



Syllabus Content:


8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)

-  show understanding that DBMS software carries out:
 - all creation/modification of the database structure using its DDL
 - query and maintenance of data using its DML
-  show understanding that the industry standard for both DDL and DML is Structured Query Language (SQL)
 - show understanding of a given SQL script
 - write simple SQL (DDL) commands using a sub-set of commands for:

Notes and guidance

- creating a database (CREATE DATABASE)
- creating a table definition (CREATE TABLE) including the creation of attributes with appropriate data types:

Notes and guidance

- CHARACTER
 - VARCHAR(n)
 - BOOLEAN
 - INTEGER(n)
 - REAL
 - DATE
 - TIME
 - changing a table definition (**ALTER TABLE**)
 - adding a primary key or foreign key to a table (**ADD PRIMARY KEY**)
 - add a foreign key to a table (**FOREIGN KEY (field) REFERENCES Table (Field)**)
-  write a SQL script for querying or modifying data (DML) which are stored in (at most two) database tables

Notes and guidance

- Queries including: SELECT, FROM, WHERE, ORDER BY, GROUP BY, INNER JOIN, SUM, COUNT, AVG
- Data maintenance including: INSERT INTO, DELETE FROM, UPDATE

Queries

Databases allow us to store and filter data to find specific information. A database can be queried using a variety of methods, although this depends on the software you are using.

Databases can use query languages or graphical methods to interrogate the data.

Query language

Query language is a **written language used only to write specific queries**. This is a powerful tool as the user can define precisely what is required in a database. SQL is a popular query language used with many databases.

Query by example (QBE)

QBE allows the user to **create queries based on a template**, usually a set of filters presented in a graphical form. If you are using database software it might have an option to connect blocks

and set the filters you want. The system presents a blank record and lets you specify the fields and values that define the query.

Database management software like MySQL, Microsoft Access and Oracle have front-end graphical interfaces which make it easier to run QBE queries.

Boolean operators

In a database we often need to filter the data to group certain results. Boolean **operators** are used to filter databases using **AND**, **OR** or **NOT**. They can search multiple fields at the same time to help us retrieve the data that we need. They are used because they provide results which are 'true' or 'false'.

Search engines also make use of Boolean operators to filter results.

AND

AND is used to search records that contain one piece of information **AND** another.



A query for the words *brown AND shoes* would return results that contain the words brown and shoes.

In general, search engines treat the query *brown shoes* as *brown AND shoes*, which means that all results will contain both words, eg *brown trousers and red shoes for sale*.

OR

OR is used to search for records that contain EITHER one piece of information **OR** another.

for example *black shoes OR white shoes*. This would present results for **any shoes that were black or white**.

Most search engines use the OR function best if the search statements are defined by speech marks, eg "*brown shoes*" **OR** "*black jeans*" would show pages which either contain brown shoes or black jeans.

NOT

NOT is used to exclude results.

The query *shoes NOT brown* will return results that **contain the word shoes but NOT the word brown**.

Arithmetic operators

A query can also be performed using **arithmetic operators**. These help to make specific searches related to numerical data.

Functions of arithmetic operators

This table shows some arithmetic operators and their functions:

Operator	Meaning
=	Equals
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<>	Not equal to

Using arithmetic operators

The table below shows some BBC TV programme listings:

ID	Title	Genre	Duration (mins)	Channel
01	EastEnders	Drama	30	BBC1
02	Dragons' Den	Entertainment	60	BBC2
03	The Voice	Entertainment	75	BBC3
04	Blue Peter	Children's	25	CBBC

ID	Title	Genre	Duration (mins)	Channel
05	Wild Brazil	Nature	60	BBC4
06	Match of the Day	Sport	80	BBC1
07	Dick and Dom	Comedy	10	CBBC

Queries are useful for searching for specific conditions. You might want to find entertainment programmes on BBC3. A query for these conditions would look like this:

```
SELECT * FROM Programmes
WHERE Genre='Entertainment'
AND Channel='BBC3';
```

This would return the programme 'The Voice'.

You may want a programme that is less than 20 minutes long or is a nature programme. A query for these conditions would look like this:

```
SELECT * FROM Programme
WHERE Duration < 20
OR Genre = 'Nature';
```

This would return the programmes "Dick and Dom" and "Wild Brazil".

SQL

SQL is a programming language used to search and query databases. SQL gives you the ability to customise your queries. It is often used within database programs.

