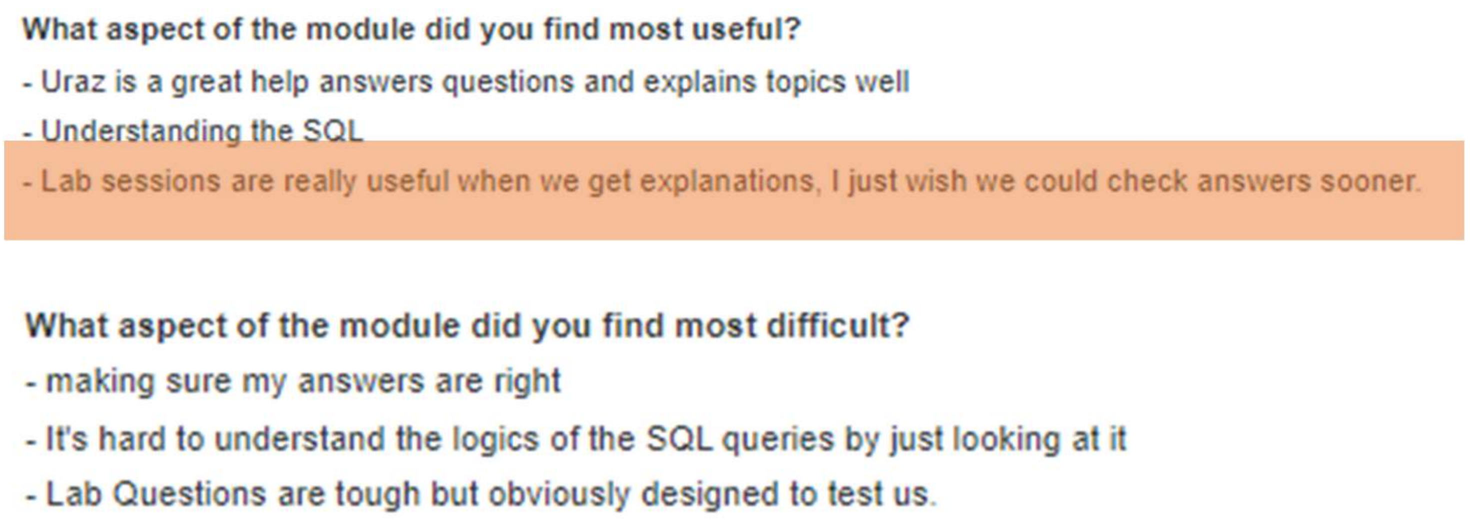

What aspect of the module did you find most useful?

- Uraz is a great help answers questions and explains topics well
- Understanding the SQL
- Lab sessions are really useful when we get explanations, I just wish we could check answers sooner.


What aspect of the module did you find most difficult?

- making sure my answers are right
- It's hard to understand the logics of the SQL queries by just looking at it
- Lab Questions are tough but obviously designed to test us.

From you



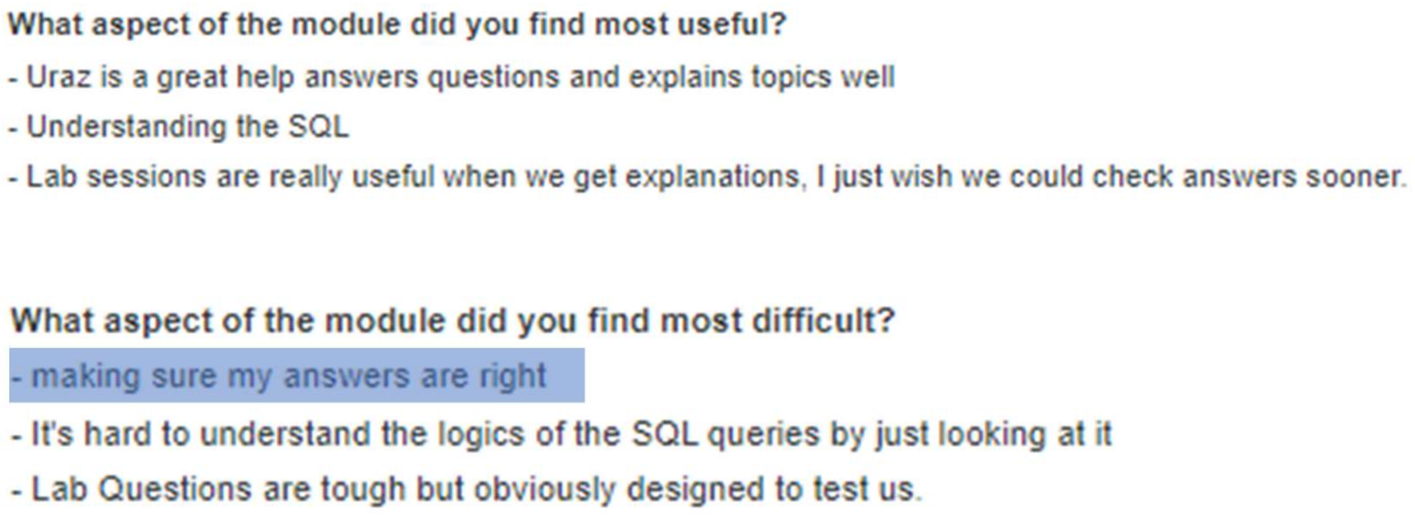
What aspect of the module did you find most useful?

- Uraz is a great help answers questions and explains topics well
- Understanding the SQL
- Lab sessions are really useful when we get explanations, I just wish we could check answers sooner.

What aspect of the module did you find most difficult?

- making sure my answers are right
- It's hard to understand the logics of the SQL queries by just looking at it
- Lab Questions are tough but obviously designed to test us.

From you



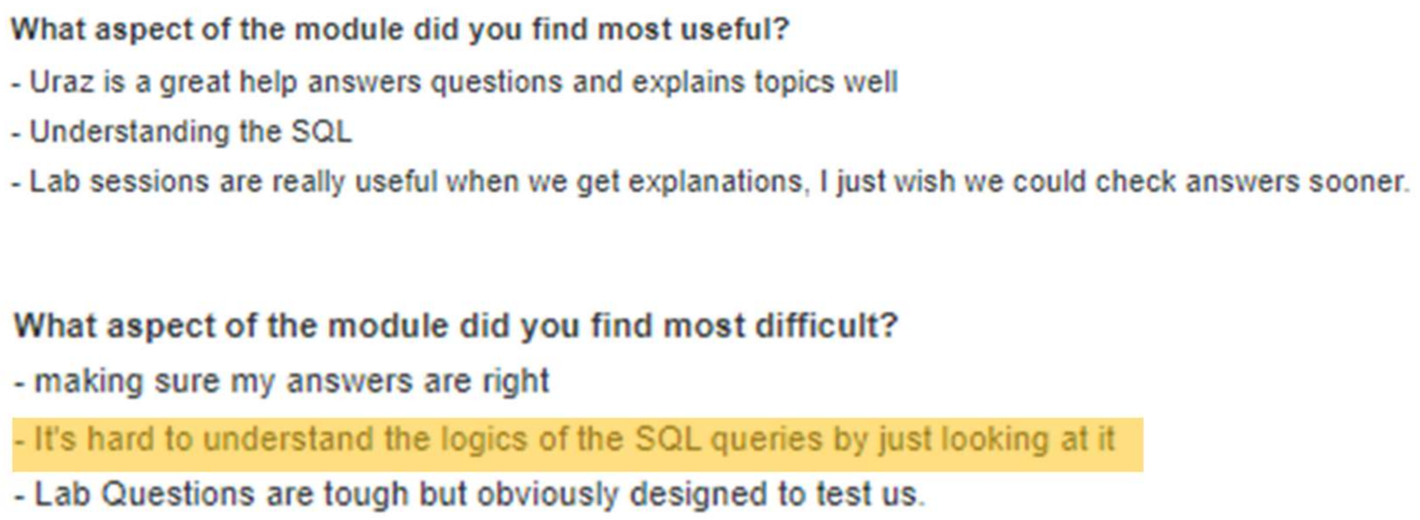
What aspect of the module did you find most useful?

