Georgina Dobson - 090470257
Database Systems - CO2209
15th January 2015
Assignment 1

Section A

A(1) Setting up a DBMS

How to install MySQL prerequisites

This set of instructions are for a Windows 7 System with a Google Chrome browser. The MySQL version to be downloaded is 5.6

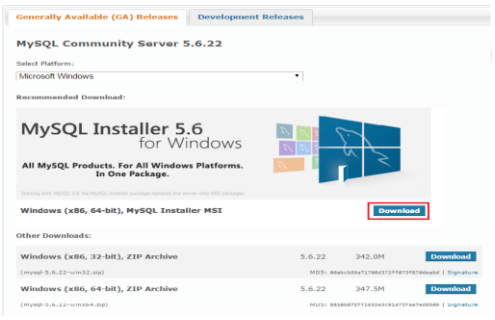
Download

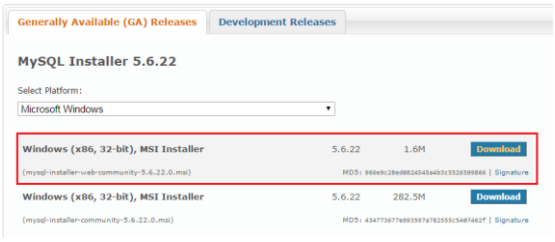
1. A version of MySQL can be downloaded from the following URL http://dev.mysql.com/downloads/mysql

The operating system must be selected in order to provide a link to the appropriate download. The current installation package for Window operating system provides both versions for a 32 bit and a 64 bit system.

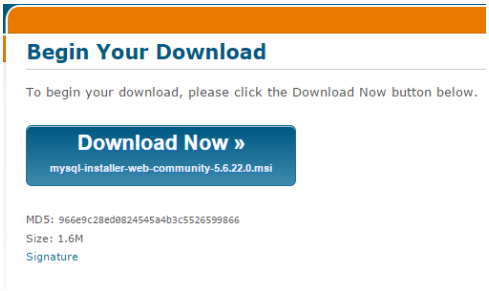
2. The '(mysql-installer-web-community-5.6.22.0.msi)' installation file will require an Internet connection as the various components selected for installation will be downloaded as required.

With no Internet connection available for installation process select '(mysql-installer-community-5.6.22.0.msi)'.



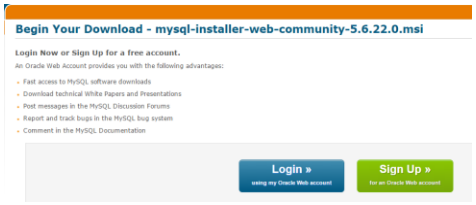


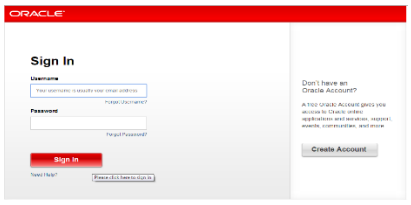
Select download



Begin download

3. MySQL can be downloaded by logging in with an Oracle account or without. For a download without click the **'No thanks, just start my download'** link. For authentication a username and password are required for a valid Oracle account. This can be set up if needed. Note: Regular notifications can be useful in order to keep up to date with bug fixed and future releases.

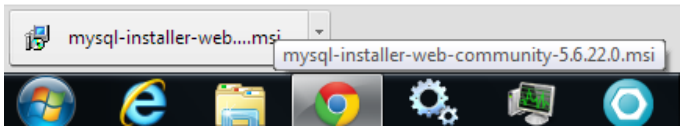




The downloaded file will automatically saved to the specified download area as per browser settings.

Installation

1. The file will appear in a download footer to the bottom left of the browser.

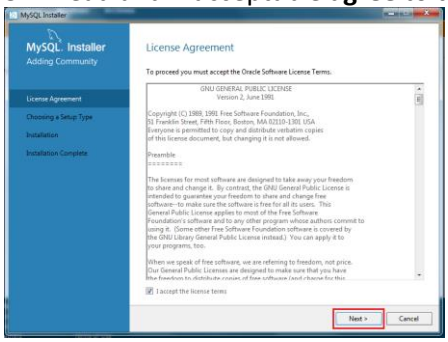


The file can be accessed from here or located in the specified download location.
Click the file to start the installation process.

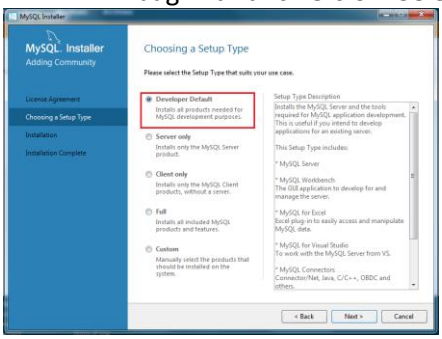
2. Windows 7 will ask for authentication to Run the installation process. Click **Run**.

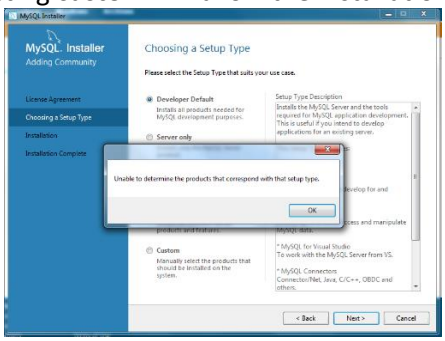
The installation program will run through a set of installation steps, each asking the user to review the status or set the installation information and accept.

