6 Data Definition in SQL

6.1 Introduction

The definition of data in SQL allows objects manipulated by the DBMS to be described. These objects can be logical objects or physical objects. Logical objects are grouped into tables and synonyms. The physical objects are the physical representation of the data – spaces, indexes and groups. The definition of physical objects is discussed in chapter 14 on data administration.

SQL's data definition commands, also called verbs, are:

- CREATE : to create objects
- ALTER : to alter the structure of objects
- DROP : to delete objects.

6.2 Defining tables

The operations for defining tables cover creation, alteration of the structure and deletion. Renaming remains an operation restricted to Oracle itself.

6.2.1 Creating tables

Creating a table consists of defining its name, the columns that it contains and their types. The command CREATE TABLE is used. Oracle has two forms for this command which are described below.

First form. This is the simpler and most frequently used. It allows an empty table to be created. The syntax is as follows:

```
CREATE TABLE table (column definition,...);
```

The table designation is a unique name, for any given user, that allows identification of the table in the database. In addition, it must not be the same as any SQL keyword.

Defining a column consists of specifying in turn its name, its type and its size. Integrity constraints may be defined by stating whether NULL values in the column are allowed.

```
name_of_column type [(size)] [NULL | NOT NULL]
```

The type defines the column type. In contrast to the SQL standard where the type defines the internal data representation (integer, decimal, floating point, etc), Oracle SQL allows the type to be defined according to the external appearance of the data (number, character string or date). However, the first definition style is still possible in Oracle. The size indicates the maximum size that a column may attain. If the NOT NULL option is included, the column may not contain the null value. This option is mainly used for columns that serve as a key in a table.

The data types possible in Oracle are

- CHAR (n) and VARCHAR (n) : a string of n characters; n must not exceed 240
- DATE : date, for which the standard form is DD-MM-YY
- LONG, LONG VARCHAR : character string of variable size up to 65,535 characters. A single column per table can be of type LONG. Columns of this type may not be used in sub-queries, functions, expressions, WHERE clauses and in indexes. Similarly, a cluster may be defined in a table that has a LONG type column
- NUMBER : integer or decimal up to maximum of 40 places.
  Numbers may be specified in two ways:
  – the digits 0 to 9, + and – signs, and the decimal point
  – using scientific notation (for example, 1.85E3 for 1850)
- NUMBER (n) : the same as NUMBER with a specified size. The maximum value for n is 105
- NUMBER (n,m) : a decimal number having a maximum number of n digits with m digits after the decimal point
- DECIMAL, INTEGER and SMALLINT : same as NUMBER
- RAW (n) : a binary number of n bytes, with n not exceeding 240. This type defines a pseudo column (internal) that identifies a line in a table. The columns in a table may not be of this type
- LONGRAW : like RAW.

Example 6.1

The following command creates a table for the client entity in our basic example. We shall assume that the client number (clientid) is the entity identifier. For this reason, it is defined with the NOT NULL option. It should be noted that this single condition is insufficient to guarantee the uniquenes of clients (see creating an index).

```
CREATE TABLE Client
   (clientid NUMBER(6) NOT NULL,
```

```
name CHAR(20),
address CHAR(80),
postcode CHAR(6),
town CHAR(10),
tel NUMBER(10),
specterm CHAR(50));
```

Second form. Apart from defining the structure of a new table, the second form of the command CREATE TABLE allows data to be inserted from one or more tables and/or views. In this context we speak of source tables and target tables. The target table is the one to be created and the source table(s) is the one from which the data is copied. The command syntax is as follows:

```
CREATE TABLE table [(column [NOT NULL],...)]
    AS query;
```

The query is a selection operation that indicates the source columns, the tables to which they belong and perhaps some selection criteria.

The specification of columns is not obligatory in this second form. If it is done, it is confined to naming the columns and possibly prescribing null values. The type and size of the columns will be inherited from the columns given in the query in the order in which they appear. The number of columns mentioned must match the number of columns selected in the query.

If no column name is given, the newly created table will have as its column names those indicated in the selection query, still in the order of their appearance.

Example 6.2

Suppose we wish to create a table containing only the London clients. We shall no longer need the postcode and town. The create command for this new table, which we call Client_london, is as follows:

```
CREATE TABLE Client_london
   AS   SELECT clientid, name, tel,
        specterm
        FROM Client
        WHERE town = 'London';
```

The result of this command is a table containing as columns: clientid, name, tel and specterm. The table will include all the lines relating to the clients whose town is London.