- Uraz is a great help answers questions and explains topics well
- Understanding the SQL
- Lab sessions are really useful when we get explanations, I just wish we could check answers sooner.

What aspect of the module did you find most difficult?

- making sure my answers are right
- It's hard to understand the logics of the SQL queries by just looking at it
- Lab Questions are tough but obviously designed to test us.

From you



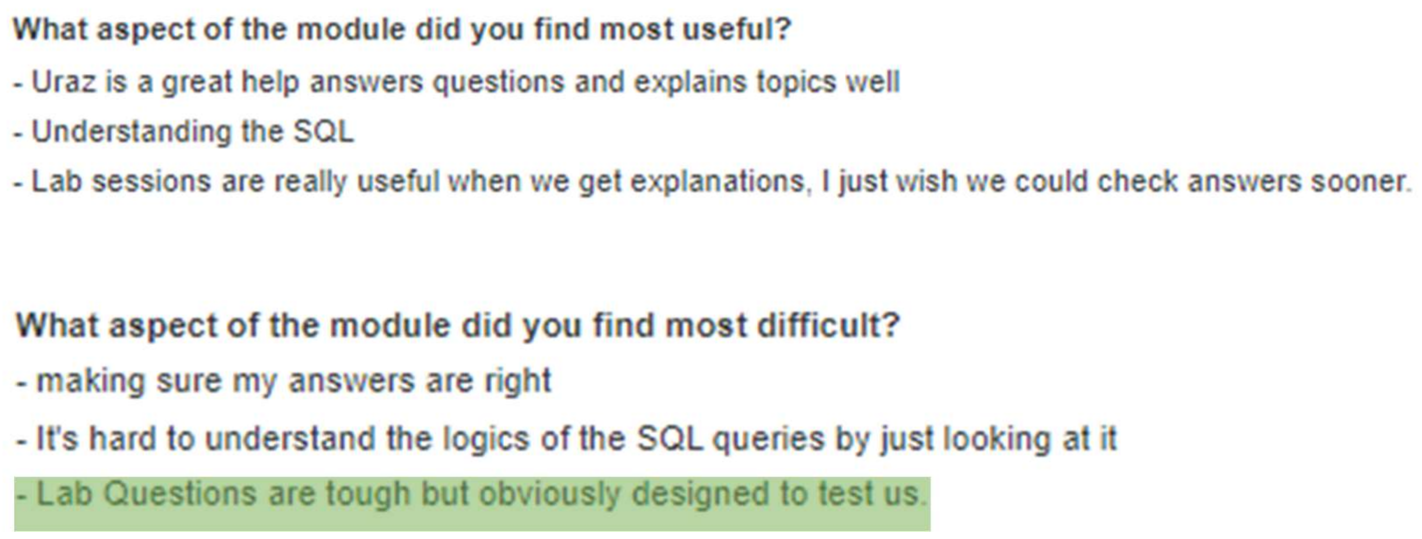
What aspect of the module did you find most useful?

- Uraz is a great help answers questions and explains topics well
- Understanding the SQL
- Lab sessions are really useful when we get explanations, I just wish we could check answers sooner.

What aspect of the module did you find most difficult?

- making sure my answers are right
- It's hard to understand the logics of the SQL queries by just looking at it
- Lab Questions are tough but obviously designed to test us.

From you



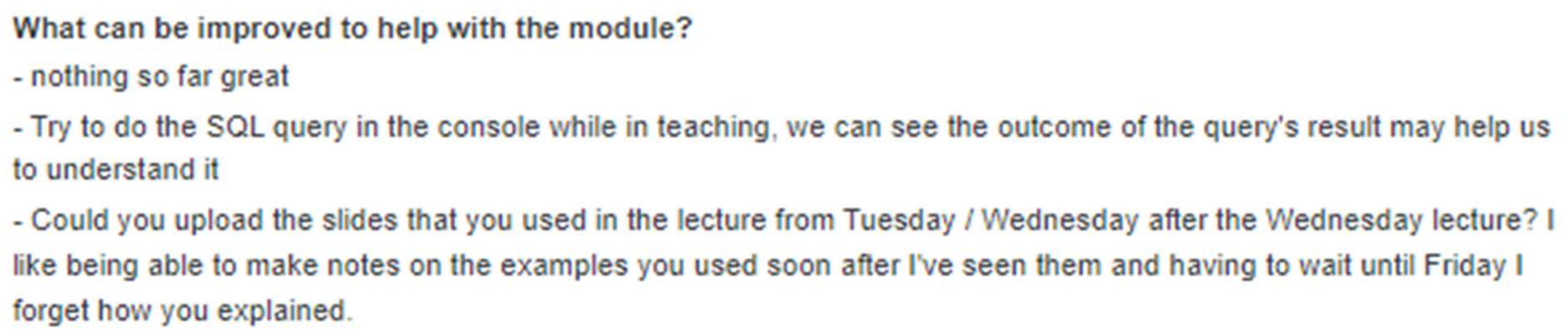
What aspect of the module did you find most useful?

- Uraz is a great help answers questions and explains topics well
- Understanding the SQL
- Lab sessions are really useful when we get explanations, I just wish we could check answers sooner.

What aspect of the module did you find most difficult?

- making sure my answers are right
- It's hard to understand the logics of the SQL queries by just looking at it
- Lab Questions are tough but obviously designed to test us.

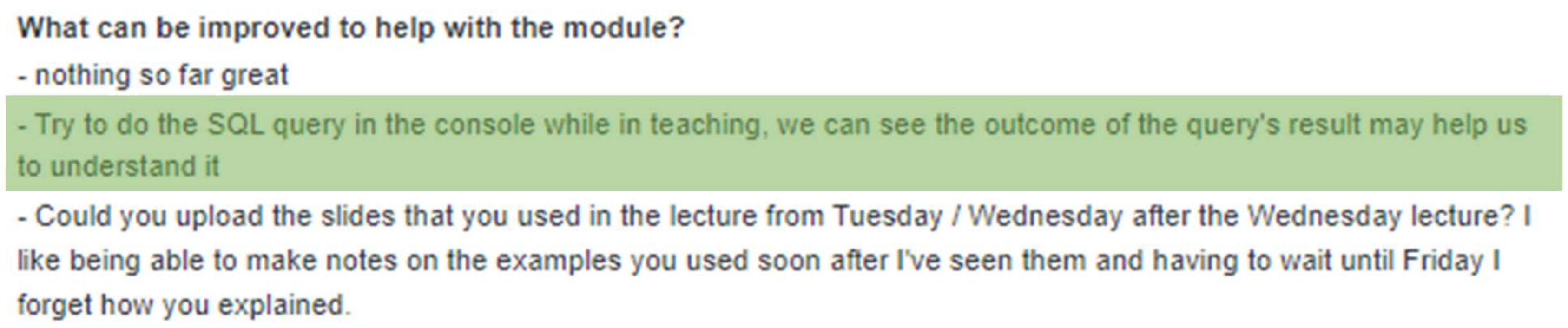
FROM YOU



What can be improved to help with the module?

