

Introduction

What is SQL?

SQL (pronounced ess-cue-ell) is the standard database language, supported to some extent by every database product in the market. It has an internationally agreed syntax defined in a document which is well over 600 pages long. Most companies implement a subset of the standard and introduce minor additions to enable it to work with their specific database product. SQL is a fourth generation language (4GL), which suggests that users concern themselves with writing queries that answer questions, rather than worrying about the technical details of *how* such queries should work.

History of SQL

Until the mid 1980's, almost all mainframe databases were either hierarchical or network databases, which lacked a rigorous mathematical foundation and were often implemented in a messy or inefficient style. In 1970, E.F. Codd published a paper, which provided a relational model for databases based on set theory. It took ten years for IBM to produce System R based on this relational model, which included a query language called *sequel*. During the period 1980 to 1983 IBM announced versions for various Operating Systems and machines. Such was the power of IBM at the time that many other companies announced a version of SQL for their machines, either as a replacement or as an alternative for their proprietary language.

The Standards Bodies

To prevent there from being multitudes of versions with differing dialects, ANSI and the ISO standards committee have produced documents detailing various standards and extensions, SQL-86, SQL-89, SQL-92, SQL 2003 and SQL 2008. SQL 92 is the most common version and is detailed in

- ISO/IEC 9075:1992, "Information Technology --- Database Languages --- SQL"
- ANSI X3.135-1992, "Database Language SQL"

The current standard is SQL 2008 – although few databases implement all the current features.

Further information about the current state of the standard can be found on the web – unfortunately formal versions from SQL92 onwards are copyright so it can be difficult to obtain copies. Descriptions of SQL86 seem to be more freely available.

The structure of a database

The Relational Model

To use SQL, the programmer must be familiar with the relational model, which is at the heart of all Relational Database Management Systems (often abbreviated to RDBMS). In simple terms, data is manipulated in *Tables* (sometimes called *Relations*), at a conceptional level several layers above how the data is actually stored. SQL users don't have to know anything about files, pointers, links or blocks.

The Table/Relation concept

EMP table

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
405	MARCH	ADMIN	938	13-Jun-97	18000		2
936	CASSY	ADMIN	734	23-Jul-02	19500		3
912	HAYES	ADMIN	824	04-Jun-01	21000		2
557	BELL	SALES	734	26-Mar-00	22500	500	3
690	AHMAD	SALES	734	05-Dec-97	22500	1400	3

Each table must have a unique name (EMP in this case). Each row (record) is a series of interconnected data items, an employee in this case. This snapshot of EMP table shows 5 rows/5 employees. A cell must hold one Atomic value (i.e. a value that wouldn't normally be divided into any smaller parts). Values can be Text (i.e. letters or Alphanumeric characters), Numbers (so that associated mathematical operations can be performed) or other types such as Dates, Times or Currency.

The table rows must be 'interrelated' in some way. The EMP table contains a set of employees in a company. Each column must have a unique column name (to that table), which indicates the kind of data items shown in the column 'below'.

Every table is supposed to mirror a mathematical set and as such there is no significance in the row or column ordering. Theoretically there are also no duplicate rows allowed (actual databases may allow duplicates). A *table* is sometimes called a *relation* in mathematics.

Operations on a table are based on the mathematical principles of *selection*, *projection*, *join* and *product* (these terms come from an area of mathematics called Relational Algebra). In an actual database these operations are performed through a language such as SQL. SQL is a non-procedural language which has an English like structure and allows the user to specify what is wanted rather than how the computer should perform the task (the major difference between a third and fourth generation language).

About MySQL

MySQL is a database server product in the same style as Oracle's Database Server or Microsoft's SQL Server. It is installed on a server computer (typically in a server room) and users access the database remotely using local tools - these could be dedicated pieces of installed software, graphical interfaces interacted with through a web browser or command line tools.



Usage options

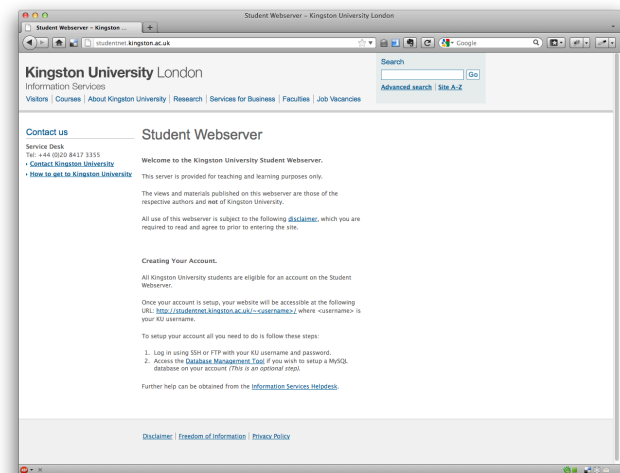
As students at Kingston University you have access to a centralised implementation of MySQL which uses a web based tool for interaction (phpmyadmin) – this is the default setup described in these notes. The downside to this architecture is that it requires Internet access to perform the tasks and activities here.