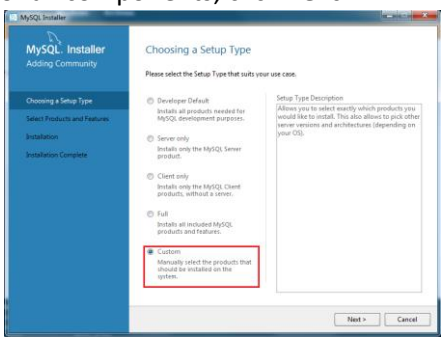
3. Read and if acceptable **agree** to the terms and conditions of the licence agreement, click **next**.



4. Selecting the type of setup. Developer default will install all the components needed, however there this a bug with this version. Selecting **custom** will allow the installation of all components, click **next**.

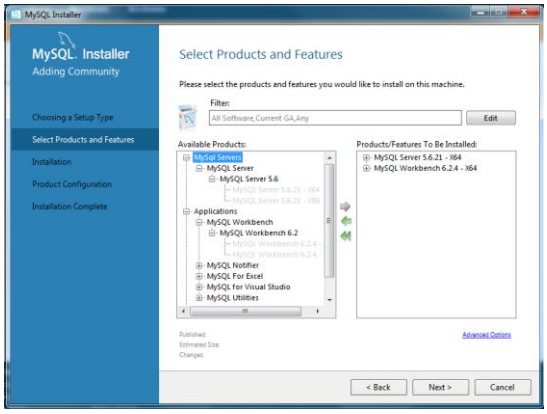






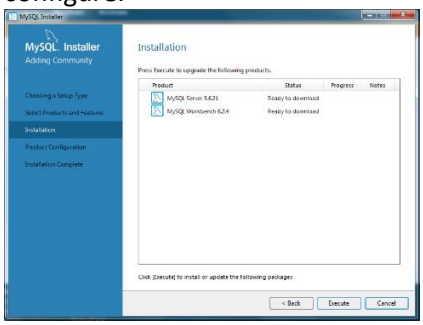
5. Select the following components.

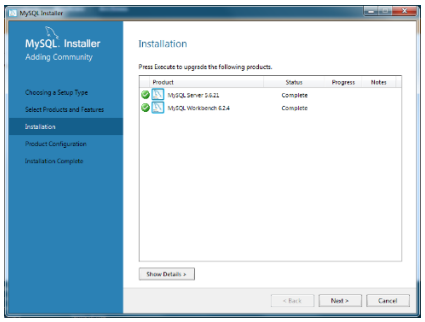
MySQL server – this is the main database management system.
MySQL Workbench – this is the graphical user interface for any operations that need to be performed.
Click **next**.

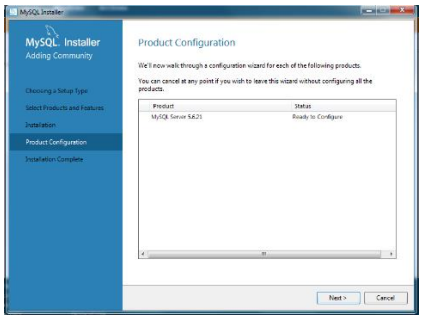


6. The next screen will give the opportunity to review the selections made and execute the installation process.

Check and click **execute**. When complete click **next**. The following view will show that we are now ready to configure.

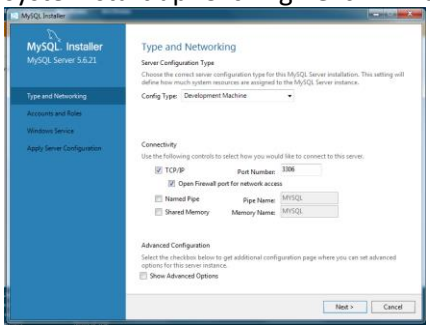


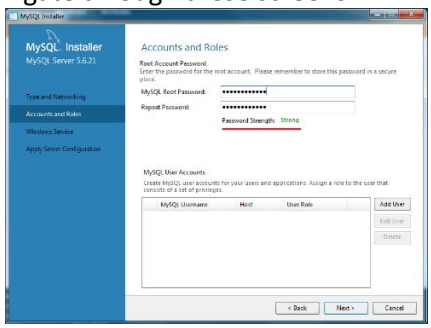


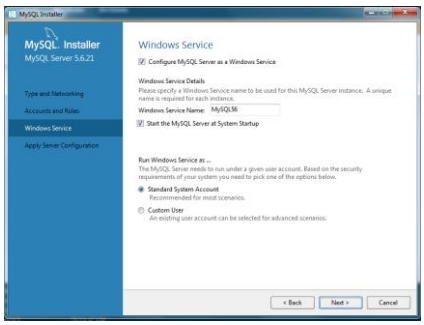


7. In the configuration screens the network information is configured, password must be set and the service named. These screens all have a default values which can be changed, however is highly recommended for the first installation these values are kept as default. Until the user fully understands which settings they are changing and what effect this will have.

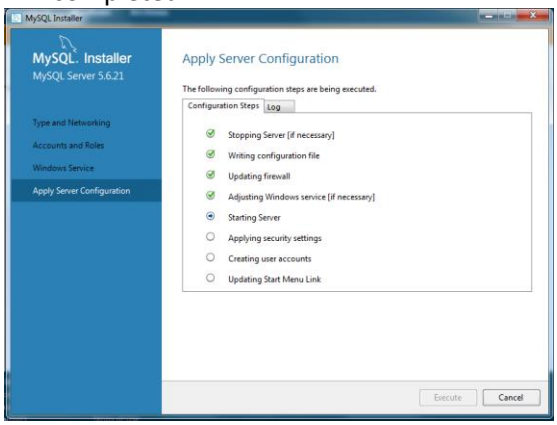
The password is not set as a default value, this must be set by the user. Only a mixture of lowercase, uppercase, numerical and punctuation characters can achieve a strong password. Check the strength indicator.
Configuration as a service will ensure the MySQL server runs in the background and will start on operating system start up. Clicking **next** will navigate through these screens.