- nothing so far great
- Try to do the SQL query in the console while in teaching, we can see the outcome of the query's result may help us to understand it
- Could you upload the slides that you used in the lecture from Tuesday / Wednesday after the Wednesday lecture? I like being able to make notes on the examples you used soon after I've seen them and having to wait until Friday I forget how you explained.

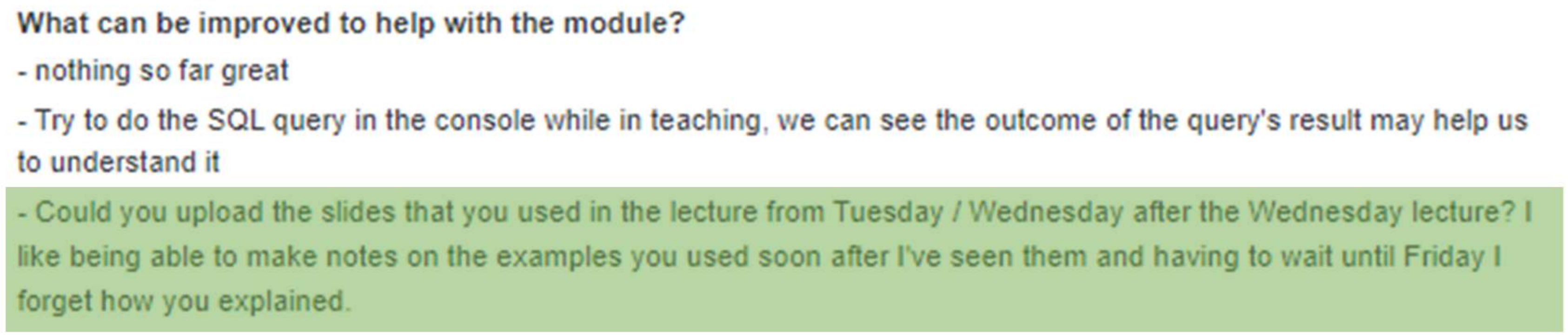
FROM YOU



What can be improved to help with the module?

- nothing so far great
- Try to do the SQL query in the console while in teaching, we can see the outcome of the query's result may help us to understand it
- Could you upload the slides that you used in the lecture from Tuesday / Wednesday after the Wednesday lecture? I like being able to make notes on the examples you used soon after I've seen them and having to wait until Friday I forget how you explained.

FROM YOU



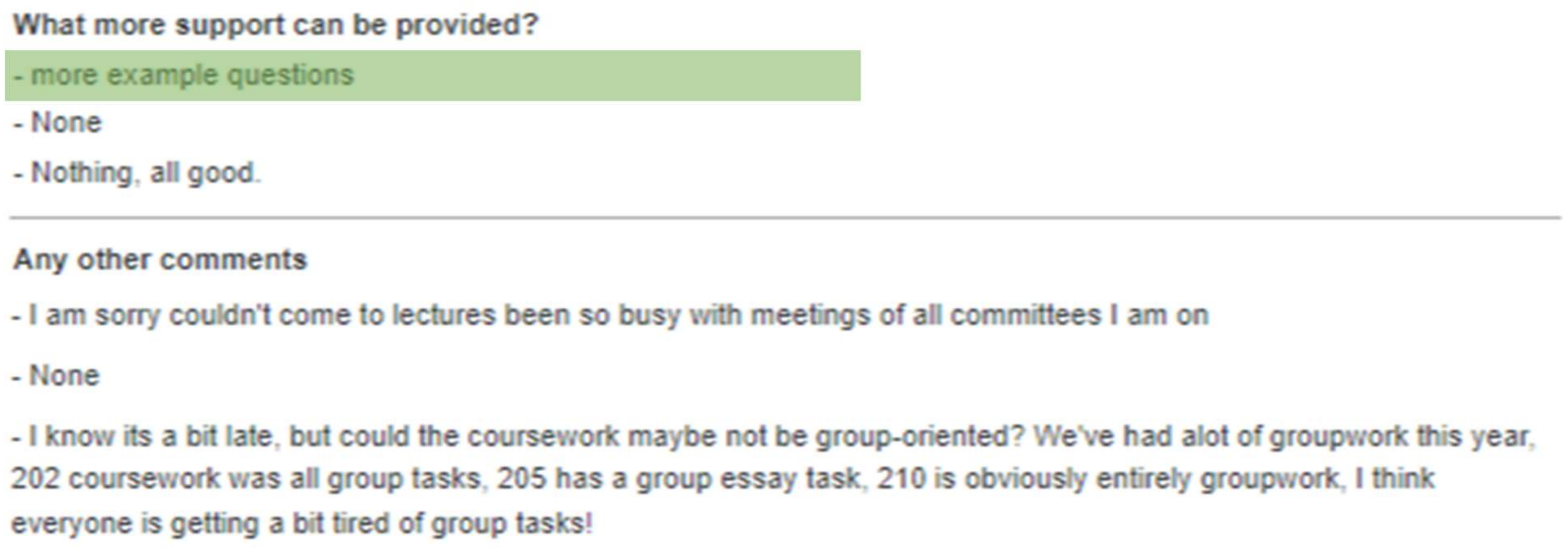
What can be improved to help with the module?

- nothing so far great

- Try to do the SQL query in the console while in teaching, we can see the outcome of the query's result may help us to understand it

- Could you upload the slides that you used in the lecture from Tuesday / Wednesday after the Wednesday lecture? I like being able to make notes on the examples you used soon after I've seen them and having to wait until Friday I forget how you explained.