An alternative solution is to install a development version on your own machine – Apple Mac users are advised to try MAMP (available from <http://www.mamp.info>), Windows users should try XAMPP (<http://www.apachefriends.org/en/xampp.html>). Both these include Apache, PHP and MySQL (there are other distributions available). Separate installation instructions are available for MAMP.

Activity: Creating your MySQL username, password and database

To use the Kingston University MySQL server, you must first create an account (a database, username and password will be set up during this process).

Use a web browser and navigate to studentnet.kingston.ac.uk

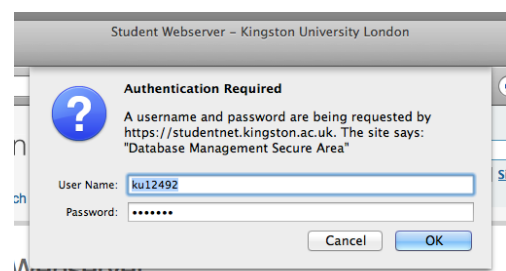


At the bottom of the page is a link to the *Database Management Tool*. Click on the link

To setup your account all you need to do is follow these steps:

1. Log in using SSH or FTP with your KU username and password.
2. Access the [Database Management Tool](#) if you wish to setup a MySQL database on your account (*This is an optional step*).

Use your Kingston University username (kxxxxxxx) and password to login in.



The database management screen allows you to Create a database (or to delete/resit one if you've already created one).

Type in a new database password – do NOT use your general password as you may be using this password at a later point in the course in plain text on the screen.

Database Management

Through this interface you can manage the MySQL database associated with your account on the Studentnet Web Server.

Create MySQL Database

Database Name:

Username:

Database Password:

Confirm Password:

Create Database

Write down the *database name*, *username*, *database host address* and *port number* details, as you will need to know them later.

Click on the phpMyAdmin link to access your MySQL database.

Database Management

The database **db_ku12492** has been successfully created.

You can now access the database using the following connection parameters:

- Database Host: studentnet.kingston.ac.uk
- Port: 3306
- Username: ku12492
- Password: As provided in the previous form

You can now:

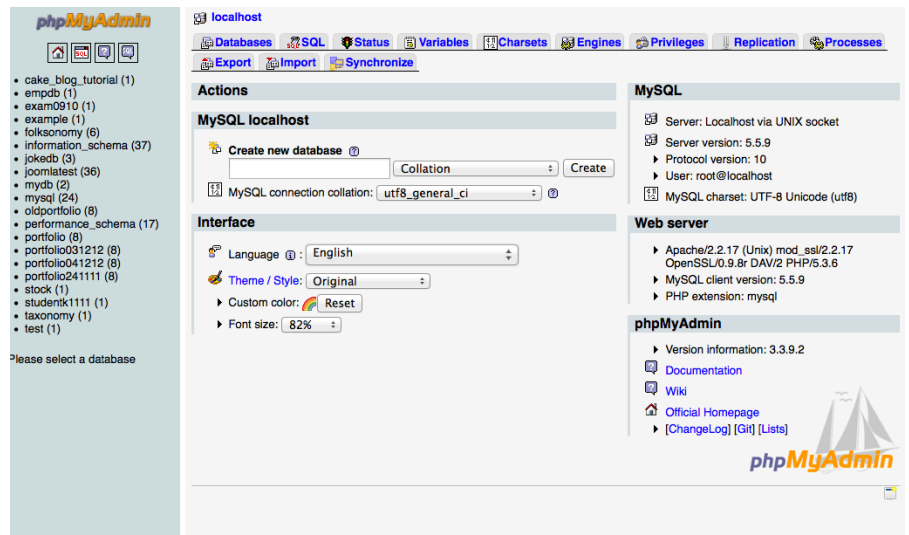
- Go back to the [Database Management Tool](#)
- Manage your database with [phpMyAdmin](#)
- Visit your [Student Website](#)? (You will need to have logged in to the Student Webserver for this to be available)

Use your kxxxxxxx username and the new password to login to the database

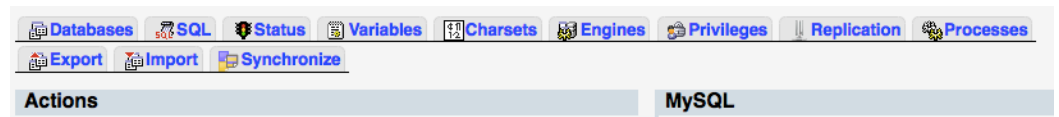


The image shows the phpMyAdmin login interface. At the top, there is a logo with a sailboat and the text 'phpMyAdmin'. Below the logo, it says 'Welcome to phpMyAdmin'. There is a 'Language' dropdown menu set to 'English'. Below that is a 'Log in' section with a 'Username' field containing 'ku12492' and a 'Password' field with masked characters. A 'Go' button is at the bottom right of the login section.