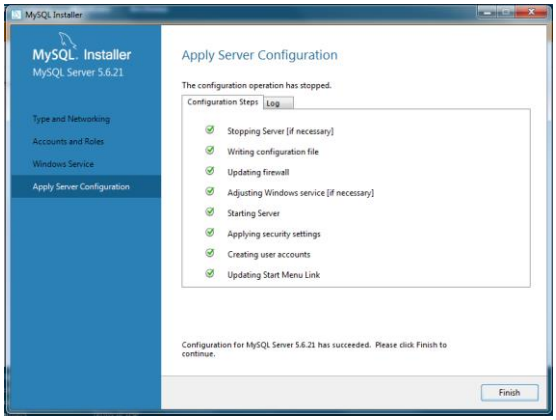




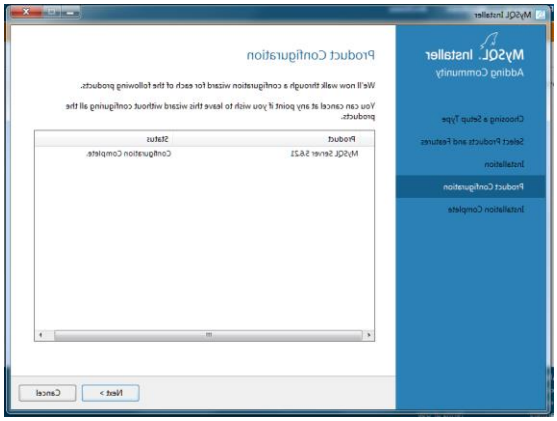


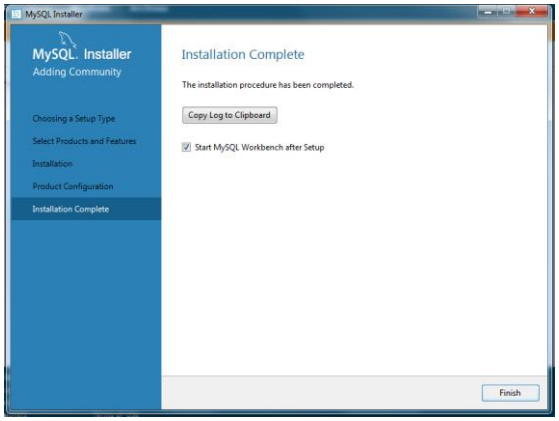
8. The final configuration screen will show a list of configuration actions to be performed. Click **execute** to start this process. Each action will have a green tick icon when completed. Click **finish** when all are completed.





The final screens will show that the configuration and the installation are complete. These can be navigated by clicking the **next** and the **finish** buttons.





A(2) Creating a (toy) database with MySQL.

(1) Create the database

```
mysql> CREATE SCHEMA toy ;
Query OK, 1 row affected (0.02 sec)
```

(2) Select database to use

```
mysql> USE TOY;
Database changed
```

(3) Create tables project, employee_master, employee_project, employee_expert, employee_expert.

```
mysql> CREATE TABLE PROJECT (PCODE VARCHAR(4) NOT NULL UNIQUE,
   -> PNAME VARCHAR(16) NOT NULL UNIQUE,
   -> CUSTNAME VARCHAR(32),
   -> STARTDATE DATE,
   -> ENDDATE DATE,
   -> PRIMARY KEY (PCODE));
Query OK, 0 rows affected (0.29 sec)

mysql> CREATE TABLE EMPLOYEE_MASTER(
   -> ENUM VARCHAR(4) NOT NULL UNIQUE,
   -> SURNAME VARCHAR(32) NOT NULL,
   -> FIRSTNAME VARCHAR(32) NOT NULL,
   -> DOB DATE,
   -> PRIMARY KEY (ENUM));
Query OK, 0 rows affected (0.11 sec)

mysql> CREATE TABLE EMPLOYEE_PROJECT(
   -> ENUM VARCHAR(4) NOT NULL,
   -> PCODE VARCHAR(4) NOT NULL,
   -> DATE_ASSIGNED DATE,
   -> PRIMARY KEY (ENUM, PCODE));
Query OK, 0 rows affected (0.10 sec)

mysql> CREATE TABLE EMPLOYEE_EXPERT (
   -> ENUM VARCHAR(4) NOT NULL,
   -> PACKAGE VARCHAR(10) NOT NULL,
   -> PRIMARY KEY (ENUM,PACKAGE));
Query OK, 0 rows affected (0.08 sec)
```

(4) Insert data

```
mysql> INSERT INTO PROJECT VALUES (
   -> "P123", "Vulweb", "Vulcan, Ltd", "2014-12-01", "2015-06-30"),
   -> ("P342", "Droktrack", "Duroks, Inc", "2014-05-15", "2014-12-31"),
   -> ("P294", "BPeacefund", "Brownpeace", "2013-11-21", "2015-10-30"),
   -> ("P493", "Vulbase", "Vulcan, Ltd", "2015-03-01", "2015-09-30"),
   -> ("P339", "Creatureweb", "Beastworks", "2014-12-01", "2015-12-31"),
   -> ("P332", "CulList", "Dept of Culture", "2014-03-01", "2014-12-31");
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> INSERT INTO EMPLOYEE_MASTER VALUES
   -> ("E246","Patel","Prabhi","1990-11-23"),
   -> ("E892","Williams","Larry","1983-03-10"),
   -> ("E933","Tan","Susan","1987-09-21"),
   -> ("E403","Ahmed","Iqbal","1985-08-26"),
   -> ("E387","Shakeel","Fatima","1988-07-07");
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> INSERT INTO EMPLOYEE_PROJECT VALUES
   -> ("E246","P123","2014-12-05"),("E246","P294","2013-11-21")
   -> ,("E387","P123","2014-12-20"),("E403","P123","2015-04-15"),
   -> ("E403","P332","2014-04-01"),("E933","P294","2013-11-21"),
   -> ( "E933","P332","2014-03-01"),("E933","P342","2014-12-20");
Query OK, 8 rows affected (0.01 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> INSERT INTO EMPLOYEE_EXPERT VALUES
   -> ("E246","Excel"),
   -> ("E246","MySQL"),
   -> ("E246","Word"),
   -> ("E246","Python"),
   -> ("E892","Excel"),
   -> ("E892","Python"),
   -> ("E892","PHP"),
   -> ("E403","MySQL"),
   -> ("E403","Java");
Query OK, 9 rows affected (0.02 sec)
Records: 9  Duplicates: 0  Warnings: 0
```