When a table is created, Oracle defines the data storage space that needs to be allocated. This definition allows the minimum and maximum space

used by the table to be fixed in relation to the total size of the database. The definition of these parameters is optional in the two forms of the CREATE TABLE command and it is done with the following clause which appears immediately after the column definition. The syntax is:

```
SPACE define_space [PCTFREE n]
```

where define_space is a pre-assigned identifier created by a space defining command CREATE SPACE [DEFINITION]. The PCTFREE option specifies the percentage of space to keep free for updates and has the same meaning as the command CREATE SPACE [DEFINITION]. Here it redefines the value already established by this command.

It is also possible to include the table to be created in an existing cluster in order to improve performance (see the command CREATE CLUSTER). If it exists, this option must follow the definition of the columns and the definition of space if there is one. The syntax of the appropriate clause is:

```
CLUSTER group (column,...)
```

where group designates a predefined group created by the command CREATE CLUSTER, and column designates the columns in the group.

6.2.2 Altering table structures

The tables controlled by Oracle may have a dynamic structure. This dynamism is expressed by the ability to be able to add new columns and/or alter already existing column types. Deleting columns is not directly possible. One method of doing this is to create, from the initial table, a new table excluding the deleted columns. The second form of the command CREATE TABLE is used with a slightly different table name. The two possibilities for altering tables are carried out by the two variants of the command ALTER TABLE.

First form. The first variant of the command allows new columns to be added to a table. The syntax is:

```
ALTER TABLE table ADD (column_definition,...);
```

where column_definition is the same form as that used in the command CREATE TABLE. The columns are added on the right of the last column in the table and they all have null values. In the definition of the columns to be added, it is therefore only possible to specify the option NOT NULL if the table has as yet no line.

Example 6.3

We will add a client_type column to the Client_london table to enable clients to be classified. We will assume that three characters are sufficient for this purpose.

```
ALTER TABLE Client_london ADD (client_type CHAR(3));
```

Second form. The second variant allows alteration of existing column types. The possible alterations are:

- increase the size: always possible
- reduce the size: only if the column contains null values
- alter the type: as for reducing size
- deleting the ban on including null values (NOT NULL instead of NULL): always possible.

The appropriate command has the following syntax:

```
ALTER TABLE table MODIFY (column_definition,...);
```

Here too, column_definition has the same syntax and meaning as the command CREATE TABLE.

Example 6.4
Suppose that three characters are now insufficient to code client types and we need to add a further two. Changing to five characters is achieved as follows:

```
ALTER TABLE Client_london MODIFY (client_type CHAR(5));
```

6.2.3 Deleting tables

The dynamic nature of an Oracle database is not restricted to adding other tables or columns at any time or altering those in existence. It applies equally in the opposite direction. Some tables may in certain cases cease to be useful. This is the case with working tables created for intermediate use which then need to be deleted.

In Oracle the command to delete is:

```
DROP TABLE table;
```

Deleting a table means deleting the associated synonyms and indexes. The views created on the table become obsolete; they must be redefined, taking account of the deletion made.

6.2.4 Renaming and creating table synonyms

Oracle offers the possibility of changing the name of a table after it has been created. This is done with the command:

```
RENAME old_name TO new_name;
```

where new_name is substituted for old_name.

This command must be used with care, especially when there are views or applications based on the table. In this case, appropriate alterations must also be made to the definition of the applications and views. The concept of the synonym makes this problem somewhat easier. It involves attributing one or more additional names, called synonyms (SYNONYM), to a table, while still retaining the original name. The table can then be designated by its original name or by the synonym. Thus, views and applications that have been defined using the original name require no alteration. The syntax of this command is:

```
CREATE SYNONYM synonym_name FOR table;
```

where synonym_name is the synonym to be attributed and table is the original name of the table.

Deleting a synonym is effected with the command:

```
DROP SYNONYM synonym;
```

Another use of this command is to abbreviate table names, especially when they are too long. A user may, in certain cases, and if properly authorised, manipulate tables belonging to other users. In this case, he is obliged to prefix each table name with the owner's user name. This can become tedious in practice. Using synonyms removes this burden.

When the user wishes to emphasise certain table names, he can make use of synonyms.

Example 6.5

Suppose that the table Client belongs to a user identified by Bloggs and there is another user who may have to manipulate this table quite often. He will then have to designate it each time with 'Bloggs.Client'. By creating a synonym 'Client_B' using the following command:

```
CREATE SYNONYM Client_B FOR Bloggs.Client;
```

he can then use either of these names.

The ability to use synonyms helps to support a local aspect peculiar to a user.