From you



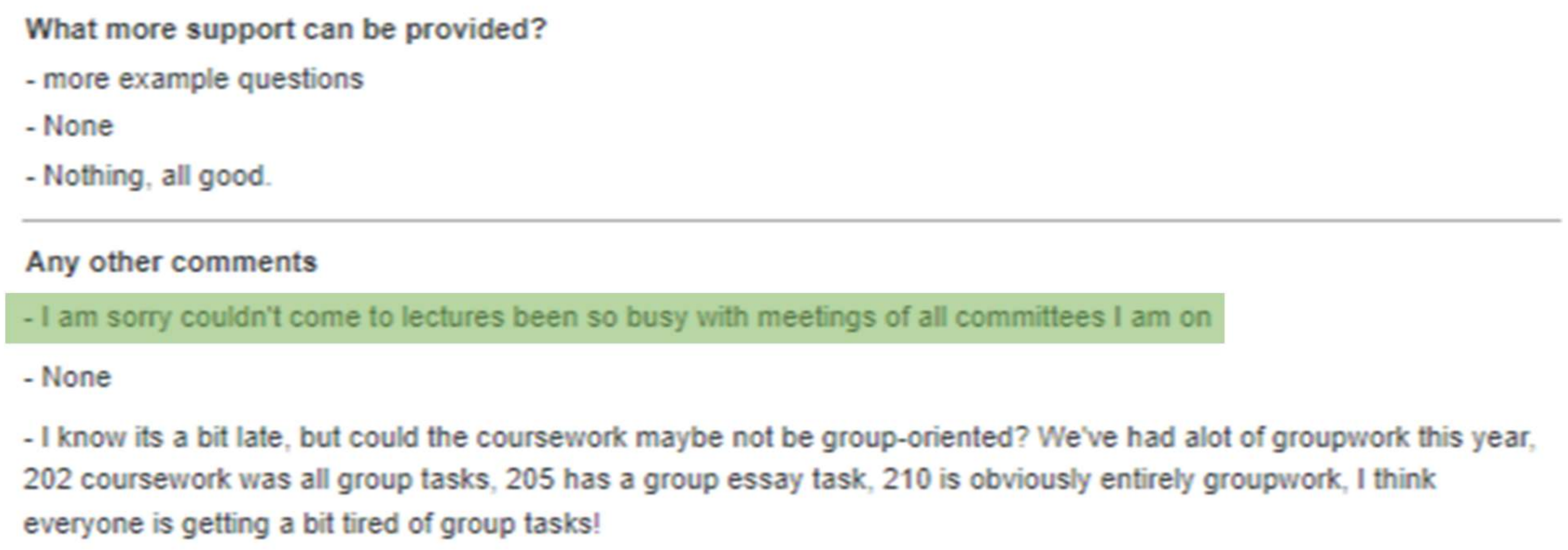
What more support can be provided?

- more example questions
- None
- Nothing, all good.

Any other comments

- I am sorry couldn't come to lectures been so busy with meetings of all committees I am on
- None
- I know its a bit late, but could the coursework maybe not be group-oriented? We've had alot of groupwork this year, 202 coursework was all group tasks, 205 has a group essay task, 210 is obviously entirely groupwork, I think everyone is getting a bit tired of group tasks!

From you



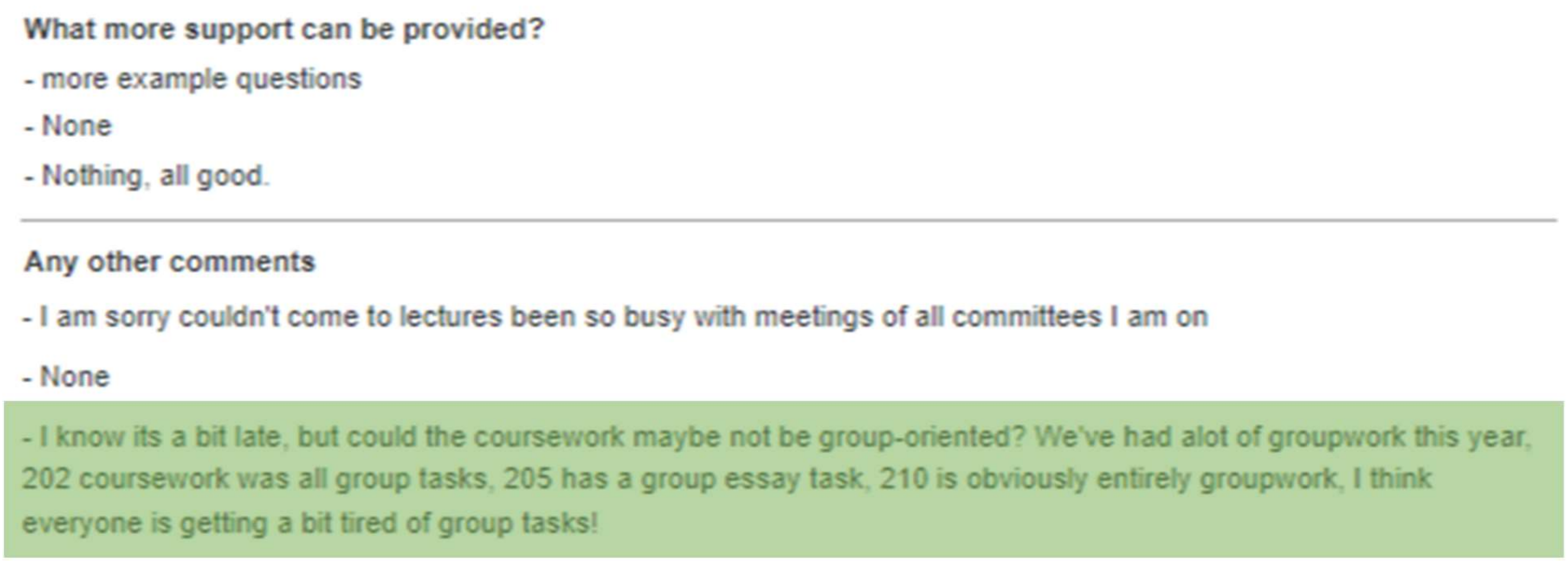
What more support can be provided?

- more example questions
- None
- Nothing, all good.

Any other comments

- I am sorry couldn't come to lectures been so busy with meetings of all committees I am on
- None
- I know its a bit late, but could the coursework maybe not be group-oriented? We've had alot of groupwork this year, 202 coursework was all group tasks, 205 has a group essay task, 210 is obviously entirely groupwork, I think everyone is getting a bit tired of group tasks!

From you



What more support can be provided?

- more example questions
- None
- Nothing, all good.

Any other comments

- I am sorry couldn't come to lectures been so busy with meetings of all committees I am on
- None
- I know its a bit late, but could the coursework maybe not be group-oriented? We've had alot of groupwork this year, 202 coursework was all group tasks, 205 has a group essay task, 210 is obviously entirely groupwork, I think everyone is getting a bit tired of group tasks!

Week6

- Reference Reading MySQL book and
- https://docs.oracle.com/javase/tutorial/jdbc/index.html