(5) List all data in tables

```
mysql> SELECT * FROM PROJECT;
+---------+-----------------+---------------------+-----------------+-----------------+
| PCODE | PNAME         | CUSTNAME       | STARTDATE  | ENDDATE    |
+---------+-----------------+---------------------+-----------------+-----------------+
| P123   | Vulweb        | Vulcan, Ltd       | 2014-12-01   | 2015-06-30  |
| P294   | BPeacefund   | Brownpeace      | 2013-11-21   | 2015-10-30 |
| P332   | CulList         | Dept of Culture | 2014-03-01   | 2014-12-31  |
| P339   | Creatureweb | Beastworks      | 2014-12-01   | 2015-12-31  |
| P342   | Droktrack      | Duroks, Inc      | 2014-05-15   | 2014-12-31  |
| P493   | Vulbase        | Vulcan, Ltd       | 2015-03-01   | 2015-09-30 |
+---------+-----------------+---------------------+-----------------+-----------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM EMPLOYEE_MASTER;
+---------+---------------+----------------+---------------+
| ENUM | SURNAME  | FIRSTNAME | DOB          |
+---------+---------------+----------------+---------------+
| E246   | Patel          | Prabhi        | 1990-11-23 |
| E387   | Shakeel      | Fatima        | 1988-07-07 |
| E403   | Ahmed       | Iqbal          | 1985-08-26 |
| E892   | Williams     | Larry          | 1983-03-10 |
| E933   | Tan            | Susan         | 1987-09-21 |
+---------+----------------+----------------+----------------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM EMPLOYEE_PROJECT;
+---------+----------+-----------------------+
| ENUM | PCODE | DATE_ASSIGNED |
+---------+----------+-----------------------+
| E246   | P123    | 2014-12-05         |
| E246   | P294    | 2013-11-21         |
| E387   | P123    | 2014-12-20         |
| E403   | P123    | 2015-04-15         |
| E403   | P332    | 2014-04-01         |
| E933   | P294    | 2013-11-21         |
| E933   | P332    | 2014-03-01         |
| E933   | P342    | 2014-12-20         |
+---------+----------+-----------------------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM EMPLOYEE_EXPERT;
+---------+-------------+
| ENUM | PACKAGE |
+---------+-------------+
| E246   | Excel       |
| E246   | MySQL     |
```

```
| E246   | Python     |
| E246   | Word       |
| E403   | Java       |
| E403   | MySQL      |
| E892   | Excel      |
| E892   | PHP        |
| E892   | Python     |
+---------+---------------+
9 rows in set (0.00 sec)
```

A(3) SQL Queries

(1) **List the first names and surnames of all our employees.**

```
mysql> SELECT FIRSTNAME, SURNAME FROM EMPLOYEE_MASTER;
+-----------------+----------------+
| FIRSTNAME | SURNAME  |
+-----------------+----------------+
| Prabhi          | Patel          |
| Fatima          | Shakeel        |
| Iqbal           | Ahmed          |
| Larry           | Williams       |
| Susan           | Tan            |
+-----------------+----------------+
5 rows in set (0.00 sec)
```

(2) List the first names and surnames of all our employees, ordered on their surnames, in ascending order.

```
mysql> SELECT FIRSTNAME, SURNAME FROM EMPLOYEE_MASTER ORDER BY SURNAME ASC;
+-----------------+----------------+
| FIRSTNAME | SURNAME  |
+-----------------+----------------+
| Iqbal           | Ahmed          |
| Prabhi          | Patel          |
| Fatima          | Shakeel        |
| Susan           | Tan            |
| Larry           | Williams       |
+-----------------+----------------+
5 rows in set (0.00 sec)
```

(3) **What are the employee numbers of employees born before 1987?**

```
mysql> SELECT ENUM FROM EMPLOYEE_MASTER WHERE DOB<"1987-01-01";
+----------+
| ENUM |
+----------+
| E403   |
| E892   |
+----------+
2    rows in set (0.00 sec)
```

(4) **What are the employee number(s) of the oldest employee(s)? (Note that there can be a 'tie', so that there is no single 'oldest' employee.)**

```
mysql> SELECT ENUM FROM EMPLOYEE_MASTER WHERE DOB= ALL(
   -> SELECT MIN(DOB) FROM EMPLOYEE_MASTER);
+----------+
| ENUM |
+----------+
| E892    |
+----------+
1 row in set (0.00 sec)
```

(5) **How many employees are there? (Note: don't just count them by hand! We want the SQL that will yield this answer; and the answer will NOT be a list of employees, but a single number.)**

```
mysql> SELECT COUNT(*) AS TOTAL_NUM_EMPLOYEES FROM EMPLOYEE_MASTER;
+-----------------------------------+
| TOTAL_NUM_EMPLOYEES |
+-----------------------------------+
|                                 5 |
+-----------------------------------+
1 row in set (0.00 sec)
```

