3 Data Definition in SQL

3.1 Data Definition Language (DDL)

The SQL language has facilities to create, manipulate and delete (drop) tables. Often these command line activities are duplicated through a GUI (such as the one in Access), however there are advantages to performing these operations through text. As an example consider a temporary table created, filled with records and then dropped with no user intervention.

3.1.1 Creating a Table

The SQL create table syntax is of the form

```
CREATE TABLE tablename
        (column_name    type [NULL/NOT NULL],
         column_name    type [NULL/NOT NULL],
         column_name    type [NULL/NOT NULL]
            ..)
```

According to the relational model rules, each `tablename` must be unique for the database and each `column_name` must be unique for each relation/table.

`Column_name` may be up to 30 characters in length starting with an initial alphabetic character. The name may consist of alphanumeric characters and the special characters _ , $, # or @.

The SQL standard suggests the following types:

CHAR (size)	Text data, maximum of 'size' characters up to 240
DATE	Dates (which include time)
LONG	Character data up to 65535 (some restrictions may apply on the use of this field in a select statement)
NUMBER	Maximum of 40 digits (will accept scientific notation)

There are other types which can be found in the SQL standard or the user guide for the particular database you are using.

`NULL` and `NOT NULL` indicate whether the field will allow `NULL` values (which is the default) or whether all cells must have a value.

Note that the way that desktop databases have implemented these types varies from version to version – Access 97/2000 offers Text, Memo, Number, Date/Time, Currency, AutoNumber, Yes/No, OLE Object and Hyperlink.

Examples

```
CREATE table branch
(
branchno number not null,
street char(15),
city char(15)
);
```

Creates a table called `branch` with the three fields `branchno`, `street` and `city`. Attempting to insert a record with null for `branchno` will generate an error as you have defined it as not null (meaning not empty) above.

```
CREATE table staff
(
staffno number,
fname char(15),
lname char(15),
job char(15),
sal number,
branchno number null
);
```

Creates a table called `staff` with six fields: staff number (`staffno`), employee first and last name (`fname, lname`), job title (`job`), salary (`sal`) and branch number (`branchno`). `NULL` values will be allowed in the `branchno` field.

3.1.2 Deleting a Table

To permanently delete a table (to 'drop' a table), use the `drop` command:

```
DROP table tablename;
```

Note that most databases regard this as an irreversible process – no undo features are typically supplied.

Example

Deleting the `branch` table:

```
DROP table branch;
```

3.1.3 Modifying a table structure

To change a table structure use the `alter table` command:

```
ALTER table tablename
      (
           [MODIFY columnname type |
           ADD columnname type ]
      );
```

Examples

To add a `spouses_name` field to the `staff` table:

```
ALTER table staff add spouses_name char(15);
```

To increase the size of the `lname` column:

```
ALTER table staff alter column lname char(20);
```

Note that different databases will react in different ways if attempts are made to:

- Delete columns where there is data present
- Decrease the size of a column where data is present
- Change `NULL` columns to `NOT NULL`
- Convert a column to a different type

Some implementations will take unpredictable 'best guess' solutions.

3.2 Data Manipulation Language (DML)

Most SQL queries allow views on the original data, without manipulating the original data set. Actual changes to rows (records or tuples) in a table are done through the `Insert, Update` or `Delete` statements.

3.2.1 Inserting records into a table

The `INSERT` statement adds records (rows) to a table and has two forms:

```
INSERT into table [(columnname, columnname, …)]
values (value, value,…)
```

This will insert a record using a supplied column list the supplied values. If no column list is supplied the record will be inserted as is, which may generate errors if the columns don't match up.

```
INSERT into table [(columnname, columnname, …)]
       select select-list
       from table(s) … etc.
```

This form allows an insert to be based on the results of a `select` query.

Example

Inserting a new record into the `dept` table:

```
INSERT into branch (branchno, street, city)
VALUES (50, "22 Deer Road", "London");
```

If a field is left off the list but is defined as `NOT NULL` an error message will be generated. Note that autoincrementing key fields can be left off the insert list and the appropriate values will be calculated and pasted in.

3.3 Exercises – DDL and DML activities

Write SQL commands to perform the following activities:

1. Create a specialised property table called *propertyBarbados*, which has the same field names as the property table.

SQL:

2. Write an appropriate SQL query to insert a new property into *propertyBarbados* with the following details:

```
propertyno – BD67
street – Sunrise St
country – Barbados
type – Villa
rooms – 7
rent – 600
year income – 14000
ownerno – CO96
```

SQL:

3. Write a query that will insert the details of other properties in Barbados into the table

SQL:

propertyno	street	country	type	rooms	rent	yearincome	ownerno
BD67	Sunrise St	Barbados	Villa	7	600	14000	CO96
PA14	16 Holhead	Barbados	Villa	6	500	12000	CO46
PL94	6 Argyll St	Barbados	Villa	4	550	15000	CO93
PG4	6 Lawrence St	Barbados	Villa	4	525	14050	CO87
PG36	2 Manor Rd	Barbados	Studio	1	475	11075	CO93

4. Write an SQL statement to drop the *propertyBarbados* table

SQL:

3.4 Summary

The SQL language has facilities to create, manipulate and delete (drop) tables. Often these command line activities are duplicated through a GUI (such as the one in Access), however there are advantages to performing these operations through text. The SQL create table syntax is of the form:

```
CREATE TABLE tablename
        (column_name    type [NULL/NOT NULL],
         column_name    type [NULL/NOT NULL],
         column_name    type [NULL/NOT NULL] ..)
```

To permanently delete a table (to 'drop' a table), use the `drop` command:

```
DROP table tablename;
```

To change a table structure use the `alter table` command:

```
ALTER table tablename
       (
              [MODIFY columnname type |
              ADD columnname type ]
       );
```

Most SQL queries allow views on the original data, without manipulating the original data set. Actual changes to rows in a table (or Data Manipulation) are done through the `Insert`, `Update` or `Delete` statements. The `INSERT` statement adds records (rows) to a table and has two forms:

```
INSERT into table [(columnname, columnname, …)]
values (value, value,…)
```

This will insert a record using a supplied column list the supplied values. If no column list is supplied the record will be inserted as is, which may generate errors if the columns don't match up.

```
INSERT into table [(columnname, columnname, …)]
       select select-list from table(s) … etc.
```

This form allows an insert to be based on the results of a `select` query.