CERTIFICATION OBJECTIVE 11.04

Create a Simple Table

Tables can be stored in the database in several ways. The simplest is the *heap* table. A heap is variable length rows in random order. There may be some correlation between the order in which rows are entered and the order in which they are stored, but this is a matter of luck. More advanced table structures, such as the following, may impose ordering and grouping on the rows or force a random distribution:

- **Index organized tables** Store rows in the order of an index key.
- **Index clusters** Can denormalize tables in parent-child relationships so that related rows from different table are stored together.
- **Hash clusters** Force a random distribution of rows, which will break down any ordering based on the entry sequence.
- **Partitioned tables** Store rows in separate physical structures, the partitions, allocating rows according to the value of a column.

Using the more advanced table structures has no effect whatsoever on SQL. Every SQL statement executed against tables defined with these options will return exactly the same results as though the tables were standard heap tables, so use of these features will not affect code. But while their use is transparent to programmers, they do give enormous benefits in performance.

Creating Tables with Column Specifications

To create a standard heap table, use this syntax:

```
CREATE TABLE [schema.]table [ORGANIZATION HEAP]
(column datatype [DEFAULT expression]
[,column datatype [DEFAULT expression]…);
```

As a minimum, specify the table name (it will be created in your own schema, if you don't specify someone else's) and at least one column with a data type. There are very few developers who ever specify ORGANIZATION HEAP, as this is the default and is industry standard SQL. The DEFAULT keyword in a column definition lets you provide an expression that will generate a value for the column when a row is inserted if a value is not provided by the INSERT statement.

Consider this statement:

```
CREATE TABLE SCOTT.EMP
(EMPNO NUMBER(4),
ENAME VARCHAR2(10),
HIREDATE DATE DEFAULT TRUNC(SYSDATE),
SAL NUMBER(7,2),
COMM NUMBER(7,2) DEFAULT 0.03);
```

This will create a table called EMP in the SCOTT schema. Either user SCOTT himself has to issue the statement (in which case nominating the schema would not actually be necessary), or another user could issue it if he has been granted permission to create tables in another user's schema. Taking the columns one by one:

- EMPNO can be 4 digits long, with no decimal places. If any decimals are included in an INSERT statement, they will be rounded (up or down) to the nearest integer.
- ENAME can store any characters at all, up to ten of them.
- HIREDATE will accept any date, optionally with the time, but if a value is not provided, today's date will be entered as at midnight.
- SAL, intended for the employee's salary, will accept numeric values with up to 7 digits. If any digits over 7 are to the right of the decimal point, they will be rounded off.
- COMM (for commission percentage) has a default value of 0.03, which will be entered if the INSERT statement does not include a value for this column.

Following creation of the table, these statements insert a row and select the result:

```
SQL> insert into scott.emp(empno,ename,sal) values(1000,'John',1000.789);
1 row created.
SQL> select * from emp;
     EMPNO ENAME      HIREDATE          SAL       COMM
---------- ---------- --------- ---------- ----------
      1000 John       19-NOV-07    1000.79        .03
```

Note that values for the columns not mentioned in the INSERT statement have been generated by the DEFAULT clauses. Had those clauses not been defined in the table definition, the columns would have been NULL. Also note the rounding of the value provided for SAL.