(6) **What project has the longest planned time?**

```
mysql> SELECT * FROM PROJECT HAVING MAX(ENDDATE - STARTDATE);
+----------+------------+----------------+-----------------+----------------+
| PCODE | PNAME  | CUSTNAME | STARTDATE  | ENDDATE    |
+-----------+------------+----------------+-----------------+----------------+
| P123     | Vulweb  | Vulcan, Ltd | 2014-12-01  | 2015-06-30 |
+-----------+------------+----------------+-----------------+----------------+
1 row in set (0.00 sec)
```

(7) **What are the employee numbers of employees who work on a project for Vulcan Ltd?**

```
mysql> SELECT EP.ENUM FROM EMPLOYEE_PROJECT AS EP LEFT JOIN PROJECT P ON EP.PCODE=P.PCODE
WHERE P.CUSTNAME="Vulcan, Ltd";
+----------+
| ENUM |
+----------+
| E246   |
| E387   |
| E403   |
+---------+
3 rows in set (0.00 sec)
```

(8) **What are the names of employees who work on a project for Vulcan Ltd?**

```
mysql> SELECT EM.FIRSTNAME, EM.SURNAME FROM EMPLOYEE_PROJECT AS EP LEFT JOIN PROJECT P ON
EP.PCODE=P.PCODE LEFT JOIN EMPLOYEE_MASTER EM ON EP.ENUM=EM.ENUM WHERE
P.CUSTNAME="Vulcan, Ltd";
+-----------------+---------------+
| FIRSTNAME | SURNAME |
+-----------------+---------------+
| Prabhi        | Patel         |
| Fatima        | Shakeel     |
| Iqbal         | Ahmed       |
+-----------------+---------------+
3 rows in set (0.00 sec)
```

(9) **What are the project codes of projects Susan Tan works on?**

```
mysql> SELECT EP.PCODE FROM EMPLOYEE_PROJECT EP
   -> LEFT JOIN EMPLOYEE_MASTER EM ON EP.ENUM=EM.ENUM
   -> WHERE EM.FIRSTNAME="Susan" AND EM.SURNAME="Tan";
+-----------+
| PCODE |
+-----------+
| P294    |
| P332    |
| P342    |
+-----------+
3 rows in set (0.00 sec)
```

(10)**What are the names of the companies on whose projects Susan Tan has worked?**

```
mysql> SELECT CUSTNAME FROM EMPLOYEE_PROJECT EP
   -> LEFT JOIN EMPLOYEE_MASTER EM ON EP.ENUM=EM.ENUM
   -> LEFT JOIN PROJECT P ON EP.PCODE=P.PCODE
   -> WHERE EM.FIRSTNAME="Susan" AND EM.SURNAME="Tan";
+----------------------+
| CUSTNAME       |
+----------------------+
| Brownpeace     |
| Dept of Culture |
| Duroks, Inc        |
+----------------------+
3 rows in set (0.00 sec)
```

(11)**What are the names (first and surname) of employees who are experts in MySQL?**

```
mysql> SELECT EM.FIRSTNAME, EM.SURNAME FROM EMPLOYEE_EXPERT EE
   -> LEFT JOIN EMPLOYEE_MASTER EM ON EE.ENUM= EM.ENUM
   -> WHERE EE.PACKAGE="MySQL";
+-----------------+---------------+
| FIRSTNAME | SURNAME |
+-----------------+---------------+
| Prabhi        | Patel         |
| Iqbal         | Ahmed      |
+----------------+----------------+
2 rows in set (0.00 sec)
```

(12)**What are the employee numbers of employees who work on two or more projects?**

```
mysql> SELECT ENUM AS PROJECT_COUNT FROM EMPLOYEE_PROJECT
   -> GROUP BY ENUM HAVING PROJECT_COUNT >1;
+----------+
| ENUM |
+----------+
| E246    |
| E403    |
| E933    |
+----------+
3 rows in set (0.01 sec)
```

(13)**What are the employee numbers of the employee(s) who have worked on the most projects?**

```
mysql> SELECT ENUM FROM EMPLOYEE_PROJECT
   -> GROUP BY ENUM
   -> HAVING COUNT(ENUM) = (
   -> SELECT MAX(PRO_COUNT) FROM (
   -> SELECT ENUM, COUNT(ENUM) AS PRO_COUNT FROM EMPLOYEE_PROJECT
   -> GROUP BY ENUM) AS DATA);
+----------+
| ENUM |
+----------+
| E933    |
+----------+
1 row in set (0.00 sec)
```

(14)**What is the average number of projects each of our employees has worked on?**

```
mysql> SELECT AVG(PRO_COUNT) AS AVG_COUNT FROM (SELECT ENUM, COUNT(*) AS PRO_COUNT FROM
EMPLOYEE_PROJECT GROUP BY ENUM) AS DATA;
+-----------------+
| AVG_COUNT |
+-----------------+
|       2.0000   |
+-----------------+
1 row in set (0.01 sec)
```

(15)**What are the employee numbers of those employees who are experts in both Python and SQL?**

```
mysql> SELECT DISTINCT ENUM FROM EMPLOYEE_EXPERT WHERE ENUM IN (
   -> SELECT ENUM FROM EMPLOYEE_EXPERT WHERE PACKAGE="Python")
   -> AND ENUM IN (
   -> SELECT ENUM FROM EMPLOYEE_EXPERT WHERE PACKAGE="MySQL");
+----------+
| ENUM |
+----------+
| E246    |
+----------+
1 row in set (0.00 sec)
```

