1 Structured Query Language

SQL, or *Structured Query Language* is the most popular *declarative* language used to work with *Relational Databases*. Originally developed at IBM, it has been subsequently standardized by various standards bodies (ANSI, ISO), and extended by various corporations adding their own features (T-SQL, PL/SQL, etc.).

There are two primary parts to SQL: The DDL and DML (& DCL).

2 DDL - Data Definition Language

DDL is a standard subset of SQL that is used to define tables (database structure), and other metadata related things. The few basic commands include: `CREATE DATABASE`, `CREATE TABLE`, `DROP TABLE`, and `ALTER TABLE`.

There are many other statements, but those are the ones most commonly used.

2.1 CREATE DATABASE

Many database servers allow for the presence of many databases[1]. In order to create a database, a relatively standard command '`CREATE DATABASE`' is used.

The general format of the command is:

```
CREATE DATABASE <database-name> ;
```

The name can be pretty much anything; usually it shouldn't have spaces (or those spaces have to be properly escaped). Some databases allow hyphens, and/or underscores in the name. The name is usually limited in size (some databases limit the name to 8 characters, others to 32—in other words, it depends on what database you use).

2.2 DROP DATABASE

Just like there is a 'create database' there is also a 'drop database', which simply removes the database. Note that it doesn't ask you for confirmation, and once you remove a database, it is *gone forever*[2].

```
DROP DATABASE <database-name> ;
```

2.3 CREATE TABLE

Probably the most common DDL statement is '`CREATE TABLE`'. Intuitively enough, it is used to create tables. The general format is something along the lines of:

```
CREATE TABLE <table-name> (
    ...
);
```

[1]Note that some severs, like Oracle, use 'name spaces' to separate the database into logical parts used by various users—so there is less emphasis on individual users having 'their own' database.

[2]It is *very important* to make regular backups!

The `...` is where column definitions go. The general format for a column definition is the column name followed by column type. For example:

```
    PERSONID INT
```

Which defines a column name `PERSONID`, of type `INT`. Column names have to be comma separated, ie:

```
CREATE TABLE PERSON (
    PERSONID INT,
    LNAME VARCHAR(20),
    FNAME VARCHAR(20) NOT NULL,
    DOB DATE,
    PRIMARY KEY(PERSONID)
);
```

The above creates a table named person, with person id, last name, first name, and date of birth. There is also the 'primary key' definition. A primary key is a column value that uniquely identifies a database record. So for example, we can have two 'person' records with the same last name and first name, but with different ids.

Besides for primary key, there are many other flags we can specify for table columns. For example, in the above example, `FNAME` is marked as `NOT NULL`, which means it is not allowed to have `NULL` values[3].

Many databases implement various extensions to the basics, and you should read the documentation to determine what features are present/absent, and how to use them.

2.4 DROP TABLE

Just like there is a 'create table' there is also a 'drop table', which simply removes the table. Note that it doesn't ask you for confirmation, and once you remove a table, it is *gone forever*.

```
DROP TABLE <table-name> ;
```

2.5 ALTER TABLE

There is a command to 'alter' tables after you create them. This is usually only useful if the table already has data, and you don't want to drop it and recreate it (which is generally much simpler). Also, most databases have varying restrictions on what 'alter table' is allowed to do. For example, Oracle allows you do add a column, but not remove a column.

The general syntax to add a field is:

```
ALTER TABLE <table-name>
ADD <field-name> <data-type>
```

The field declaration is pretty much exactly what it is in the 'create table' statement.

The general syntax to drop a field is:

[3]Note that just because something is not null, doesn't mean it cannot have an 'empty' value; i.e.: a string '' is empty, but not null.

```
ALTER TABLE <table-name>
DROP <field-name>
```

Note that very few databases let you drop a field. The drop command is mostly present to allow for dropping of constraints (such as indexes, etc.) on the table.

The general syntax to modify a field (change its type, etc.) is:

```
ALTER TABLE <table-name>
MODIFY <field-name> <new-field-declaration>
```

Note that you can only do this to a certain extent on most databases. Just as with 'drop', this is mostly useful for working with table constraints (changing 'not null' to 'null', etc.)

3 DML - Data Manipulation Language

This is a standard subset of SQL that is used for data manipulation. Intuitively, we need to first inset data into the database. Once it's there, we can retrieve it, modify it, and delete it. These directly correspond to: `INSERT`, `SELECT`, `UPDATE`, and `DELETE` statements.

