Topic: 1.8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)

Data Definition Language (DDL)

DDL, which is usually part of a DBMS, is used to define and manage all attributes and properties of a database, including row layouts, column definitions, key columns, file locations, and storage strategy. DDL statements are used to build and modify the structure of tables and other objects such as views, triggers, stored procedures, and so on. For each object, there are usually CREATE, ALTER, and DROP statements (such as, CREATE TABLE, ALTER TABLE, and DROP TABLE). Most DDL statements take the following form:

- CREATE object _ name
- ALTER object _ name
- DROP object _ name

In DDL statements, object_namecan be a table, view, trigger, stored procedure, and so on.

CREATE DATABASE

Many database servers allow for the presence of many databases. 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).

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.

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

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> (
...
);
```

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, i.e.:

```
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.

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.

Topic: 1.8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)

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:
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.)

Topic: 1.8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)

Data Manipulation Language (DML)

DML is used to select, insert, update, or delete data in the objects defined with DDL. All database users can use these commands during the routine operations on a database. The different DML statements are as follows:

```
    SELECT statement
    INSERT statement
    UPDATE statement
    DELETE statement
```

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 ‘createtable’ 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’);
```

SELECT Statement

Probably the most used statement in all of SQL is the SELECT statement. The select statementh as 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;
```

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 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').`

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';
```

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;
```

Topic: 1.8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)

SQL Quick Reference from www.w3schools.com/sql

SQL Statement	Syntax

AND / OR	SELECT column_name(s)
	FROM table_name
	WHERE condition
	AND|OR condition

ALTER TABLE	ALTER TABLE table_name ADD column_name datatype or ALTER TABLE table_name DROP COLUMN column_name

AS (alias)	SELECT column_name AS column_alias
	FROM table_name
	or
	SELECT column_name
	FROM table_name AS table_alias

BETWEEN	SELECT column_name(s)
	FROM table_name
	WHERE column_name
	BETWEEN value1 AND value2

CREATE DATABASE	CREATE DATABASE database_name
CREATE TABLE	CREATE TABLE table_name (column_name1 data_type, column_name2 data_type, column_name3 data_type, ...)
DELETE	DELETE FROM table_name WHERE some_column=some_value or DELETE FROM table_name (**Note:** Deletes the entire table!!) DELETE * FROM table_name (**Note:** Deletes the entire table!!)
DROP DATABASE	DROP DATABASE database_name
DROP TABLE	DROP TABLE table_name

GROUP BY	SELECT column_name, aggregate_function(column_name)
	FROM table_name
	WHERE column_name operator value
	GROUP BY column_name

HAVING	SELECT column_name, aggregate_function(column_name)
	FROM table_name
	WHERE column_name operator value
	GROUP BY column_name
	HAVING aggregate_function(column_name) operator value

Topic: 1.8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)

IN	SELECT column_name(s)
	FROM table_name
	WHERE column_name
	IN (value1,value2,..)

INSERT INTO	INSERT INTO table_name
	VALUES (value1, value2, value3,....)
	or
	INSERT INTO table_name
	(column1, column2, column3,...)
	VALUES (value1, value2, value3,....)

INNER JOIN	SELECT column_name(s)
	FROM table_name1
	INNER JOIN table_name2
	ON table_name1.column_name=table_name2.column_name

LIKE	SELECT column_name(s)
	FROM table_name
	WHERE column_name LIKE pattern

ORDER BY	SELECT column_name(s)
	FROM table_name
	ORDER BY column_name [ASC|DESC]

SELECT	SELECT column_name(s)
	FROM table_name

SELECT *	SELECT *
	FROM table_name

SELECT DISTINCT	SELECT DISTINCT column_name(s)
	FROM table_name

UPDATE	UPDATE table_name
	SET column1=value, column2=value,...
	WHERE some_column=some_value

WHERE	SELECT column_name(s)
	FROM table_name
	WHERE column_name operator value