(16)**What are the employee numbers of those employees who are experts in both Python or SQL?**

```
mysql> SELECT DISTINCT ENUM FROM EMPLOYEE_EXPERT WHERE ENUM IN (
   -> SELECT ENUM FROM EMPLOYEE_EXPERT WHERE PACKAGE="Python")
   -> OR ENUM IN (
   -> SELECT ENUM FROM EMPLOYEE_EXPERT WHERE PACKAGE="MySQL");
+----------+
| ENUM |
+----------+
| E246   |
| E403   |
| E892   |
+----------+
3 rows in set (0.01 sec)
```

(17)**What are the employee numbers of those employees who are not experts in Python?**

```
mysql> SELECT DISTINCT ENUM FROM EMPLOYEE_EXPERT WHERE ENUM IN (
   -> SELECT ENUM FROM EMPLOYEE_EXPERT WHERE PACKAGE!="Python")
   -> AND ENUM NOT IN (
   -> SELECT ENUM FROM EMPLOYEE_EXPERT WHERE PACKAGE="Python");
+----------+
| ENUM |
+----------+
| E403    |
+----------+
1 row in set (0.01 sec)
```

(18)**What are the employee numbers of those employees, if any, who have *not* been assigned to a project and who know Python?**

```
mysql> SELECT DISTINCT ENUM FROM EMPLOYEE_MASTER
   -> WHERE ENUM NOT IN (
   -> SELECT DISTINCT ENUM FROM EMPLOYEE_PROJECT)
   -> AND ENUM IN (
   -> SELECT ENUM FROM EMPLOYEE_EXPERT WHERE PACKAGE="Python");
+----------+
| ENUM |
+----------+
| E892    |
+----------+
1 row in set (0.00 sec)
```

(19)**We want to add a new employee, whose number will be** E787, whose name is Shiran Ebadi, and who was born on the 8th of August, 1989, to the database. What command can we use do that? mysql> INSERT INTO EMPLOYEE_MASTER VALUES ("E787","Shiran","Ebadi","1989-08-08");

Query OK, 1 row affected (0.05 sec)

mysql> SELECT * FROM EMPLOYEE_MASTER;

ENUM	SURNAME	FIRSTNAME	DOB
E246	Patel	Prabhi	1990-11-23
E387	Shakeel	Fatima	1988-07-07
E403	Ahmed	Iqbal	1985-08-26
E787	Shiran	Ebadi	1989-08-08
E892	Williams	Larry	1983-03-10
E933	Tan	Susan	1987-09-21

6 rows in set (0.01 sec)

(20)What is wrong with the following query:
SELECT SURNAME
FROM **EMPLOYEE-MASTER**
WHERE **DOB = NULL;**

mysql> SELECT SURNAME FROM EMPLOYEE_MASTER WHERE DOB=NULL;

This will give a false result, the query should read: SELECT * FROM EMPLOYEE_MASTER WHERE DOB IS NULL;
Null is not a value and not to be confused with " " empty space or 0. Null represents empty space and therefore the value of a field cannot be compared with a comparison operator. It is possible to check for a null using the syntax WHERE <some field> IS NULL;

A(4) Getting help

The question I have found on the Stackoverflow website, is as follows:

My table is "Activity_table", which have 4 columns. How can I create a function, which SUM each Persons 2 previous and current activities? My Activity_table

```
ID  PID AID Act
1   1   1   12
2   1   2   32
3   2   1   5
4   1   3   21
5   2   2   12
6   2   3   19
7   1   4   11
8   2   4   6
```

PID-PersonID; AID-ActivitieID; Act-Activitie Value

My target:

```
ID  PID AID Act SUM
1   1   1   12  12
2   1   2   32  44
3   2   1   5   5
4   1   3   21  65
5   2   2   12  17
6   2   3   19  36
7   1   4   11  64
8   2   4   6   37
```

Sum1=12; Sum2=32+12; Sum3=5; Sum4=21+32+12; Sum5=12+5; Sum6=19+12+5; Sum7=11+21+32; Sum8=6+19+12; Thank you,

The user wants to make a sum of the previous 2 action and current action, for each person.
The following is the first suggestion:

```
SELECT ID,PID,AID,Act,
    (SELECT SUM(Act)
     FROM Activity_table
     WHERE ID <= a.ID
     AND ID >= a.ID - 2
     AND PID = a.PID)
FROM Activity_table a;
```

This is the second suggestion:

```
SELECT ID,PID,AID,Act,
    (SELECT SUM(Act)
     FROM Activity_table
```

```
     WHERE ID <= a.ID
     AND PID = a.PID)
FROM Activity_table a;
```

Then if you need only the flow of SUM of a particular PID, you can change it like this,

```
SELECT ID,PID,AID,Act,
    (SELECT SUM(Act)
     FROM Activity_table
     WHERE ID <= a.ID
     AND PID = a.PID)
FROM Activity_table a
WHERE PID = 2;
```

Both suggestions haven't understood the question. The first suggestion doesn't take into account that we need to group by person, if he used the, sequential id for the sum he will get invalid results. As ID 3 is for person 2 when the value for person ID 4 is calculated it will sum the actual act from ID4, ID3 and ID2.
The second suggestion filters by person with the where condition, but again doesn't take into account that the filtered results may not have and sequential ID. This solution also doesn't take into account the 2 previous values needed.

In conclusion

The forum are a really good source of help, but you should always display your question in detail and always test the answers given produce the correct results before use.

Section B

B(1) Some basics

a. "The boss wants us to add more information to the employee relations… she wants us to add some attributes about each employee's education."
Level: **Conceptual**

b. "We're binging in an outside auditor to look at our sales data… I want you to create a relation for him to see with a sample – say, every thousandth record, for all sales last month – of the SALES table, but leave out the customer name and address."
Level: **External**

c. "The whole database have been running more and more slowly this year. I want you to investigate replacing our hard discs with Solid-State Storage."
Level: **Internal**

B(2) What is a relation?

