CERTIFICATION OBJECTIVE 10.04

Delete Rows from a Table

To remove rows from a table, there are two options: the DELETE command and the TRUNCATE command. DELETE is less drastic, in that a deletion can be rolled back whereas a truncation cannot be. DELETE is also more controllable, in that it is possible to choose which rows to delete, whereas a truncation always affects the whole table. DELETE is, however, a lot slower and can place a lot of strain on the database. TRUNCATE is virtually instantaneous and effortless.

Removing Rows with DELETE

The DELETE commands removes rows from a single table. The syntax is as follows:

DELETE FROM *table* [WHERE *condition*];

This is the simplest of the DML commands, particularly if the condition is omitted. In that case, every row in the table will be removed with no prompt. The only complication is in the condition. This can be a simple match of a column to a literal:

```
delete from employees where employee_id=206;
delete from employees where last_name like 'S%';
delete from employees where department_id=&Which_department;
delete from employees where department_id is null;
```

The first statement identifies a row by primary key. One row only will be removed—or no row at all, if the value given does not find a match. The second statement uses a nonequality predicate that could result in the deletion of many rows: every employee whose surname begins with an uppercase "S." The third statement uses an equality predicate but not on the primary key. It prompts for a department number with a substitution variable, and all employees in that department will go. The final statement removes all employees who are not currently assigned to a department.

The condition can also be a subquery:

```
delete from employees where department_id in
(select department_id from departments where location_id in
  (select location_id from locations where country_id in
    (select country_id from countries where region_id in
      (select region_id from regions where region_name='Europe')
    )
  )
)
```

This example uses a subquery for row selection that navigates the HR geographical tree (with more subqueries) to delete every employee who works for any department that is based in Europe. The same rule for the number of values returned by the subquery applies as for an UPDATE command: if the row selection is based on an equality predicate (as in the preceding example) the subquery must be scalar, but if it uses IN the subquery can return several rows.

If the DELETE command finds no rows to delete, this is not an error. The command will return the message "0 rows deleted" rather than an error message because the statement did complete successfully—it just didn't find anything to do.

EXERCISE 10-3

Use the DELETE Command

In this exercise, use various techniques to delete rows in a table. It is assumed that the HR.REGIONS table is as seen in the illustration at the end of Exercise 10-2. If not, adjust the values as necessary.

1. Connect to the HR schema using SQL Developer or SQL*Plus.
2. Remove one row, using the equality predicate on the primary key:

   ```
   delete from regions where region_id=204;
   ```

   This should return the message "1 row deleted."
3. Attempt to remove every row in the table by omitting a WHERE clause:

   ```
   delete from regions;
   ```

   This will fail, due to a constraint violation.
4. Remove rows with the row selection based on a subquery:

   ```
   delete from regions where
   region_id in (select region_id from regions where region_
   name='Iberia');
   ```

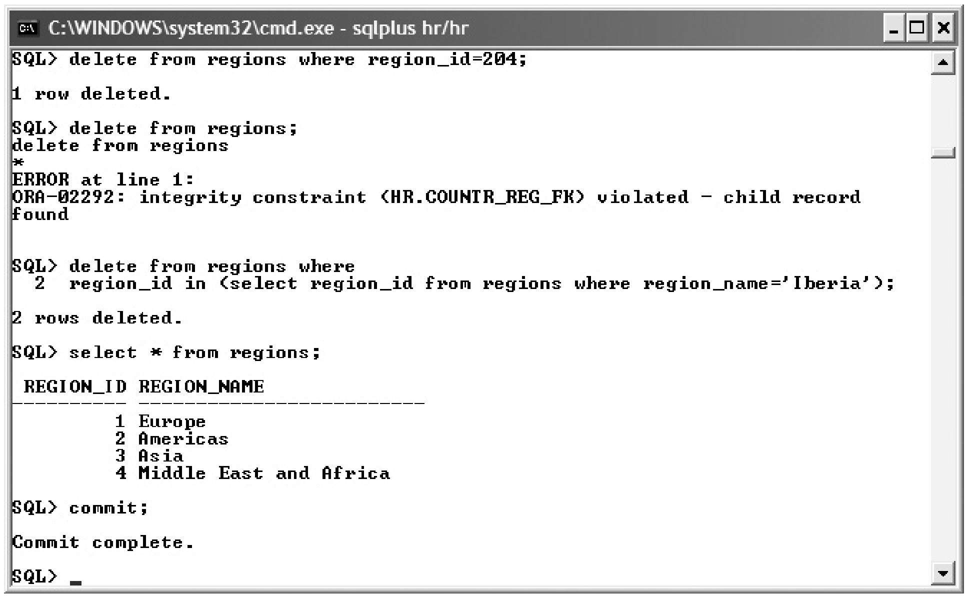
   This will return the message "2 rows deleted."
5. Confirm that the REGIONS table now contains just the original four rows:

   ```
   select * from regions;
   ```

6. Commit the deletions:

   ```
   commit;
   ```

The following illustration shows the exercise, as done from SQL*Plus:



```
SQL> delete from regions where region_id=204;

1 row deleted.

SQL> delete from regions;
delete from regions
*
ERROR at line 1:
ORA-02292: integrity constraint (HR.COUNTR_REG_FK) violated - child record
found


SQL> delete from regions where
  2  region_id in (select region_id from regions where region_name='Iberia');

2 rows deleted.

SQL> select * from regions;

 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa

SQL> commit;

Commit complete.

SQL> _
```

Removing Rows with TRUNCATE

TRUNCATE is a DDL (Data Definition Language) command. It operates within the data dictionary and affects the structure of the table, not the contents of the table. However, the change it makes to the structure has the side effect of destroying all the rows in the table.

exam watch ***TRUNCATE completely empties the table. There is no concept of row selection, as there is with a DELETE.***

One part of the definition of a table as stored in the data dictionary is the table's physical location. When first created, a table is allocated a single area of space, of fixed size, in the database's data files. This is known as an *extent* and will be empty. Then, as rows are inserted, the extent fills up. Once it is full, more extents will be allocated to the table automatically. A table therefore consists of one or more extents, which hold the rows. As well as tracking the extent allocation, the data dictionary also tracks how much of the space allocated to the table has been used. This is done with the *high water mark*. The high water mark is the last position in the last extent that has been used; all space below the high water mark has been used for rows at one time or another, and none of the space above the high water mark has been used yet.

