SQL - Manipulating tables

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.

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],
          ..)
```

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)	Character data, maximum of 'size' characters upto 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.

Example

The following DDL code will create a table called *dept2* with three fields.

```
create table dept2
(
deptno number not null,
dname char(15),
loc char(15)
);
```

Attempting to insert a record with null for *deptno* into *dept2* will generate an error.

```
create table emp2
(
empno number,
ename char(15),
job char(15),
sal number,
comm number,
deptno number null
);
```

This *emp2* table has six fields: employee no (empno), employee name (ename), job title (job), salary (sal), commission (comm) and department no (deptno). NULL values will be allowed in the *deptno* field.

Additional information about a field (such as whether it's a primary or foreign key, or automatically incremented) can also be appended to a field description. It is common for a primary key value to be auto incremented so that newly created records will have a valid key field value (+1 from the previous higher value).

```
create table dept3
(
deptno autoincrement primary key;
dname char(15),
loc char(15)
);
```

Creates a table called *dept3* with a change to the *deptno* field so that it is an auto incremented primary key.

A correctly constructed empno key for the emp table would look like this:

```
create table emp3
(
empno autoincrement primary key,
ename char(15),
job char(15),
sal number,
comm number,
deptno number
);
```

SQL has a syntax for foreign key integrity constraints that are of the form (not supported in Access):

```
fieldname type CONSTRAINT constraintname FOREIGN KEY
     (fieldname) REFERENCES tablename
```

Creating an *emp4* table with the correct *deptno* attributes:

```
create table emp4
(
empno autoincrement primary key,
ename char(15),
job char(15),
sal number,
comm number,
deptno number constraint deptnocons FOREIGN KEY   (deptno)
references dept
);
```

Again, there are various levels of support for this syntax in desktop databases (and some, such as Access offer an alternative graphical drag and drop approach). mySQL 3.x does not support the action of foreign keys but does not explicitly reject the syntax.

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 a irreversible process – no undo features are typically supplied

Example

Deleting the *dept2* table:

```
drop table dept2
```

Modifying a table structure

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

In SQL:

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

In Access:

```
alter table tablename
      [ALTER COLUMN columnname type |
         ADD columnname type ]
```

Examples

To add a *spouses_name* field to the *emp2* table:

```
alter table emp2 add spouses_name char
```

To increase the size of the *ename* column in SQL:

```
alter table emp modify ename char(20)
```

In Access use 'ALTER COLUMN' instead of the standard term 'modify':

```
alter table emp alter column ename 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.

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.

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.

Examples

Inserting a new record into the *dept* table:

```
insert into dept2 (deptno,dname)
     values (50,"Marketing")
```

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.

Inserting staff members from dept 10 into a table called *emp2*:

```
insert into emp2 (empno, ename, sal, job, hiredate)
     select empno, , ename, sal, job, hiredate from
     emp where deptno=10;
```

Deleting records in a table

The delete statement removes records (rows) from a table:

```
delete * from table [where condition]
```

Note that if no condition is supplied, all records may be deleted (leaving an empty table structure).

```
DELETE *
FROM department
WHERE deptno=10 or deptno=15
```

Exercises – DDL and DML activities

Write SQL commands to perform the following activities

One

Create a specialised employee table called *empLondon* , which has the same field names as the employee table.

Two

Write an appropriate SQL query to insert a new member of staff into *empLondon* with the following details:

Employee no: 697

Ename: Parsons

Job: Manager

Manager: 875

Hiredate: today

Sal: 27000

Comm: 0

Dept: 1

Three

Write a query that will insert the details of other personnel (who work in London) into the table

Four

Write an SQL statement to drop the empLondon table

Five

Access SQL doesn't support the minus key – see if you can implement a version using selection, projection and cartesian croduct that will solve the following - Write a query that will show the salaries for all staff who are not earning the highest salary for their grade.