3.1 `INSERT` Statement

To get data into a database, we need to use the 'insert' statement. The general syntax is:

```
INSERT INTO <table-name> (<column1>,<column2>,<column3>,...)
    VALUES (<column-value1>,<column-value2>,<column-value3>);
```

The column names (i.e.: `column1`, etc.) must correspond to column values (i.e.: `column-value1`, etc.). There is a short-hand for the statement:

```
INSERT INTO <table-name>
    VALUES (<column-value1>,<column-value2>,<column-value3>);
```

In which the column values must correspond exactly to the order columns appear in the 'create table' declaration. It must be noted, that this sort of statement should (or rather, *must*) be avoided! If someone changes the table, moves columns around in the table declaration, the code using the shorthand insert statement will fail.

A typical example, of inserting the 'person' record we've created earlier would be:

```
INSERT INTO PERSON(PERSONID,LNAME,FNAME,DOB)
    VALUES(1,'DOE','JOHN','1956-11-23');
```

3.2 `SELECT` Statement

Probably the most used statement in all of SQL is the `SELECT` statement. The select statement has the general format of:

```
SELECT <column-list>
FROM <table-list>
WHERE <search-condition>
```

The `column-list` indicates what columns you're interested in (the ones which you want to appear in the result), the `table-list` is the list of tables to be used in the query, and `search-condition` specifies what criteria you're looking for.

An example of a short-hand version to retrieve all 'person' records we've been using:

```
SELECT * FROM PERSON;
```

3.3 The `WHERE` Clause

The `WHERE` clause is used in `UPDATE`, `DELETE`, and `SELECT` statements, and has the same format in all these cases. It has to be evaluated to either *true* or *false*. Table 1 lists some of the common operators.

=	equals to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
<>	not equal to

Table 1: SQL Operators

There is also `IS`, which can be used to check for `NULL` values, for example:

```
column-name IS NULL
```

We can also use `AND`, `OR`[4] and parenthesis to group expressions.

Besides for these operators, we can also call built-in functions (as well as stored procedures we define ourselves—that is, if the database supports stored procedures).

An example of the operators in use would be: `something < 5 OR something is NULL AND somedate = TO_DATE('01/03/93','MM/DD/YY')`[5].

3.4 `UPDATE` Statement

The update statement is used for changing records. The general syntax is:

```
UPDATE <table-name>
SET <column1> = <value1>, <column2> = <value2>, ...
WHERE <criteria>
```

The criteria is what selects the records for update. The 'set' portion indicates which columns should be updated and to what values. An example of the use would be:

```
UPDATE PERSON
SET FNAME='Clark', LNAME='Kent'
WHERE FNAME='Superman';
```

[4]Usually, the `AND` operator has higher precendence than `OR`.

[5]The function `TO_DATE` is available on Oracle.

3.5 DELETE Statement

The ‘delete’ is used to remove elements from the database. The syntax is very similar to update and select statements:

```
DELETE FROM <table-name>
WHERE <criteria>
```

Basically we select which records we want to delete using the where clause. An example use would be:

```
DELETE FROM PERSON
WHERE PERSONID=12345;
```

4 DCL - Data Control Language

This is a standard subset of SQL that is used for security management. Most databases have their own flavored syntax, but generally, there exist two commands: `GRANT`, and `REVOKE`, these ‘grant’ and ‘remove’ privileges.

4.1 GRANT Statement

The general format is something like this:

```
GRANT <privilege> ON <object> TO <who> ;
```

Basically, a privilege can be something like ‘update’ or ‘select’, etc., or it can be ‘all’ when granting ‘all’ privileges. An ‘object’ can be pretty much anything, but is often a database, or database table. The ‘who’ is generally a database login/user.

Some databases (like MySQL) will actually create the user if they don’t already exist. In some cases, you also have the option of specifying the user password, via: ‘identified by’ parameter.

4.2 REVOKE Statement

The revoke is the opposite of GRANT. The general format is:

```
REVOKE <privilege> ON <object> FROM <who> ;
```

The individual elements are the same as for `GRANT` statement.

5 Flavors of SQL

Basically every database implements its own version of SQL. The basic statements (like the ones described) are almost always there and are almost always exactly the same on all databases.

Extensions come in when you're trying to do something fancy, like create stored procedures, or work with a specific database environment. Microsoft SQL Server and Sybase use a variant called T-SQL, for Transact-SQL. To contrast, Oracle uses PL/SQL, for Procedural Language SQL. Some databases allow developers to write stored procedures in Java (like Oracle) or something as simple as C language (PostgreSQL). And some databases don't even have stored procedures (at this time), like MySQL.