Note that it is possible for there to be plenty of space below the high water mark that is not being used at the moment; this is because of rows having been removed with a DELETE command. Inserting rows into a table pushes the high water mark up. Deleting them leaves the high water mark where it is; the space they occupied remains assigned to the table but is freed up for inserting more rows.

Truncating a table resets the high water mark. Within the data dictionary, the recorded position of the high water mark is moved to the beginning of the table's first extent. As Oracle assumes that there can be no rows above the high water mark, this has the effect of removing every row from the table. The table is emptied and remains empty until subsequent insertions begin to push the high water mark back up again. In this manner, one DDL command, which does little more than make an update in the data dictionary, can annihilate billions of rows in a table.

on the job

A truncation is fast: virtually instantaneous, irrespective of whether the table has many millions of rows or none. A deletion may take seconds, minutes, hours—and it places much more strain on the database than a truncation. But a truncation is all or nothing.

The syntax to truncate a table couldn't be simpler:

TRUNCATE TABLE *table*;

Figure 10-4 shows access to the TRUNCATE command through the SQL Developer navigation tree, but of course it can also be executed from SQL*Plus.

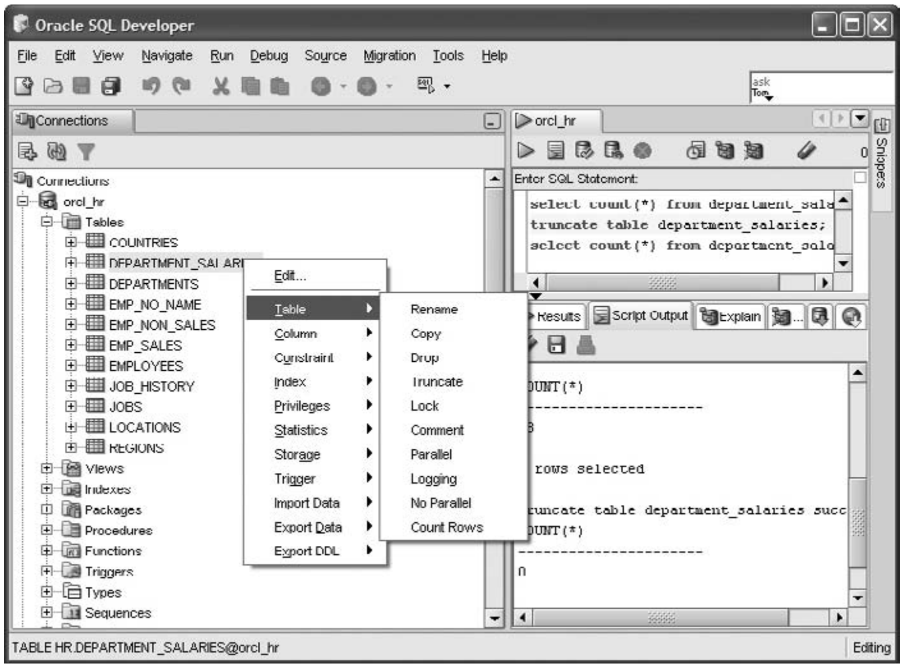
MERGE

The MERGE command is often ignored, because it does nothing that cannot be done with INSERT, UPDATE, and DELETE. It is, however, very powerful, in that with one pass through the data it can carry out all three operations. This can improve performance dramatically. Use of MERGE is not on the OCP syllabus, but for completeness here is a simple example:

```
merge into employees e using new_employees n
  on (e.employee_id = n.employee_id)
when matched then
  update set e.salary=n.salary
when not matched then
  insert (employee_id,last_name,salary)
  values (n.employee_id,n.last_name,n.salary);
```

FIGURE 10-4

The TRUNCATE command in SQL Developer, from the command line and from the menus



```
select count(*) from department_sala
truncate table department_salaries;
select count(*) from department_sala
```

```
OUNT(*)
----------------------
3

 rows selected

runcate table department_salaries succ
OUNT(*)
----------------------
0
```

TABLE HR.DEPARTMENT_SALARIES@orcl_hr

The preceding statement uses the contents of a table NEW_EMPLOYEES to update or insert rows in EMPLOYEES. The situation could be that EMPLOYEES is a table of all staff, and NEW_EMPLOYEES is a table with rows for new staff and for salary changes for existing staff. The command will pass through NEW_EMPLOYEES, and for each row, attempt to find a row in EMPLOYEES with the same EMPLOYEE_ID. If there is a row found, its SALARY column will be updated with the value of the row in NEW_EMPLOYEES. If there is not such a row, one will be inserted. Variations on the syntax allow the use of a subquery to select the source rows, and it is even possible to delete matching rows.

CERTIFICATION OBJECTIVE 10.05

Control Transactions

The concepts behind a *transaction* are a part of the relational database paradigm. A transaction consists of one or more DML statements, followed by either a ROLLBACK or a COMMIT command. It is possible use the SAVEPOINT