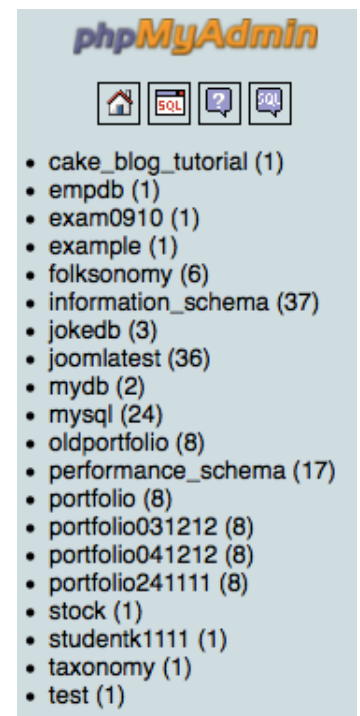
The phpMyAdmin interface lists Databases down the column on the left and activities you can perform across the top, shown in tabs



Tabs across the top



Hint: Before you do anything else – you must **always** either select a database from the column on the left (or create a new one). As part of the creation process you will have an entry for a single database, typically called something like db_kxxxxxxx



Using MySQL to create and use databases

The employee database is used in various SQL notes – these steps show how to create the *employee* database, with its EMP, DEPT and GRADE tables.

Click on the entry for your database db_kxxxxxxx in the column on the left

You should then be taken into an empty database. Note how the tabs change to show the different actions that you can perform on the *employee* database.

The key here is to remember that you should always select the database you wish to work with in the column on the left before attempting to run any queries, otherwise you won't have the correct tabs/functionality across the top of the page.

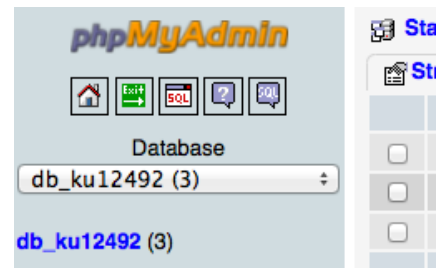
Creating the EMP, DEPT and GRADE tables

Rather than using the interface to create tables and insert records, we can run a batch set of SQL statements to set the tables up.

Download the zip file at the bottom of this web page and extract the text file inside

<http://www.barryavery.com/blog/2009/10/24/relational-databases-session-1/>

This file contains SQL statements to *create* and *insert* the three tables used in the examples

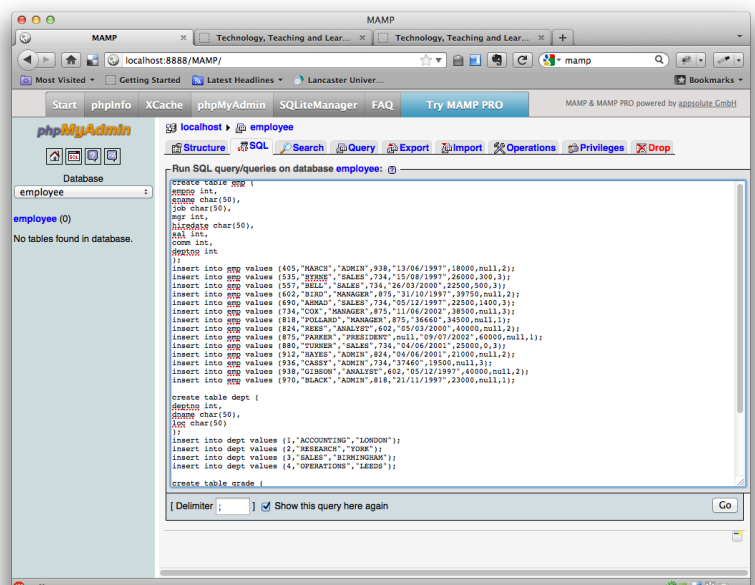


```
create table emp (
  empno int,
  ename char(50),
  job char(50),
  hiredate char(50),
  sal int,
  comm int,
  deptno int
);
insert into emp values (405,"MARCH","ADMIN",938,"13/06/1997",18000,null,2);
insert into emp values (535,"BRYNE","SALES",734,"15/08/1997",26000,300,3);
insert into emp values (557,"BELL","SALES",734,"25/03/2000",22500,500,3);
insert into emp values (602,"BIRD","MANAGER",875,"31/10/1997",39750,null,2);
insert into emp values (606,"ARMAD","SALES",734,"05/12/1997",22500,1400,3);
insert into emp values (734,"COX","MANAGER",875,"11/06/2002",38500,null,3);
insert into emp values (818,"POLLARD","MANAGER",875,"36/68",34500,null,1);
insert into emp values (824,"KEES","ANALYST",602,"05/03/2000",40000,null,2);
insert into emp values (875,"PARKER","PRESIDENT",null,"09/07/2002",60000,null,1);
insert into emp values (880,"TURNER","SALES",734,"04/06/2001",25000,0,3);
insert into emp values (912,"HAYES","ADMIN",824,"04/06/2001",21000,null,2);
insert into emp values (938,"CASSY","ADMIN",734,"07/60",19500,null,3);
insert into emp values (938,"GIBSON","ANALYST",602,"05/12/1997",40000,null,2);
insert into emp values (970,"BLACK","ADMIN",818,"21/11/1997",23000,null,1);

create table dept (
  deptno int,
  dname char(50),
  loc char(50)
);
insert into dept values (1,"ACCOUNTING","LONDON");
insert into dept values (2,"RESEARCH","YORK");
insert into dept values (3,"SALES","BIRMINGHAM");
insert into dept values (4,"OPERATIONS","LEEDS");

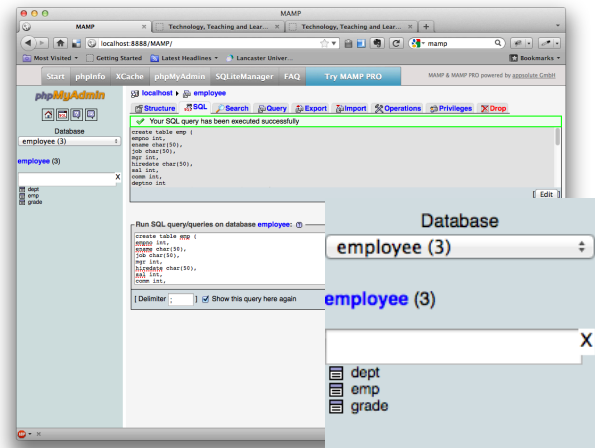
create table grade (
  grade int,
  losal int,
  hisal int
);
insert into grade values (1,17000,21999);
insert into grade values (2,20000,23999);
insert into grade values (3,24000,29999);
insert into grade values (4,30000,49999);
insert into grade values (5,50000,99999);
```

