7 Data Manipulation in SQL

7.1 Displaying data

7.1.1 Simple table search

The simplest form of a query consists of searching all the lines of a given table. The syntax is:

```
SELECT *
FROM table;
```

where table is the table or view name to be consulted. The * character indicates that all the columns of a table are required. A query formulated in this way is equivalent to: 'list all the contents of a table'.

Example 7.1
List the contents of the Client table:

```
SELECT *
FROM Client;
```

In practice, it is rare to require all the columns in a table. More usually only a few are required. In this case, the SELECT clause takes the following form:

```
SELECT column,...
FROM table
```

Example 7.2
List the name and town of all clients:

```
SELECT name, town
FROM Client;
```

The result of a query is displayed in tabular form. The heading is made up of the column names in the order of their appearance in the SELECT clause, or in the order in which they were given when the table was created if the * character is mentioned in SELECT. However, it is possible to

choose different heading names for columns by specifying them in the SELECT clause. Such headings are called aliases. The syntax for the SELECT clause then becomes:

```
SELECT column [alias],...
```

If alias consists of more than one word, it must be included within quote marks.

Example 7.3
To select the identifier and name of each client.

```
SELECT clientid "Client number", name
FROM Client;
```

In the result, the clientid column will be designated by "Client number".

7.1.2 Search with qualification

In the majority of cases, search queries are not concerned with a table in its totality. Only a few lines therefore need to be selected depending on certain conditions. Such conditions may be expressed by the WHERE clause which comes after the FROM clause and has the following syntax:

```
WHERE condition
```

The condition is generally made up of three terms:

- a column name
- a comparison operator
- a constant, a column name, or a list of values.

The following common comparison operators are available:

```
=          : equal
!= or <>   : different
<          : less than
<=         : less than or equal
>          : greater than
>=         : greater than or equal.
```

In addition, Oracle contains other specific operators:
IS NULL : tests whether the content of a column is a null value.
IN (list of values) : tests whether the content of a column coincides with one of the values in the list.
BETWEEN v1 AND v2 : tests if the content of a column lies between the two values v1 and v2 (>= v1 and <= v2).

LIKE generic string : tests if the content of a column resembles a character string obtained from the generic string. The latter is a string of characters in which there is one of the following generic characters:

% any character string, including the empty string
_ any character.

The string to be compared is therefore obtained by substituting the generic character with a character or a string in the generic string. For example, the generic string 'Sm%' is equivalent to any string beginning with 'Sm' such as 'Smith', 'Smart', 'Smythe', etc, whereas the generic string 'SMIT_' is equivalent to a five-character string whose fifth character is any character. The strings SMITH and SMITE are examples of valid strings.

All the specific operators may be put in negative form by prefixing them with the negation operator NOT. We therefore have IS NOT NULL, NOT IN, NOT BETWEEN and NOT LIKE.

Example 7.4

1) Select all products with less than ten units in stock

```
SELECT *
FROM Product
WHERE qtystock < 10;
```

2) Select the designation and unit price of products whose unit price lies between £50 and £100.

```
SELECT designation, unitprice
FROM  Product
WHERE unitprice BETWEEN 50 AND 100;
```

3) Select clients in London, Bournemouth and Manchester

```
SELECT *
FROM Client
WHERE town IN ('London', 'Bournemouth', 'Manchester');
```

The condition in the WHERE clause may be a composite condition, that is, two or more conditions linked by the logical operators AND and OR.

Example 7.5

Select the designation and unit price of products available whose unit price lies between £50 and £100.

```
SELECT designation, unitprice
FROM Product
WHERE unitprice BETWEEN 50 AND 100
AND qtystock > 0;
```

7.1.3 Arithmetic expressions

Arithmetic expressions may be used in SELECT or WHERE clauses. The formulation of these expressions uses:

- column names
- constants
- the arithmetic operators: +, -, * and /
- arithmetic functions.

The arithmetic functions available in SQL Oracle are:

```
ABS (n)      absolute value of n
CEIL (n)     the smallest integer greater than or equal to n
FLOOR (n)    the integer part of n
MOD (m,n)    the remainder of the integer division of m by n
POWER (m,n)  m raised to the power of n
ROUND (n,m)  n rounded to a number with m decimal places. m may be
omitted if it is equal to 0.
SIGN (n)     -1 if n < 0
             0 if n = 0
             1 if n > 0
SQRT (n)     the square root of n
TRUNC (n,m)  n truncated to m decimal places after the point. If m
is negative, truncation is made before the decimal point.
```

The arithmetic expressions only apply to numeric type columns.