	NAME	NUMBER	LOCATION
1	John Smith	123	New York
2	Akira Kurosawa	234	Tokyo
3	Mario Vargas Llosa	345,367,393	Buenos Aires
4	NULL	786	Tokyo
5	John Simth	123	New York
6	Akira Kurosawa	123	Tokyo

1. The primary key NAME has a NULL value. (Row 4)
2. Two tuples have the same primary key. (Row 2 and 6)
3. Two tuples are completely identical. (Row 1 and 5)
4. The numbers stored for Mario Vargas Llosa should be stored separately in another relation and linked with a foreign key. (Row 3)

B(3) Relational concepts

a. **Attribute** of the relation or table is a column of data sharing the same type, SURNAME column in the relation EMPLOYEE_MASTER is an attribute.
b. **Tuple** of the relation is the row of data. The employee with the ENUM E246 in the relation EMPLOYEE_MASTER and all associated information horizontally is the tuple.
c. **Degree** of a relation is the number of attributes contained within it. The relation EMPLOYEE_MASTER has a degree of 4.
d. **Cardinality** is the number of tuples contained in the relation. EMPLOYEE_MASTER has a cardinality of 5.
e. **Candidate Key** is an attribute that can be identified for its uniqueness for one attribute only. In the example tables below the ENUM attribute can be a candidate key.
f. **Primary Key** is a candidate key that is for uniquely identifying the tuple.
g. **Alternate Key** is a primary key that has been selected that cannot be categorized as a candidate key, in the case where the relation has and auto increment ID.

h. **Foreign Key** is an attribute that forms a relation with another relation by matching the candidate key in the other relation. In the example relations, the ENUM is the foreign key in the EMPLOYEE_EXPERT as it matches the primary key in the EMPLOYEE_MASTER relation.
i. **Functional Dependency** is the relationship between attribute where one attribute depends on another. In EMPLOYEE_EXPERT the PACKAGE attribute functionally depends on the ENUM attribute, as the PACKAGE can be determined from ENUM. ENUM → PACKAGE.
The ENUM cannot be determined from PACKAGE therefore ENUM does not functionally depend on PACKAGE.
j. **Determination** is the dependent from functional dependency. In the example above the determination would be the PACKAGE.
k. **Compound** (**or composite**) **Key** is when the candidate key is made up of 2 or more attributes, for example see the ENUM and PACKAGE attribute from the EMPLOYEE_EXPERT relation, neither and uniquely identify the tuple on their own but must work as a combination.
l. **Null value** represents empty space and cannot be likened to " " a space or a 0 value.

EMPLOYEE_MASTER

ENUM	SURNAME	FIRSTNAME	DOB
E246	Patel	Prabhi	1990-11-23
E298	Williams	Larry	1983-03-10
E933	Tan	Susan	1987-09-21
E403	Ahmed	Iqbal	1985-08-26
E387	Shakeel	Fatima	1988-07-07

EMPLOYEE_EXPERT

ENUM	PACKAGE
E246	Excel
E246	MySQL
E246	Word
E246	Python
E892	Excel
E892	Python
E892	PHP
E403	MySQL
E403	Java

B(4) Data formats I

The table below will show various data types a definition and the range of values that will be accepted.

	Type	Definition	Range
(0)	Tinyint	A very small integer	Signed -128 to 127 Unsigned 0 to 255
(1)	Char	A fixed length string	1 to 255 characters
(2)	Varchar	A variable length string	1 to 255 characters
(3)	Bigint	A large integer	Signed -9223372036854775808 to 9223372036854775807, Unsigned 0 to 18446744073709551615
(4)	Mediumint	A medium integer	Signed -8388608 to 8388607, Unsigned 0 to 16777215
(5)	Smallint	A small integer	Signed -32768 to 32767, Unsigned 0 to 65535
(6)	Decimal	Unpacked floating point number	Number or digits permitted to the left of the decimal point in the range of 1 to 65 and denoted by (M). Number of digits to the right of the decimal point should be in the range of 0 to 30, denoted by (N) and be no greater than the value of M.
(7)	Date	Date	YYYY-MM-DD range 1000-01-01 to 9999-12-31
(8)	Time	Time	HH:MM:SS or HHH:MM:SS -838:59:59 to 838:59:59 The large range is needed to store elapsed time in addition to actual time.
(9)	Datetime	Data and time combination	YYYY-MM-DD HH:MM:SS 1000-01-01 00:00:00 to 9999-12-31 23:59:59
(10)	Timestamp	Timestamp	YYYYMMDDHHMMSS 1970-01-01 00:00:01 UTC to 2038-01-19 03:14:07 UTC

The timestamp is recorded in UTC – As the timestamp is recorded each time the tuple is updated it can be used for documenting at what time an action occurred, for example, when the tuple was created or last modified.

B(5) Data Formats II

Phone numbers are primarily made up of a string of numerical characters, ie: 0035799564456.
We could store this value as a type integer. However, this will lead to a few issues. Quite often users like to identify the country code or area code, (00357) 99564456 or 00357-99564456. The characters () - are not acceptable for the type integer. If the phone number is entered as 0845-123456, this will be treated as a sum. On insert the sum will be execute and the value saved will be -122611.
In recent years, the introduction of text messages being sent from applications has been introduced. For this the phone numbers need to be stored with a prefix + character, for example, +447789123452 and this not possible with an integer type. It is also difficult to implement a partial search for the number as an integer.

B(6) Primary Keys

Five design consultants suggested 5 types of primary key for the following table.