Uses of SQL

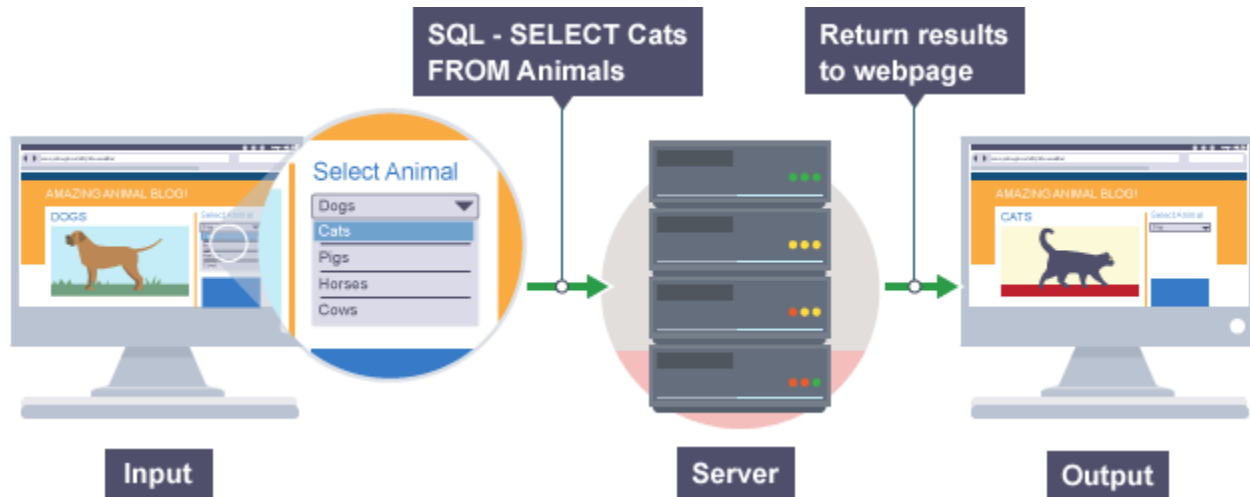
SQLite and MySQL are popular open source database applications. If you use a blog site, it is very likely that MySQL has been running behind the scenes to store entries as records and allow you to add, edit and delete blog posts.

SQL databases can be used to create lots of applications for use on the internet. They are often used by large companies because they allow the data tables to be stored on secure servers. A SQL server simply stores the data for a database - it does not provide front-end features. Therefore, the user does not need to know that a database is working behind a website.

A SQL database is manipulated using SQL code. SQL has also been designed so that lots of users can access it at the same time - it has a high capacity for storage space.

Syntax

For example: "SELECT these fields FROM this table WHERE this is happening".



Working with SQL: Step 1

The following examples show how to use SQL for basic database functions. We will work through a series of steps to create this table, showing a record for a collection of BBC programmes:

ID	Title	Genre	Duration
01	EastEnders	Drama	30
02	Newsnight	Current affairs	50
03	The Voice	Entertainment	75
04	Blue Peter	Children's	25
05	Wild Brazil	Nature	60
06	Sherlock	Drama	90

ID	Title	Genre	Duration
07	Top Gear	Entertainment	60

Data Definition Language (DDL)

Data Definition Language (DDL) is the part of SQL provided for creating or altering tables. These commands only create the structure. They do not put any data into the database.

SQL (DDL) commands and scripts

SQL (DDL) command	Description
CREATE DATABASE	Creates a database
CREATE TABLE	Creates a table definition
ALTER TABLE	Changes the definition of a table
PRIMARY KEY	Adds a primary key to a table
FOREIGN KEY ... REFERENCES ...	Adds a foreign key to a table

Data types:




Data types for attributes	Description
CHARACTER	Fixed length text
VARCHAR(n)	Variable length text
BOOLEAN	True or False; SQL uses the integers 1 and 0
INTEGER	Whole number
REAL	Number with decimal places
DATE	A date usually formatted as YYYY-MM-DD
TIME	A time usually formatted as HH:MM:SS

Creating a table

A table can be created in SQL code using the following template:

```
CREATE DATABASE Databasename;
CREATE TABLE tablename (column1 datatype, column2 datatype,
column3 datatype);
ALTER TABLE tablename ADD PRIMARY KEY (column);
ALTER TABLE table2 FOREIGN KEY (column REFERENCES table1(column));
```

The creator of the table has to decide:

-  the name of the table - "tablename"
-  the title of each field - "column"
-  the data type that is required for each field - "datatype"(eg character strings - 'varchar' or numbers 'int')

In the TV programmes example the table can be created using the following SQL code:

```
CREATE TABLE Programs
(ID int(2), Title varchar(20), Genre varchar(20), Duration
int(3));
ALTER TABLE Programs ADD PRIMARY KEY (ID);
```

The data type **varchar** indicates that only character **strings** are allowed, and the number in brackets indicates the maximum number of digits for each data type.