Key people

- DBMS Implementer.
- DB Designer.
-
-

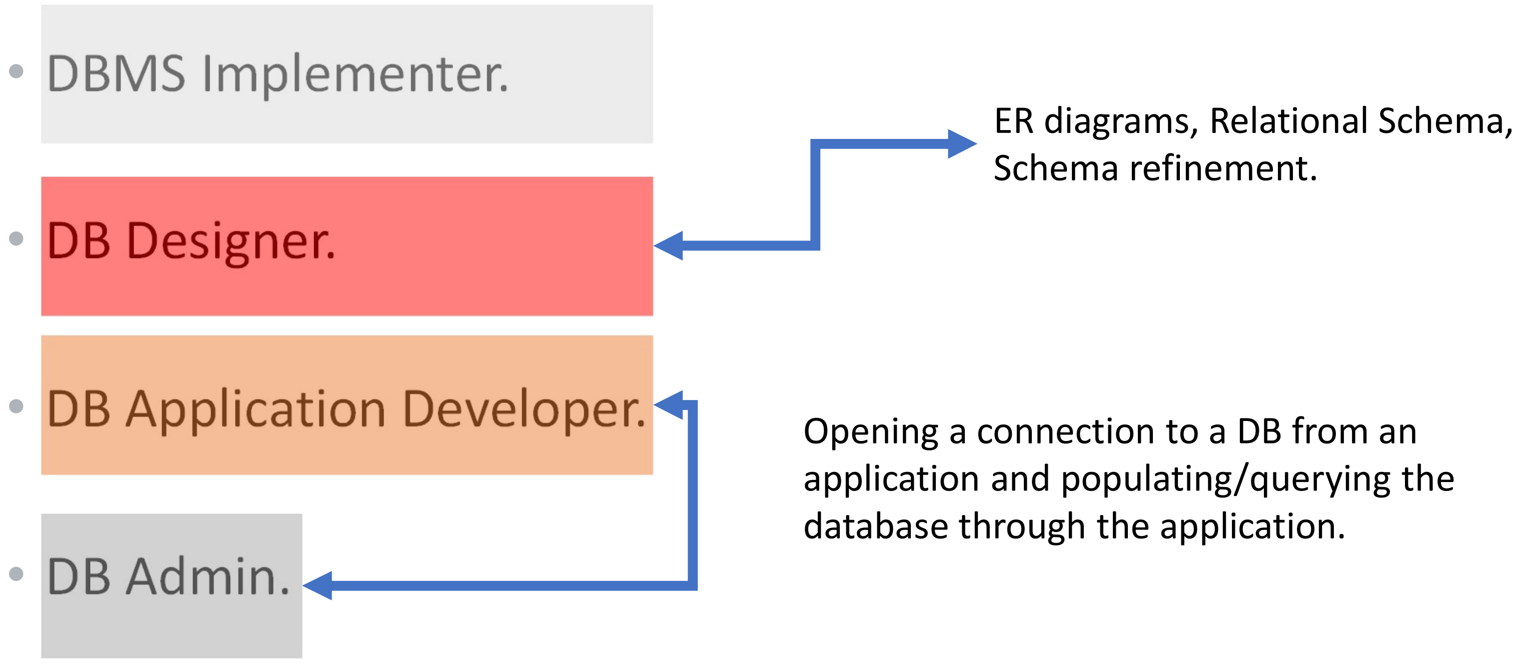
ER diagrams, Relational Schema, Schema refinement.

Key people

- DBMS Implementer.
- DB Designer.
- DB Application Developer.
- DB Admin.

ER diagrams, Relational Schema,
Schema refinement.

Key people



- DBMS Implementer.
- DB Designer.
- DB Application Developer.
- DB Admin.

ER diagrams, Relational Schema, Schema refinement.

Opening a connection to a DB from an application and populating/querying the database through the application.

Inserting tuples

- How?
-
    -
-
    -

Inserting tuples

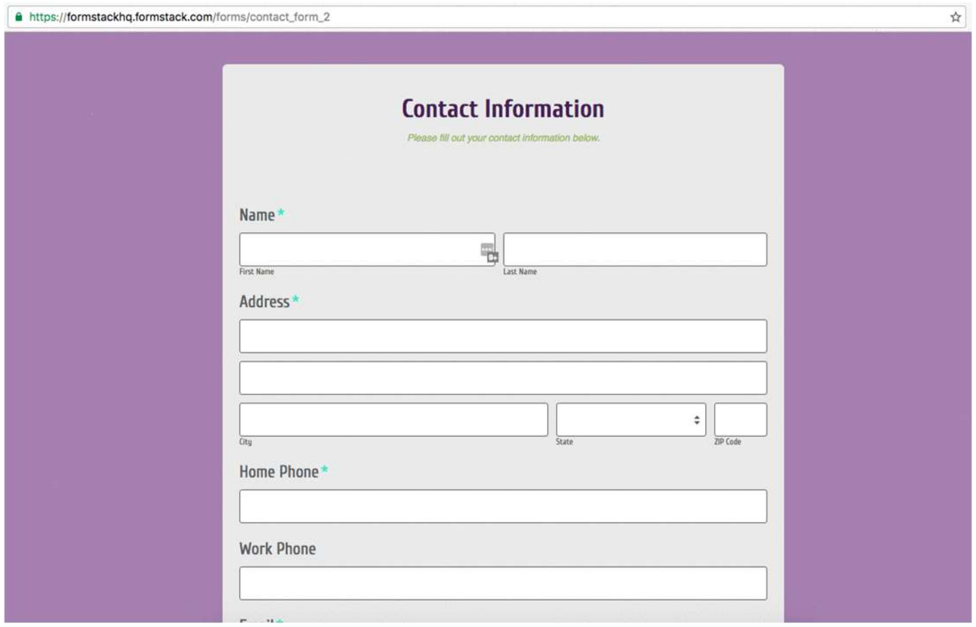
- How?
-
    -
-
    -

Inserting tuples

- How?
- OK, but shouldn’t we be able to insert values through the application’s interface?

Inserting tuples

- How?
- OK, but shouldn't we be able to insert values through the application's interface?
  - Yes, of course.



https://formstackhq.formstack.com/forms/contact_form_2

Contact Information

Please fill out your contact information below.

Name *

First Name

Last Name

Address *

City

State

ZIP Code

Home Phone *

Work Phone

Inserting tuples

- How?
- OK, but shouldn't we be able to insert values through the application's interface?
  - Yes, of course.
- How?
  -

Inserting tuples

- How?
- OK, but shouldn't we be able to insert values through the application's interface?
  - Yes, of course.
- How?
  - By opening a connection (Portal) from our application to DB.

Java Database Connectivity Framework

- Java applications use JDBC framework to connect with a database.
  - There are many database management systems
    - Oracle DBMS,
    - Microsoft Access,
    - MySQL,
    - SQLite,
    - …
- To encapsulate this variety, JDBC introduced a set of protocols (Drivers).

What is JDBC ?

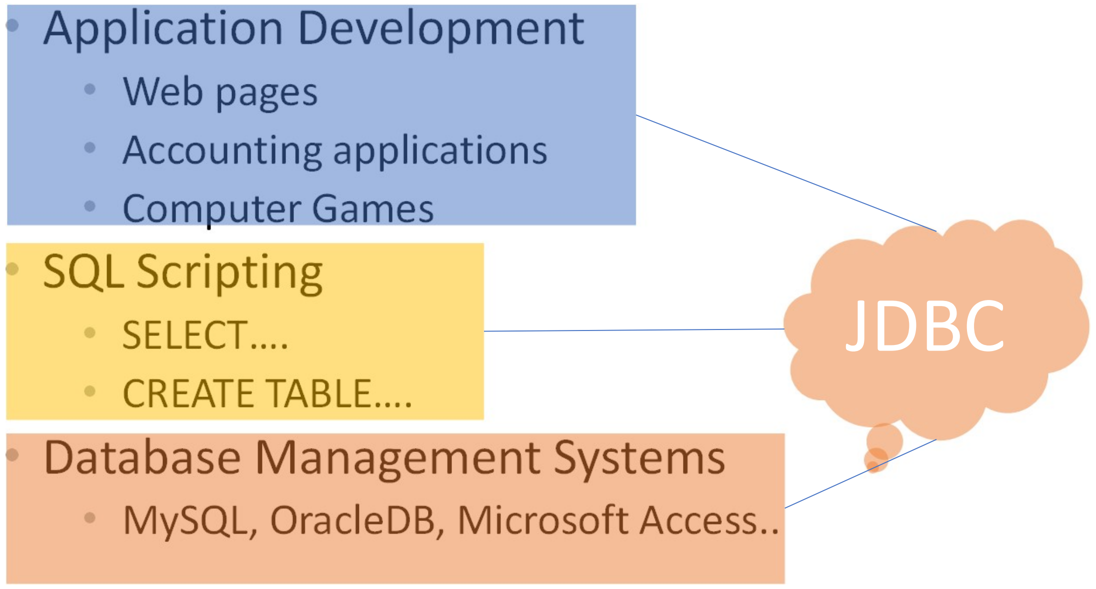
- We have Three Worlds
  - Application Development
    - Web pages
    - Accounting applications
    - Computer Games
  - SQL Scripting
    - SELECT….
    - CREATE TABLE….
  - Database Management Systems
    - MySQL, OracleDB, Microsoft Access..