In phpMyAdmin, click on the SQL tab and copy/paste in all the SQL commands from the text file. Click the *Go* button to execute the statements - note that you should have previously selected the *employee* database for this to work.



If this has worked correctly you should have three tables (and data) shown in the column on the left

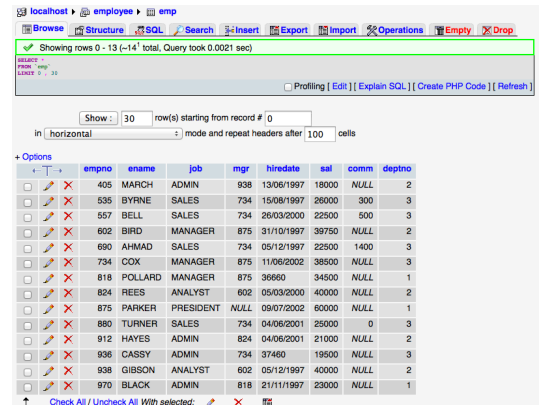
You can use the SQL tab to run any SQL statement (or series of statements separated by ';').



Other parts of the interface

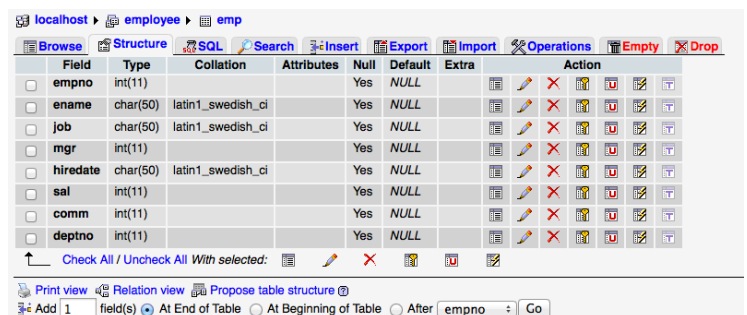
Click on a table name (in the column on the left) to be able perform the following actions

Browse a table's data



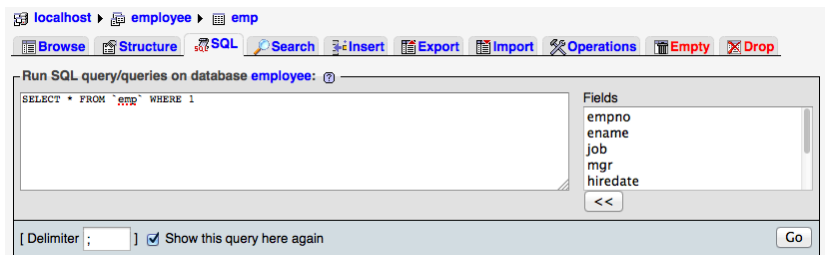
empno	ename	job	mgr	hiredate	sal	comm	deptno
405	MARCH	ADMIN	938	13/06/1997	18000	NULL	2
535	BYRNE	SALES	734	15/08/1997	26000	300	3
557	BELL	SALES	734	26/03/2000	22500	500	3
602	BIRD	MANAGER	875	31/10/1997	39750	NULL	2
690	AHMAD	SALES	734	05/12/1997	22500	1400	3
734	COX	MANAGER	875	11/06/2002	38500	NULL	3
818	POLLARD	MANAGER	875	36660	34500	NULL	1
824	REES	ANALYST	602	05/03/2000	40000	NULL	2
875	PARKER	PRESIDENT	NULL	09/07/2002	60000	NULL	1
880	TURNER	SALES	734	04/06/2001	25000	0	3
912	HAYES	ADMIN	824	04/06/2001	21000	NULL	2
936	CASSY	ADMIN	734	37460	19500	NULL	3
938	GIBSON	ANALYST	602	05/12/1997	40000	NULL	2
970	BLACK	ADMIN	818	21/11/1997	23000	NULL	1