SUBJECT_ID	SUBJECT	DATE	GRADE
S123	Maths	2013-03-15	43
S234	Physics	2013-03-17	68

With the additions of 2 new tuples to the table, the following results should be seen. When a new tuple for S123, Maths, **2014**-30-15, grade 76 is entered this should be accepted.
When a new tuple for S234, Physics, **2014**-03-17, grade 83 is entered where 2014 is entered incorrectly as **2013**, this should be rejected.
The best primary key proposal is that of Omar, which will allow the student to enter an exam multiple times, score the same grade in the same subject, but not on the same data. See the table below.

	S123 – Should be accepted	S134 – Should be rejected	Proposed primary key
Michael	REJECT	REJECT	Student number
Narwaz	REJECT	REJECT	Student number + Subject
Omar	ACCEPT	REJECT	Student number + Subject + Date
Dariuz	ACCEPT	ACCEPT	Student number + Subject + Date
Shawna	ACCEPT	ACCEPT	New column with an auto incremented value.

B(7) The physical Layer

Indexes

Indexing is the method used to improve the speed of data retrieval operations. For relations containing many attributes and tuples, the retrieval time can be slow whilst each tuple is traversed.

Indexing works to improve the efficiency by creating a copy of the relation which only contains attributes that are used as common search terms. An **index** only contains one attribute where a **composite index** can contain multiple attributes. The index is structured as a doubly linked list that stores keys and references to the data stored in the relation, which is stored at a different physical location. All index entries are stored in a specified order to allow faster searching. In a doubly linked list the indexes are not stored in physical sorted order but contain a virtual sorted order where a link will show the location of the next index in the list, doubly linked list also store links to the previous index, therefore the list can be traversed forward or in reverse.

The indexes are stored in a balanced tree, this means that all root to leaf paths are of equal depth. The leaf stores sets of sorted indexes. The maximum key value from a leaf is stored in the parent node in the branch, this nodes' maximum value is also stored in its parent node. This process continues until the keys are store in the balanced tree root node.

Functions can also be used on indexes in order to store only variants of the attribute e.g. upper(LASTNAME) only store the last name as upper case characters.
Commonly the indexed attributes are the primary and foreign keys.

In order to perform a search the lookup process must follow 3 steps:

1. Traverse the balanced tree searching for associated values.
2. Follow the chain in the leaf nodes of the linked lists
3. Retrieve the date from the tuples in the relations to which the index refers.

Example

A relation PERSON with 8 attributes LASTNAME, FIRSTNAME, ADDRESS1, ADDRESS2, CITY, POSTCODE, DOB. (Where DOB denotes date of birth)
The attributes that are used as search terms most commonly are FIRSTNAME, LASTNAME, PHONE, DOB.
If the index is set to LASTNAME when the look up 'WHERE LASTNAME="Smith"' is used. The stored index will contain the index and a reference to the actual tuple for this person.
If a composite index is used containing 3 attributes, LASTNAME, FIRSTNAME, DOB. All three attributes must be used in the search term. If only 2 are used, FIRSTNAME, LASTNAME, the search will first look up the leaf node containing first name, then search for last name, therefore performing 2 lookups. This is a major disadvantage.

Other disadvantages are that whilst the performance is being improved for searching and sorting the data insert, update, delete operations become slower as the indexes must also be updated.

Searching for partial values can also cause an issues. For our PERSON relation with the index DOB, if we only know the birthday and not the year of birth the search term will be "%-03-11", where % denote a wild card for any given value. This can be addressed by using a reverse function where the field is matched in reverse. In the case where the year is entered as a 2 digit format and the day is unknown the search term will be "%66-03-%". In this case it is not possible to use the indexes to improve the searching speed.

References

http://en.wikipedia.org/wiki/Database_index
http://stackoverflow.com/questions/1108/how-does-database-indexing-work
http://use-the-index-luke.com/sql/anatomy

Solid State Drives vs Hard Disk Drives

Hard Disk Drives (HDD) have been around of many years and are the main data storage medium at present. These are slowly being replaced by a new type called Solid State Drives (SSD).

The HDD consists of rotating disks or platters that consist of tracks and sectors. For an operation the disc must start spinning if it isn't already, a read / write head actuator can move over the various positions on the rotating platters in order to read / write data to the tracks. The faster the platters can spin the faster data can be read or written.
With these type of disks there is a latency, which is the time delay between a request and the action. The latencies can be the seek time, the average time taken to access the track, and rotational latency, the time taken for the platter to spin to the correct location in order to be read or written to.
The transfer rate measures how long it takes for the head to read from or write to the platter.
The main advantages to the HDD is the cost and the amount of storage space available.
The disadvantages are that it can be easily damaged by dropping and it is possible to be damaged by foreign objects entering the mechanism.

The SSD are made up of microchips using NAND technology, solid state memory, very similar to a USB storage device. As there are no moving parts the speed at which data can be written or read is much faster than that of the HDD. The data can be accessed instantaneously using system wide addressing.
The data is split into word length sections and as the NAND technology is used the memory is said to be non-volatile, which means that a loss of power will not result in data loss.
The advantage of this type of disk is that they can access data faster and therefore a system boot up will be faster. They are safe from any kind of damage that may be caused by magnetic fields. As there are no moving parts any drops or knocks to the drive are less likely to cause any damage.
The only disadvantage is the cost is much higher than the HDD and although the technology has been around for a while the cost of providing similar storage volumes to the HDD has been very expensive. There are still limitations on storage space within these drives.

Conclusion

The solid state drives are much faster and will provide better data access efficiency, however this must be considered against the extra cost and the reduction in storage space.

References

http://pctechguide.com/hard-disks/hard-disk-hard-drive-performance-transfer-rates-latency-and-seek-times
http://pcguide.com/ref/hdd/perf/perf/spec/posSeek-c.html
http://www.storagereview.com/ssd_vs_hdd