What is JDBC ?

- We have Three Worlds
  - Application Development
    - Web pages
    - Accounting applications
    - Computer Games
  - SQL Scripting
    - SELECT….
    - CREATE TABLE….
  - Database Management Systems
    - MySQL, OracleDB, Microsoft Access..

What is JDBC ?

- We have Three Worlds



  - Application Development
    - Web pages
    - Accounting applications
    - Computer Games
  - SQL Scripting
    - SELECT….
    - CREATE TABLE….
  - Database Management Systems
    - MySQL, OracleDB, Microsoft Access..

What is JDBC ?

- Java database connectivity
    - Developed in 1996

What is JDBC ?

- Java database connectivity
  - Developed in 1996

JDBC is often thought to stand for "Java Database Connectivity", but in fact, JDBC is a trademark name, not an acronym

What is JDBC ?

- Java database connectivity
  - Developed in 1996
- Its tasks:
  - Establish connection
  - Send SQL statements
  - Process the results

JDBC is often thought to stand for "Java Database Connectivity", but in fact, JDBC is a trademark name, not an acronym

General JDBC Structure

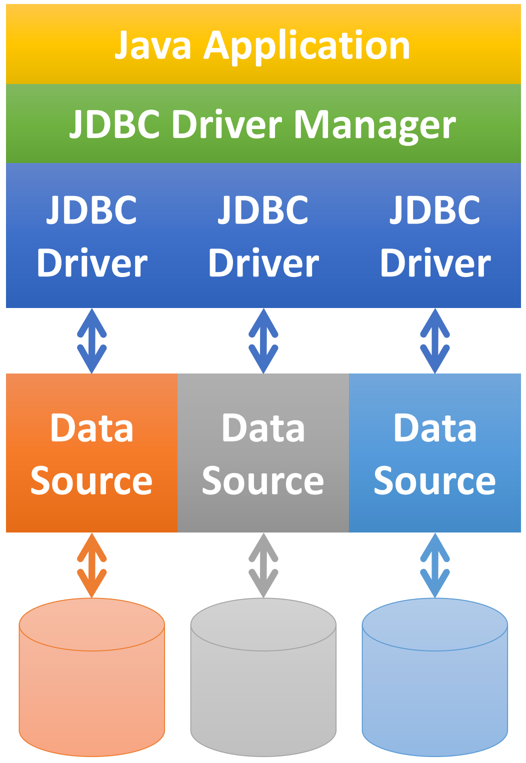
- JDBC API consists of two main interfaces :
  - An API for application writers (for you!)
  -
    -
    -
    -

General JDBC Structure

- JDBC API consists of two main interfaces :
  - An API for application writers (for you!)
  - A lower-level API for driver writers, example drivers being…
    - SQLite driver
    - MySQL driver
    - Postgres
      etc.

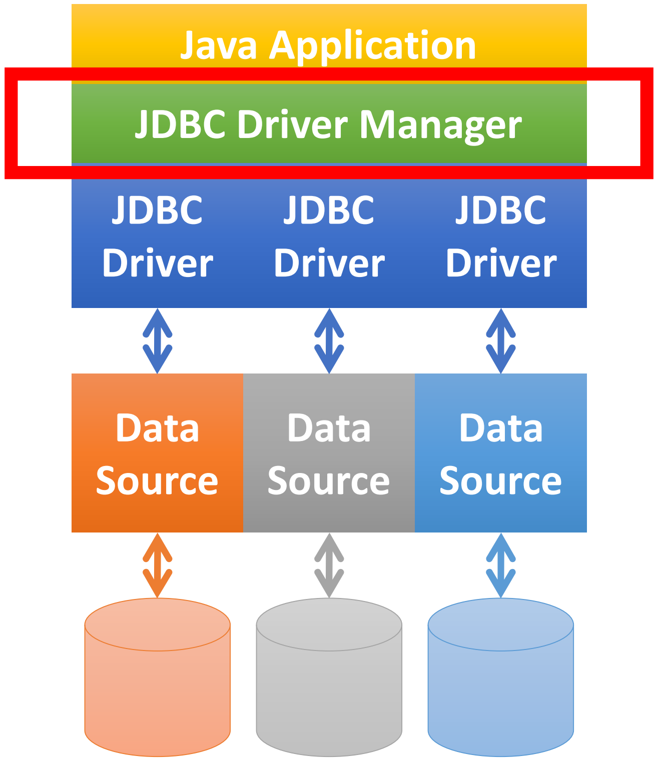
General JDBC Structure

- JDBC API consists of two main interfaces :
  - An API for application writers (for you!)
  - A lower-level API for driver writers, example drivers being…
    - SQLite driver
    - MySQL driver
    - Postgres
    etc.



General JDBC Structure

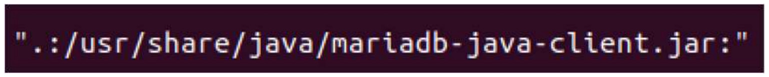
- JDBC API consists of two main interfaces :
  - An API for application writers (for you!)
  - A lower-level API for driver writers, example drivers being…
    - SQLite driver
    - MySQL driver
    - Postgres
    etc.



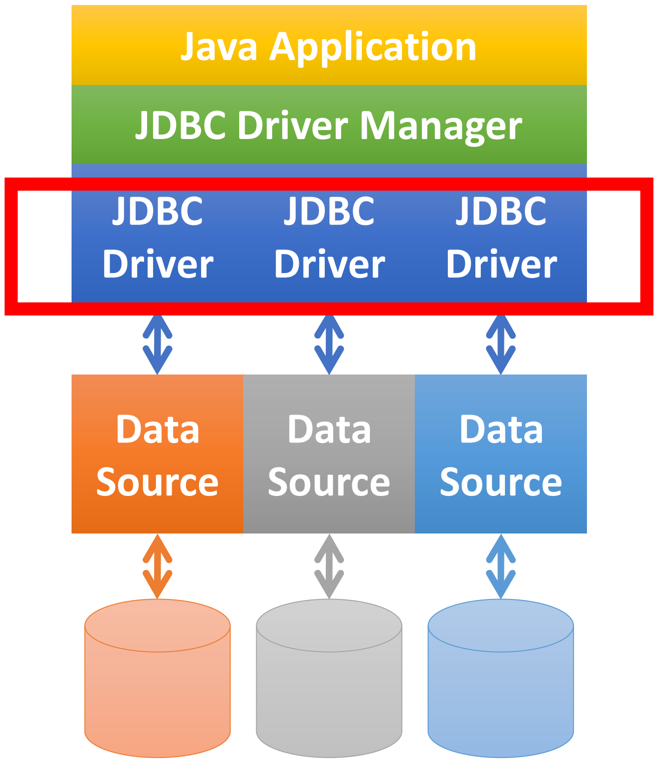
https://www.ibm.com/docs/en/i/7.1?topic=connections-java-drivermanager-class