Example 7.6

1) Give the price in French francs given an exchange rate of £1 = 9.38F.

```
SELECT designation, unitprice*9.38 ''price in francs''
FROM Product;
```

2) Give the prices of all products rounded in pounds.

```
SELECT designation, ROUND(unitprice) ''price in pounds''
FROM Product;
```

7.1.4 Aggregate functions

The arithmetic functions apply to a single value in a column. They are single variable functions. Oracle contains another family of functions called aggregate functions that apply to a set of values in a column. These functions allow information to be obtained that relates to a set of data, such as the average, maximum or minimum value, or the sum of a series of values. These functions are:

```
AVG       : arithmetic mean
COUNT     : number of occurrences
MAX       : maximum value
MIN       : minimum value
STDDEV    : standard deviation
SUM       : sum
VARIANCE  : variance.
```

Each of these functions has as its argument a column name or an arithmetic expression (whose result is a numeric value). They take no account of null values. The column or expression to which an aggregate function is applied may have values that repeat. To indicate that all values or only distinctive values are to be included, the column or expression needs to be prefixed with DISTINCT or ALL. If no indication is given, ALL is the default.

The function COUNT may take as argument the * character. In this case, it returns as result the number of selected lines.

Example 7.7

1) Find the number of clients in the town of London.

```
SELECT COUNT(*)
FROM Client
WHERE town = 'London';
```

2) Find the average of the product unit prices.

```
SELECT AVG(unitprice)
FROM Product;
```

Notes

(i) It is not possible to have in the same SELECT clause a query to select an aggregate function and a column name or an expression. The following formulation is invalid:

```
SELECT designation, AVG(unitprice)...
```

The reason for this is that the designation column will have as many values as selected lines, whereas the AVG function returns a unique value, namely the average price of all the selected products.

(ii) Aggregate functions may not be used in the WHERE clause, in contrast to arithmetic functions which may be used in SELECT and WHERE clauses. This is explained by the fact that arithmetic functions apply individually to lines in the same way as the WHERE clause condition, whereas aggregate functions apply to a group of lines.

7.1.5 Other functions

Oracle has functions that apply t
strings and dates). We have alread
to numeric data. We shall now
manipulating character strings and
Oracle manuals.

Functions for manipulating charac
The main functions are:

INITCAP (ch) : makes the first l
INSTAR (ch1, ch2, n, m) : locat
character n. If n and m are omitte
LENGTH (ch) : length of ch
LOWER (ch) : change ch to lowe
SUBSTR (ch, m, n) : withdraw
character and of length n. If n is
TRANSLATE (ch, origin, targe
origin occur in ch, transforms the
UPPER (ch) : change ch to upper
ch1 | ch2 : concatenate ch1 and c

Date manipulation functions
The main functions for manipulatin

ADD_MONTHS (d,n) : add n mo
GREATEST (d1, d2) : the more r
LEAST (d1, d2) : the older date o
MONTHS_BETWEEN (d1, d2) :

Conversion functions
The conversion functions allow tra
The main functions are:

TO_CHAR (d or n, format) : con
character string as specified by fo
TO_DATE (ch, format) : converts
into a date value as specified in fo
TO_NUMBER (ch) : converts a ch
into a number.

7.1.5 Other functions

Oracle has functions that apply to all types of data (numeric, character strings and dates). We have already seen the arithmetic functions that apply to numeric data. We shall now briefly review the main functions for manipulating character strings and the date. Full details will be found in the Oracle manuals.

Functions for manipulating character strings
The main functions are:

INITCAP (ch) : makes the first letter of each word in the string a capital
INSTAR (ch1, ch2, n, m) : locates the mth occurrence of ch2 in ch1 from character n. If n and m are omitted, they have a default value of 1
LENGTH (ch) : length of ch
LOWER (ch) : change ch to lower case
SUBSTR (ch, m, n) : withdraw a substring of ch starting at the mth character and of length n. If n is omitted, take the remainder of the string
TRANSLATE (ch, origin, target) : wherever the characters given in origin occur in ch, transforms them into the characters given in target
UPPER (ch) : change ch to upper case
ch1 | ch2 : concatenate ch1 and ch2.

Date manipulation functions
The main functions for manipulating dates are:

ADD_MONTHS (d,n) : add n months to date d
GREATEST (d1, d2) : the more recent date of d1 and d2
LEAST (d1, d2) : the older date of d1 and d2
MONTHS_BETWEEN (d1, d2) : number of months between d1 and d2.

Conversion functions
The conversion functions allow transformations of data types to be made. The main functions are:

TO_CHAR (d or n, format) : converts a date d or numeric value n into a character string as specified by format
TO_DATE (ch, format) : converts a character string representing a date ch into a date value as specified in format
TO_NUMBER (ch) : converts a character string representing a number ch into a number.