See a table's structure



Field	Type	Collation	Attributes	Null	Default	Extra	Action
empno	int(11)			Yes	NULL		
ename	char(50)	latin1_swedish_ci		Yes	NULL		
job	char(50)	latin1_swedish_ci		Yes	NULL		
mgr	int(11)			Yes	NULL		
hiredate	char(50)	latin1_swedish_ci		Yes	NULL		
sal	int(11)			Yes	NULL		
comm	int(11)			Yes	NULL		
deptno	int(11)			Yes	NULL		

Run SQL statements (not just on the selected table but on any table)



Run SQL query/queries on database employee: ?

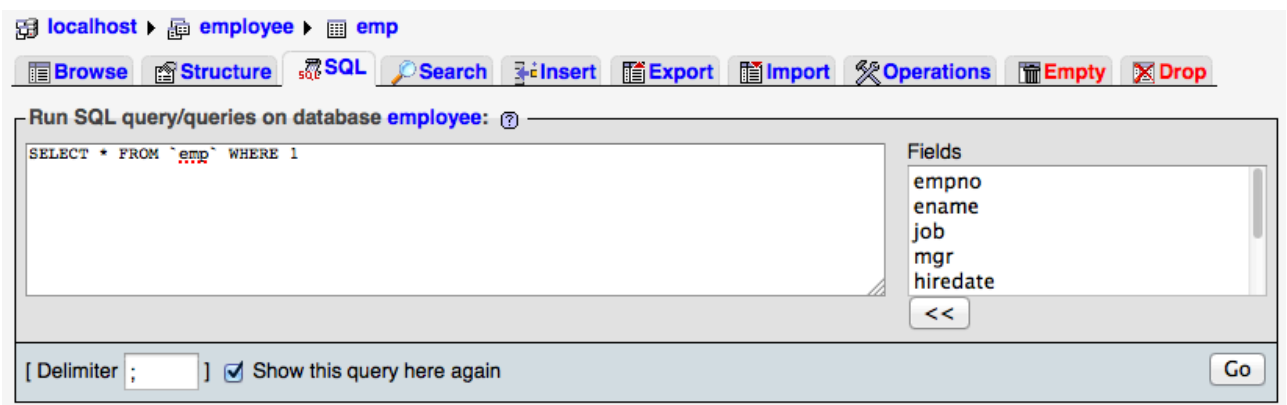
SELECT * FROM `emp` WHERE 1

Fields: empno, ename, job, mgr, hiredate

[Delimiter ;] ☒ Show this query here again

You can also *Insert* rows (records), *Export* or *Import* a table's data in a variety of formats, *Empty* a table (i.e. delete all the rows) or *Drop* a table (remove the table entirely).

For learning and using SQL – the most important functionality comes from the SQL panel



localhost ▶ employee ▶ emp

Browse Structure SQL Search Insert Export Import Operations Empty Drop

Run SQL query/queries on database employee: ?

SELECT * FROM `emp` WHERE 1

Fields: empno, ename, job, mgr, hiredate

[Delimiter ;] ☒ Show this query here again

Tables used on this course

The database used in these exercises contains three tables, EMP, DEPT and GRADE.

- EMP is a table containing employee information such as name, employment date, salary and employee number.
- DEPT is a table containing the names of each department.
- GRADE contains the salary scales for each salary grade.

Activity: View the EMP, DEPT and GRADE table

To view these tables, single click on the table name in the column on the left. Clicking on the EMP table shows something like the following

The screenshot shows the phpMyAdmin interface for a database named 'db_ku12492'. The left sidebar lists the tables: 'dept', 'emp', and 'grade'. The 'emp' table is selected. The main area displays the table structure and a list of 14 rows of employee data. The table has columns: empno, ename, job, mgr, hiredate, sal, comm, and deptno. The data is displayed in a table format with 14 rows and 8 columns. The first row is highlighted in orange. The interface includes a top navigation bar with buttons for Browse, Structure, SQL, Search, Insert, Export, Import, Operations, and Empty. A status bar at the bottom indicates 'Showing rows 0 - 13 (14 total, Query took 0.0003 sec)'.

empno	ename	job	mgr	hiredate	sal	comm	deptno
535	BYRNE	SALES	734	15/08/1997	26000	300	3
557	BELL	SALES	734	26/03/2000	22500	500	3
602	BIRD	MANAGER	875	31/10/1997	39750	NULL	2
690	AHMAD	SALES	734	05/12/1997	22500	1400	3
734	COX	MANAGER	875	11/06/2002	38500	NULL	3
818	POLLARD	MANAGER	875	14/05/2000	34500	NULL	1
824	REES	ANALYST	602	05/03/2000	40000	NULL	2
875	PARKER	PRESIDENT	NULL	09/07/2002	60000	NULL	1
880	TURNER	SALES	734	04/06/2001	25000	0	3
912	HAYES	ADMIN	824	04/06/2001	21000	NULL	2
936	CASSY	ADMIN	734	23/07/2002	19500	NULL	3
938	GIBSON	ANALYST	602	05/12/1997	40000	NULL	2
970	BLACK	ADMIN	818	21/11/1997	23000	NULL	1
405	MARCH	ADMIN	938	13/06/1997	18000	NULL	2

The contents of the three tables are reproduced here and may be useful when writing queries later.