General JDBC Structure

- JDBC API consists of two main interfaces :
  - An API for application writers (for you!)
  - A lower-level API for driver writers, example drivers being…
    - SQLite driver
    - MySQL driver
    - Postgres
    etc.



```
".:/usr/share/java/mariadb-java-client.jar:"
```



Types of JDBC Drivers

- **Type 1** drivers are "bridge" drivers. They use another technology, such as Open Database Connectivity (ODBC), to communicate with a database. This is an advantage because ODBC drivers exist for many Relational Database Management System (RDBMS) platforms. The Java Native Interface (JNI) calls ODBC functions from the JDBC driver.
- A Type 1 driver must have the bridge driver installed and configured before JDBC can be used. This can be a severe drawback for a production application. Type 1 drivers cannot be used in an applet since applets cannot load native code.
- **Type 2** drivers use a native API to communicate with a database system. Java native methods invoke the API functions that perform database operations. Type 2 drivers are generally faster than Type 1 drivers.
- Type 2 drivers need native binary code installed and configured to work. A Type 2 driver also uses the JNI. You cannot use a Type 2 driver in an applet since applets cannot load native code. A Type 2 JDBC driver may require installing some DBMS networking software.
- **Type 3** drivers use a networking protocol and middleware to communicate with a server. The server then translates the protocol to DBMS function calls specific to DBMS.
- Type 3 JDBC drivers are the most flexible JDBC solution because they do not require any native binary code on the client. A Type 3 driver does not need any client installation.
- **A Type 4** driver implements a DBMS vendor networking using Java. Since the protocols are usually proprietary, DBMS vendors are generally the only companies providing a Type 4 JDBC driver.
- Type 4 drivers are all Java drivers. This means that there is no client installation or configuration. However, a Type 4 driver may not be suitable for some applications if the underlying protocol does not handle security and network connectivity issues well.

Types of JDBC Drivers

- **Type 1** drivers are "bridge" drivers. They use another technology, such as Open Database Connectivity (ODBC), to communicate with a database. This is an advantage because ODBC drivers exist for many Relational Database Management System (RDBMS) platforms. The Java Native Interface (JNI) calls ODBC functions from the JDBC driver.
- A Type 1 driver must have the bridge driver installed and configured before JDBC can be used. This can be a severe drawback for a production application. Type 1 drivers cannot be used in an applet since applets cannot load native code.
- **Type 2** drivers use a native API to communicate with a database system. Java native methods invoke the API functions that perform database operations. Type 2 drivers are generally faster than Type 1 drivers.
- Type 2 drivers need native binary code installed and configured to work. A Type 2 driver also uses the JNI. You cannot use a Type 2 driver in an applet since applets cannot load native code. A Type 2 JDBC driver may require installing some DBMS networking software.
- **Type 3** drivers use a networking protocol and middleware to communicate with a server. The server then translates the protocol to DBMS function calls specific to DBMS.
- Type 3 JDBC drivers are the most flexible JDBC solution because they do not require any native binary code on the client. A Type 3 driver does not need any client installation.
- **A Type 4** driver implements a DBMS vendor protocol using Java. Since the protocols are usually proprietary, DBMS vendors are generally the only companies providing a Type 4 JDBC driver.
- Type 4 drivers are all Java drivers. This means that there is no client installation or configuration. However, a Type 4 driver may not be suitable for some applications if the underlying protocol does not handle security and network connectivity issues well.

Types of JDBC Drivers

- **Type 1** drivers are "bridge" drivers. They use another technology, such as Open Database Connectivity (ODBC), to communicate with a database. This is an advantage because ODBC drivers exist for many Relational Database Management System (RDBMS) platforms. The Java Native Interface (JNI) calls ODBC functions from the JDBC driver.
- A Type 1 driver must have the bridge driver installed and configured before JDBC can be used. This can be a severe drawback for a production application. Type 1 drivers cannot be used in an applet since applets cannot load native code.
- **Type 2** drivers use a native API to communicate with a database system. Java native methods invoke the API functions that perform database operations. Type 2 drivers are generally faster than Type 1 drivers.
- Type 2 drivers need native binary code installed and configured to work. A Type 2 driver also uses the JNI. You cannot use a Type 2 driver in an applet since applets cannot load native code. A Type 2 JDBC driver may require installing some DBMS networking software.
- **Type 3** drivers use a networking protocol and middleware to communicate with a server. The server then translates the protocol to DBMS function calls specific to DBMS.
- Type 3 JDBC drivers are the most flexible JDBC solution because they do not require any native binary code on the client. A Type 3 driver does not need any client installation.
- **A Type 4** driver implements a DBMS vendor protocol using Java. Since the protocols are usually proprietary, DBMS vendors are generally the only companies providing a Type 4 JDBC driver.
- Type 4 drivers are all Java drivers. This means that there is no client installation or configuration. However, a Type 4 driver may not be suitable for some applications if the underlying protocol does not handle security and network connectivity issues well.