These examples show that once the database has been created the tables can be created and the attributes defined. It is possible to define a primary key and a foreign key within the **CREATE TABLE** command but the **ALTER TABLE** command can be used as shown (it can also be used to add extra attributes).

```
CREATE DATABASE BandBooking;
CREATE TABLE Band (
BandName varchar2(25),
NumberOfMembers number(1));
ALTER TABLE Band ADD PRIMARY KEY(BandName) ;
ALTER TABLE Band-Booking ADD FOREIGN KEY (BandName
REFERENCES Band (BandName));
```

Data Manipulation Language (DML):

Data Manipulation Language (DML) is used when a database is first created, to populate the tables with data. It can then be used for ongoing maintenance. The following code shows a selection of the use of the commands:

```
INSERT INTO Band ('ComputerKidz',5);
INSERT INTO Band-Booking (BandName, BookingID)
VALUES ('ComputerKidz', '2016/023');
UPDATE Band
SET NumberOf Members = 6;
DELETE FROM BandName
WHERE BandName = 'ITWizz';
```

Working with SQL: Step 2

You can query this table using SQL code.

ID	Title	Genre	Duration
01	EastEnders	Drama	30
02	Newsnight	Current affairs	50
03	The Voice	Entertainment	75
04	Blue Peter	Children's	25
05	Wild Brazil	Nature	60
06	Sherlock	Drama	90

You may decide that you wish to sort the programmes by duration. The SQL code needed would look like the example below:

```
SELECT Programmes.Duration, Programmes.Title
FROM Programmes
ORDER BY Programmes.Duration;
```

- the **SELECT** statement states which fields to look at - the Title and Duration fields

- the **FROM** statement states which table to look at - Programmes
- the **ORDER BY** statement sorts the Duration field in ascending order by default

This table shows the results from this query:

Duration	Title
25	Blue Peter
30	EastEnders
50	Newsnight
60	Wild Brazil
75	The Voice
90	Sherlock

Working with SQL: Step 3

The SQL **WHERE** statement is used to isolate one record or several records with similar attributes.

ID	Title	Genre	Duration
01	EastEnders	Drama	30
02	Newsnight	Current affairs	50
03	The Voice	Entertainment	75
04	Blue Peter	Children's	25
05	Wild Brazil	Nature	60

ID	Title	Genre	Duration
06	Sherlock	Drama	90

The following code searches the Title field of the table to find the words 'The Voice'.

```
SELECT Programmes.ID, Programmes.Title, Programmes.Genre,
Programmes.Duration
FROM Programmes
WHERE ((Programmes.Title)="The Voice");
```

- the **WHERE** statement specifies which text to look for

This table shows the results from this query:

ID	Title	Genre	Duration
03	The Voice	Entertainment	75

Alternatively, you could find all of the programmes which are less than 30 minutes long using this code:

```
...
WHERE ((Programmes.Duration)<30);
```

This table shows the results from this query:

ID	Title	Genre	Duration
04	Blue Peter	Children's	25

Wildcards

The wildcard uses the * symbol, and is used in place of any number of unknown characters. For example, the following code searches for all programmes with the letter **i** in the title:

```
...
WHERE ((Programmes.Title) LIKE "*i*");
```

This table shows the results from this query:

ID	Title	Genre	Duration
02	Newsnight	Current Affairs	50
03	The Voice	Entertainment	75
05	Wild Brazil	Nature	60

Adding and editing data with SQL

SQL can also be used to add and edit the data stored on an SQL server.

Adding records

To add new data you use the function **INSERT INTO**.

If a new BBC programme is created and you want to add it to the database, then you would need to use the INSERT INTO function followed by the VALUES separated by a comma:

```
INSERT INTO Programmes
```

```
VALUES (07, "Top Gas", "Entertainment", 60);
```

Note that quotes surround string entries. Numbers do not have quotes.