EMP table

empno	ename	job	mgr	hiredate	sal	comm	deptno
535	BYRNE	SALES	734	15/08/1997	26000	300	3
557	BELL	SALES	734	26/03/2000	22500	500	3
602	BIRD	MANAGER	875	31/10/1997	39750	NULL	2
690	AHMAD	SALES	734	05/12/1997	22500	1400	3
734	COX	MANAGER	875	11/06/2002	38500	NULL	3
818	POLLARD	MANAGER	875	14/05/2000	34500	NULL	1
824	REES	ANALYST	602	05/03/2000	40000	NULL	2
875	PARKER	PRESIDENT	NULL	09/07/2002	60000	NULL	1
880	TURNER	SALES	734	04/06/2001	25000	0	3
912	HAYES	ADMIN	824	04/06/2001	21000	NULL	2
936	CASSY	ADMIN	734	23/07/2002	19500	NULL	3
938	GIBSON	ANALYST	602	05/12/1997	40000	NULL	2
970	BLACK	ADMIN	818	21/11/1997	23000	NULL	1
405	MARCH	ADMIN	938	13/06/1997	18000	NULL	2

DEPT table

deptno	dname	loc
1	ACCOUNTING	LONDON
2	RESEARCH	YORK
3	SALES	BIRMINGHAM
4	OPERATIONS	LEEDS

GRADE table

grade	losal	hisal
1	17000	21999
2	22000	23999
3	24000	29999
4	30000	49999
5	50000	99999

What is a Query?

A Query can be classified as a question that we require the database to provide an answer to. Example queries could be questions such as 'Who earns more than £23,000 per year?' or 'What will salaries look like if we give everyone a 5% pay increase?'

After creating the query, we execute it and the database produces a response.

Queries can be edited and then re-executed if they are not quite correct.

Select queries allow us to view data without changing any of the core information.

- Show me what would happen if we gave everyone a 5% increase

is different to

- Give everyone a 5% increase

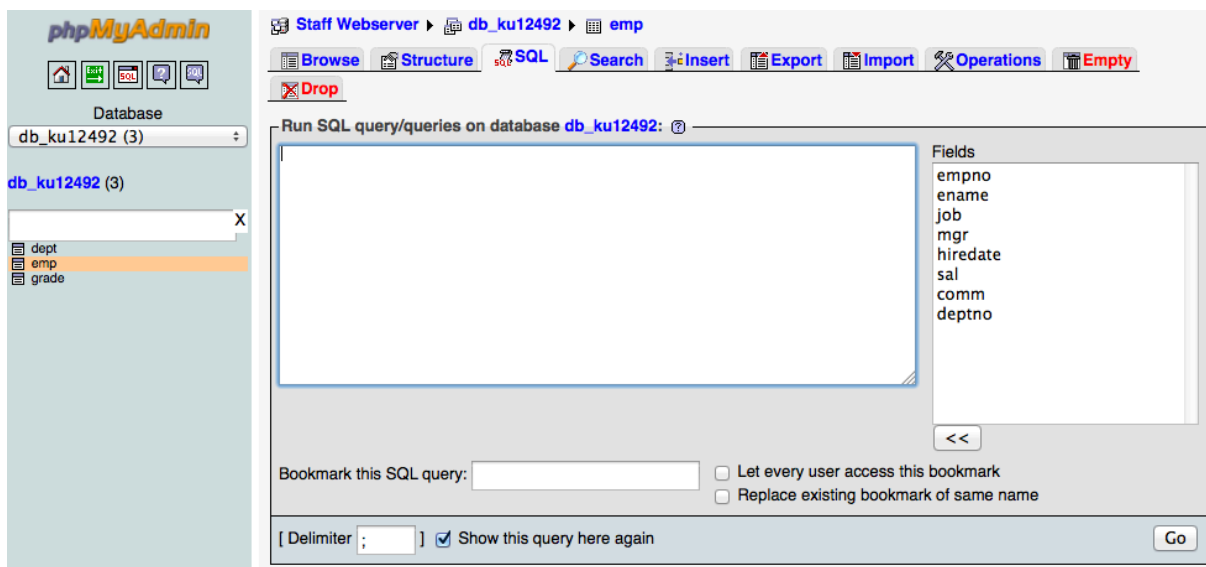
These queries (called *update* queries) will be covered later in the course.

Activity: Writing your first query

The SQL statement

```
select *  
from emp
```

Will show all the contents of the EMP table. To execute this query, select the SQL tab in phpMyAdmin and type in the query. Press GO to see the results



The results of the query (or an error message if the query is not quite correct) should be displayed as shown.