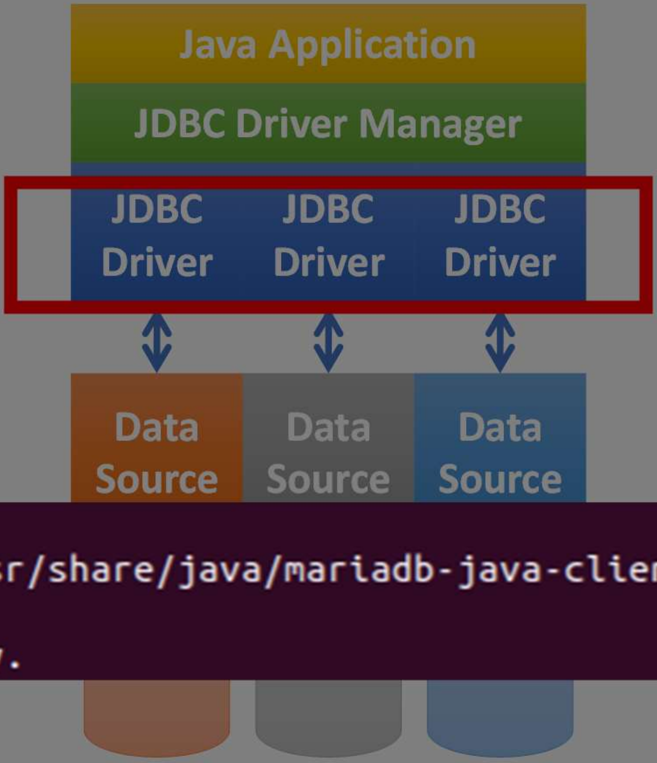
Types of JDBC Drivers

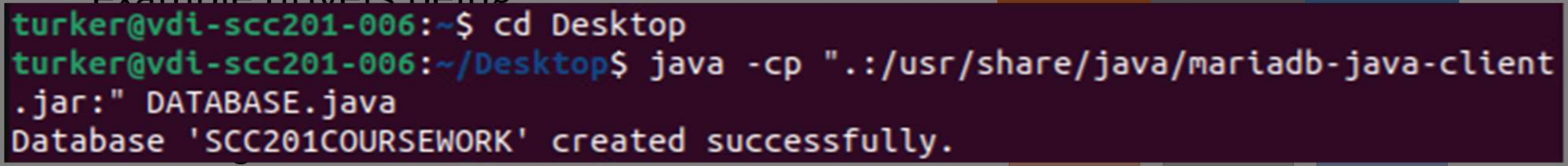
- **Type 1** drivers are "bridge" drivers. They use another technology, such as Open Database Connectivity (ODBC), to communicate with a database. This is an advantage because ODBC drivers exist for many Relational Database Management System (RDBMS) platforms. The Java Native Interface (JNI) calls ODBC functions from the JDBC driver.
- A Type 1 driver must have the bridge driver installed and configured before JDBC can be used. This can be a severe drawback for a production application. Type 1 drivers cannot be used in an applet since applets cannot load native code.
- **Type 2** drivers use a native API to communicate with a database system. Java native methods invoke the API functions that perform database operations. Type 2 drivers are generally faster than Type 1 drivers.
- Type 2 drivers need native binary code installed and configured to work. A Type 2 driver also uses the JNI. You cannot use a Type 2 driver in an applet since applets cannot load native code. A Type 2 JDBC driver may require installing some DBMS networking software.
- **Type 3** drivers use a networking protocol and middleware to communicate with a server. The server then translates the protocol to DBMS function calls specific to DBMS.
- Type 3 JDBC drivers are the most flexible JDBC solution because they do not require any native binary code on the client. A Type 3 driver does not need any client installation.
- **A Type 4** driver implements a DBMS vendor protocol using Java. Since the protocols are usually proprietary, DBMS vendors are generally the only companies providing a Type 4 JDBC driver.
- Type 4 drivers are all Java drivers. This means that there is no client installation or configuration. However, a Type 4 driver may not be suitable for some applications if the underlying protocol does not handle security and network connectivity issues well.

General JDBC Structure

- JDBC API consists of two main interfaces :
  - An API for application writers (for you!)
  - A lower-level API for driver writers, example drivers being

etc.




```
turker@vdi-scc201-006:~$ cd Desktop
turker@vdi-scc201-006:~/Desktop$ java -cp ".:/usr/share/java/mariadb-java-client
.jar:" DATABASE.java
Database 'SCC201COURSEWORK' created successfully.
```

JDBC API: interfaces, classes and exceptions

- Now look at some of these classes and interfaces in more detail
- See pp 1004-1010, C&B 5th Ed.
- JDBC API available in packages *java.sql* and *javax.sql*

Classes
DriverManager
SQLException
SQLWarning

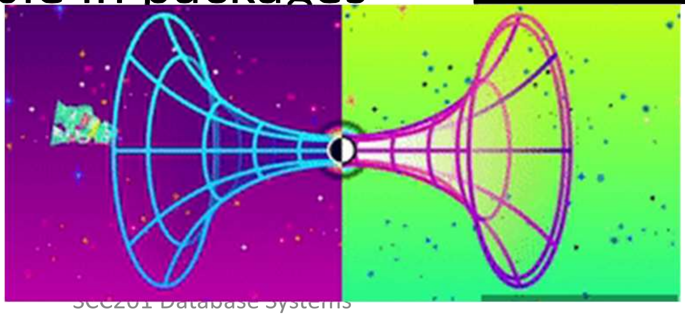
Interfaces
Connection
Statement
PreparedStatement
CallableStatement
ResultSet
DatabaseMetaData
ResultSetMetaData

JDBC API: interfaces, classes and exceptions

- Now look at some of these classes and interfaces in more detail
- See pp 1004-1010, C&B 5th Ed.
- JDBC API available in packages *java.sql* and *javax.sql*

Classes
DriverManager
SQLException
SQLWarning

Interfaces
Connection
Statement
PreparedStatement
CallableStatement
ResultSet
DatabaseMetaData
ResultSetMetaData



Driver Manager

- **Driver Registration:** A Java program must register the appropriate database driver using DriverManager before establishing a database connection.
  - Nowadays, it is done on the fly while connection establishment.
- **Connection Establishment:** Once the driver is registered, DriverManager can be used to obtain a database connection. The getConnection() method of DriverManager is responsible for creating **a connection** to the database.
- **Driver Discovery:** DriverManager uses the registered drivers to find the appropriate driver to handle the connection request. It iterates through the list of registered drivers and delegates the responsibility to the first driver that claims to accept the connection URL.
- **Connection Pooling:** DriverManager is also involved in connection pooling, where it can manage a pool of database connections, helping to improve performance and efficiency by reusing existing connections rather than creating new ones for each database operation.

Driver Manager

- **Driver Registration:** A Java program must register the appropriate database driver using DriverManager before establishing a database connection.
  - Nowadays, it is done on the fly while connection establishment.
- **Connection Establishment:** Once the driver is registered, DriverManager can be used to obtain a database connection. The getConnection() method of DriverManager is responsible for creating **a connection** to the database.
- **Driver Discovery:** DriverManager uses the registered drivers to find the appropriate driver to handle the connection request. It iterates through the list of registered drivers and delegates the responsibility to the first driver that claims to accept the connection URL.
- **Connection Pooling:** DriverManager is also involved in connection pooling, where it can manage a pool of database connections, helping to improve performance and efficiency by reusing existing connections rather than creating new ones for each database operation.

Driver Manager

- **Driver Registration:** A Java program must register the appropriate database driver using DriverManager before establishing a database connection.
  - Nowadays, it is done on the fly while connection establishment.
- **Connection Establishment:** Once the driver is registered, DriverManager can be used to obtain a database connection. The getConnection() method of DriverManager is responsible for creating **a connection** to the database.
- **Driver Discovery:** DriverManager uses the registered drivers to find the appropriate driver to handle the connection request. It iterates through the list of registered drivers and delegates the responsibility to the first driver that claims to accept the connection URL.
- **Connection Pooling:** DriverManager is also involved in connection pooling, where it can manage a pool of database connections, helping to improve performance and efficiency by reusing existing connections rather than creating new ones for each database operation.

Driver Manager

- **Driver Registration:** A Java program must register the appropriate database driver using DriverManager before establishing a database connection.
  - Nowadays, it is done on the fly while connection establishment.
- **Connection Establishment:** Once the driver is registered, DriverManager can be used to obtain a database connection. The getConnection() method of DriverManager is responsible for creating **a connection** to the database.
- **Driver Discovery:** DriverManager uses the registered drivers to find the appropriate driver to handle the connection request. It iterates through the list of registered drivers and delegates the responsibility to the first driver that claims to accept the connection URL.
- **Connection Pooling:** DriverManager is also involved in connection pooling, where it can manage a pool of database connections, helping to improve performance and efficiency by reusing existing connections rather than creating new ones for each database operation.