After inputting this code the table would look like this:

ID	Title	Genre	Duration
01	EastEnders	Drama	30
02	Newsnight	Current affairs	50
03	The Voice	Entertainment	75
04	Blue Peter	Children's	25
05	Wild Brazil	Nature	60
06	Sherlock	Drama	90
07	Top Gas	Entertainment	60

Editing records

SQL also has the **UPDATE** function for editing data.

In this example, there was an error with the previous entry and you need to change the name from "Top Gas" to "Top Gear". You need to:

1. Identify the table to be updated using **UPDATE** - UPDATE Programmes
2. Identify what the field needs to be changed to using **SET** - SET Programmes.Title = "Top Gear"
3. Identify which record needs to be updated using **WHERE** - WHERE Programmes.ID = 07

In full, this is:

```
UPDATE Programmes
SET Programmes.Title = "Top Gear"
WHERE Programmes.ID = 07;
```

This will go to the **Programmes** table, find the programme with an **ID** of 07 and change the **Title** field to **Top Gear**. The amended table will look like this:

ID	Title	Genre	Duration
01	EastEnders	Drama	30
02	Newsnight	Current affairs	50
03	The Voice	Entertainment	75
04	Blue Peter	Children's	25
05	Wild Brazil	Nature	60
06	Sherlock	Drama	90
07	Top Gear	Entertainment	60

SQL also has the **DELETE** function for deletion of data. So running the below mentioned query

```
DELETE FROM Programs
WHERE ID = '07';
```

Will delete

07	Top Gear	Entertainment	60
----	----------	---------------	----

COUNT() Syntax

The **COUNT()** function returns the number of rows that matches a specified criterion.

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

DEMO DATABASE

Table Products

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

```
SELECT COUNT(ProductID)
FROM Products;
```

The **COUNT()** function returns the number of rows that matches a specified criteria.

SUM() Syntax

The `SUM()` function returns the total sum of a numeric column.

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

Demo Database

Below is a selection from the "**OrderDetails**" table in the sample database:

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

SUM() Example

The following SQL statement finds the sum of the "Quantity" fields in the "OrderDetails" table:

Example

```
SELECT SUM (Quantity)
FROM OrderDetails;
```

The following SQL statement finds the sum of the "Quantity" fields in the "OrderDetails" table:

SQL Join Statement

Join is a statement that lets you put together two tables, matching rows that are related to each other, and **keeping only the rows that can be matched**, not keeping unpaired rows.

```
SELECT * FROM table1
INNER JOIN table2
ON table1.id = table2.id;
```

The SQL **INNER JOIN** command joins two tables based on a common **FIELD** and selects **RECORDS** with matching values in those **FIELDS**.

For example:

Table: **Customers**

Customer_id	First_name
1	John
2	Robert
3	David
4	John
5	Betty

Table: **Orders**

order_id	amount	customer
1	200	10
2	500	3
3	300	6
4	800	5
5	150	8

INNER JOIN Query

```
SELECT Customers.customer_id , Customers.first_name , Orders.amount
FROM Customers INNER JOIN Orders
ON Customer.customer_id = Order.customer_id ;
```



It is possible to use multiple join statements together to join more than one table at the same time.

```
SELECT *
FROM table1
INNER JOIN table2
ON table1.id = table2.id
INNER JOIN table3
ON table2.id = table3.id;
```

Generic INNER JOIN statement between three tables

The **GROUP BY** statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The **GROUP BY** statement is often used with aggregate functions (**COUNT()**, **MAX()**, **MIN()**, **SUM()**, **AVG()**) to group the result-set by one or more columns.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Demo Database

Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

If we use the SQL Command






```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

Output of all the record in Database displayed **GROUP BY** Country

Number of Records: 21

COUNT(CustomerID)	Country
3	Argentina
2	Austria
2	Belgium
9	Brazil
3	Canada
2	Denmark
2	Finland
11	France
11	Germany
1	Ireland
3	Italy
5	Mexico
1	Norway
1	Poland
2	Portugal
5	Spain
2	Sweden
2	Switzerland
7	UK
13	USA
4	Venezuela

References:

-  <https://www.bbc.com/education/guides/z37tb9g/revision/8>
-  Cambridge AS and A level Book by Sylvia Langfield and Dave Duddell
-  Cambridge (Hodder) AS and A level Book by David Watson & Helen Williams
-  <https://www.programiz.com/sql/inner-join>
-  <https://www.freecodecamp.org/news/sql-inner-join-how-to-join-3-tables-in-sql-and-mysql/#:~:text=SQL%20Join%20Statement&text=The%20SELECT%20...,id%20%3D%20table2.id%20.>