Database: db_ku12492 (3)

db_ku12492 (3)

Showing rows 0 - 13 (14 total, Query took 0.0003 sec)

SELECT *
FROM emp
LIMIT 0, 30

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

+ Options

	empno	ename	job	mgr	hiredate	sal	comm	deptno
<input type="checkbox"/>	535	BYRNE	SALES	734	15/08/1997	26000	300	3
<input type="checkbox"/>	557	BELL	SALES	734	26/03/2000	22500	500	3
<input type="checkbox"/>	602	BIRD	MANAGER	875	31/10/1997	39750	NULL	2
<input type="checkbox"/>	690	AHMAD	SALES	734	05/12/1997	22500	1400	3
<input type="checkbox"/>	734	COX	MANAGER	875	11/06/2002	38500	NULL	3
<input type="checkbox"/>	818	POLLARD	MANAGER	875	14/05/2000	34500	NULL	1
<input type="checkbox"/>	824	REES	ANALYST	602	05/03/2000	40000	NULL	2
<input type="checkbox"/>	875	PARKER	PRESIDENT	NULL	09/07/2002	60000	NULL	1
<input type="checkbox"/>	880	TURNER	SALES	734	04/06/2001	25000	0	3
<input type="checkbox"/>	912	HAYES	ADMIN	824	04/06/2001	21000	NULL	2
<input type="checkbox"/>	936	CASSY	ADMIN	734	23/07/2002	19500	NULL	3
<input type="checkbox"/>	938	GIBSON	ANALYST	602	05/12/1997	40000	NULL	2
<input type="checkbox"/>	970	BLACK	ADMIN	818	21/11/1997	23000	NULL	1
<input type="checkbox"/>	405	MARCH	ADMIN	938	13/06/1997	18000	NULL	2

Check All / Uncheck All With selected:

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

Types of statements

SQL statements use the following reserved words:

Select	Used to retrieve data from the database, the most commonly used statement
Insert	Used to enter, remove or change rows from a table.
Delete	Together with Select, collectively known as the DML or
Update	Data Manipulation Language
Create	Used to set up, change or remove data structures such
Alter	as tables, views or indexes. Collectively known as the DDL
Drop	or Data Definition Language
Grant	Used to give or remove access rights to data and
Revoke	data structures within an SQL database
Validate	

A *reserved word* is one that is specified as part of the core language and cannot be redefined by the user. Hence attempting to name a column 'select' will result in an error.

Projection in Relational Algebra and SQL

The Select statement

The Select statement is used to pull out and display information from a table. Its basic structure has this form:

```
SELECT selectexpression
FROM tables
```

In its simplest form, the select expression is a series of column names, separated by a comma.

select ename,job from emp;	select deptno from emp;	select ename,empno from emp;
-------------------------------	----------------------------	---------------------------------

Select mirrors the Relational Algebra Projection operator (designated by a Π) which can be visualised as something that slices appropriate columns from a table. In Relational Algebra columns to be included in the result are generally listed (in subscript), before the relation name. The result of this operation is another relation / table.

$\Pi_{\text{ename, job}}(\text{Emp})$	$\Pi_{\text{deptno}}(\text{Emp})$	$\Pi_{\text{ename, empno}}(\text{Emp})$
Gives the ename and job column	Gives all unique department numbers	Gives ename and empno from the emp relation

The result of a Relational Algebra operation will not contain duplicate rows (like in set theory). This is not quite the same as the SQL select operation as most databases list any duplicates entries.

The items after the *select* keyword can be one or more of the following, separated by commas:

	Example	Explanation
Column name	empno	Show the empno column
*	*	Show <i>all</i> columns
Arithmetic expression	sal+1000	Show salary + 1000
Character expression	ename&job	Combine two columns together
Arithmetic function	round(sal, 2)	Round salary to a certain number of places

There are other possible entries such as Date expressions, Aggregate functions, or actual values (character, numeric or date values). Date and character values must be given in single quotes (i.e. 'Smith' or '21-APR-86'). More information will be given on the date format later in the course.

Activities: Trying various Projection queries

The following examples demonstrate variations on the *Select* statement. Type these queries in and verify that they produce the results indicated.

Examples

Display all columns and rows from table EMP

```
select *
from emp;
```

empno	ename	job	mgr	hiredate	sal	comm	deptno
535	BYRNE	SALES	734	15/08/1997	26000	300	3
557	BELL	SALES	734	26/03/2000	22500	500	3
602	BIRD	MANAGER	875	31/10/1997	39750	NULL	2
690	AHMAD	SALES	734	05/12/1997	22500	1400	3
734	COX	MANAGER	875	11/06/2002	38500	NULL	3
818	POLLARD	MANAGER	875	14/05/2000	34500	NULL	1
824	REES	ANALYST	602	05/03/2000	40000	NULL	2
875	PARKER	PRESIDENT	NULL	09/07/2002	60000	NULL	1
880	TURNER	SALES	734	04/06/2001	25000	0	3
912	HAYES	ADMIN	824	04/06/2001	21000	NULL	2
936	CASSY	ADMIN	734	23/07/2002	19500	NULL	3
938	GIBSON	ANALYST	602	05/12/1997	40000	NULL	2
970	BLACK	ADMIN	818	21/11/1997	23000	NULL	1
405	MARCH	ADMIN	938	13/06/1997	18000	NULL	2

Display department numbers, employee names and their manager's number.

```
select deptno,ename,mgr
from emp;
```

In Relational Algebra:

$\Pi_{deptno, ename, mgr}(emp)$

deptno	ename	mgr
3	BYRNE	734
3	BELL	734
2	BIRD	875
3	AHMAD	734
3	COX	875
1	POLLARD	875
2	REES	602
1	PARKER	NULL
3	TURNER	734
2	HAYES	824
3	CASSY	734
2	GIBSON	602
1	BLACK	818
2	MARCH	938

Display all employees, along with the amount that a 50 % rise would make to the salary.
Note that this doesn't change the value in the emp table.

```
select ename, (sal/100*50)+sal  
from emp;
```

In Relational Algebra:

$$\Pi_{\text{ename}, (\text{sal}/100*50)+\text{sal}} (\text{emp})$$

ename	(sal/100*50)+sal
BYRNE	39000.0000
BELL	33750.0000
BIRD	59625.0000
AHMAD	33750.0000
COX	57750.0000
POLLARD	51750.0000
REES	60000.0000
PARKER	90000.0000
TURNER	37500.0000
HAYES	31500.0000
CASSY	29250.0000
GIBSON	60000.0000
BLACK	34500.0000
MARCH	27000.0000

Note that SQL generates a column name (as the mathematical expression `(sal/100*50)+sal` can't be used). To force a more sensible name to be used, use ' as `columnname`'.

Example: Try using *newsal* as the new calculated column name

```
select ename, (sal/100*50)+sal as newsal  
from emp;
```

Display all the unique department numbers in the EMP table

```
select distinct deptno  
from emp;
```

In Relational Algebra:

$$\Pi_{\text{deptno}} (\text{emp})$$

deptno
3
2
1

Concatenate and display the employee name and employee number in one column.

```
select concat(empno, "-", ename) AS field1
from emp;
```

In Relational Algebra:

$\Pi_{\text{empno} \& \text{"-"} \& \text{ename}} (\text{emp})$

field1
535-BYRNE
557-BELL
602-BIRD
690-AHMAD
734-COX
818-POLLARD
824-REES
875-PARKER
880-TURNER
912-HAYES
936-CASSY
938-GIBSON
970-BLACK
405-MARCH

Concatenate means combine two pieces of data together into one. This operation is typically performed on text. The SQL *concat* function takes a collection of items and combines them into a string.

Activities: Writing your own Projection queries

In the following exercises, the query must be specified to produce the suggested result. There are spaces for you to write the SQL query in. Try and write down the equivalent Relational Algebra expression. Note that you may have to use the AS command to get correct column headings in SQL.

1. Display all the information in the table called grade

SQL:

GRADE	LOSAL	HISAL
1	17000	21999
2	22000	23999
3	24000	29999
4	30000	49999
5	50000	99999

In Relational Algebra:

2. Display the employee names and their hiredates

SQL:

ENAME	HIREDATE
MARCH	13/06/1997
BYRNE	15/08/1997
BELL	26/03/2000
BIRD	31/10/1997
AHMAD	05/12/1997
COX	11/06/2002
POLLARD	14/05/2000
REES	05/03/2000
PARKER	09/07/2002
TURNER	04/06/2001
HAYES	04/06/2001
CASSY	23/07/2002
GIBSON	05/12/1997
BLACK	21/11/1997

In Relational Algebra:

3. Display all the different types of jobs

SQL:

job
ADMIN
ANALYST
MANAGER
PRESIDENT
SALES

In Relational Algebra:

4. Display a list of employee names with monthly earnings (calculated from salary / 12).
Note that you should call the resulting column *monthlsal*

SQL:

ename	monthlsal
MARCH	1500
BYRNE	2166.66666
BELL	1875
BIRD	3312.5
AHMAD	1875
COX	3208.33333
POLLARD	2875
REES	3333.33333
PARKER	5000
TURNER	2083.33333
HAYES	1750
CASSY	1625
GIBSON	3333.33333
BLACK	1916.66666

In Relational Algebra:

5. Display a list of the locations where departments are located

SQL:

loc
LONDON
YORK
BIRMINGHA
LEEDS

In Relational Algebra:

6. Display the grade, and the difference a 12% rise in the losal figure would make to each grade

SQL:

grade	increase
1	19040
2	24640
3	26880
4	33600
5	56000

In Relational Algebra:

7. Display the employee name and employee number as a single column separated by a hyphen, along with what difference a 5% increase in salary would make

SQL:

employ	newsal
405-MARCH	18900
535-BYRNE	27300
557-BELL	23625
602-BIRD	41737.5
690-AHMAD	23625
734-COX	40425
818-POLLARD	36225
824-REES	42000
875-PARKER	63000
880-TURNER	26250
912-HAYES	22050
936-CASSY	20475
938-GIBSON	42000
970-BLACK	24150

In Relational Algebra: