INSTITUTO DE INVESTIGACIÓN FIIS-UNI
Facultad de Ingeniería Industrial y de Sistemas

TALLER DE PROGRAMACION CON ORACLE PL/SQL

Capítulo 08
LENGUAJE SQL

Contenido

- Insertando Filas
- Modificando Datos
- Eliminando Filas
- Transacciones
- Propiedades de una Transacción
- Operación de Transacciones

Insertando Filas

Inserciones una Sola Fila

Script 1

```
SQL> connect hr/hr
Connected.

SQL> insert into
  2  departments(department_id, department_name, manager_id, location_id)
  3  values(300, 'Departamento 300', 100, 1800);

1 row created.

SQL> commit;
Commit complete.
```

Insertando Filas con Valores Nulos

Script 2

Método Implícito: Se omiten las columnas que aceptan valores nulos.

```
SQL> insert into
  2  departments(department_id, department_name)
  3  values(301, 'Departamento 301');

1 row created.

SQL> commit;
Commit complete.
```

Script 3

Método Explicito: Especificamos la palabra clave NULL en las columnas donde queremos insertar un valor nulo.

```
SQL> insert into departments
  2  values(302, 'Departamento 302', NULL, NULL);

1 row created.

SQL> commit;
Commit complete.
```

Insertando Valores Especiales

Script 4

```
SQL> insert into employees (employee_id,
  2       first_name, last_name,
  3       email, phone_number,
  4       hire_date, job_id, salary,
  5       commission_pct, manager_id,
  6       department_id)
  7  values(250,
  8       'Gustavo', 'Coronel',
  9       'gcoronel@miempresa.com', '511.481.1070',
 10       sysdate, 'FI_MGR', 14000,
 11       NULL, 102, 100);

1 row created.

SQL> commit;
Commit complete.
```

Insertando Valores Específicos de Fecha

Script 5

```
SQL> insert into employees
  2  values(251, 'Ricardo', 'Marcelo',
  3       'rmarcelo@techsoft.com', '511.555.4567',
  4       to_date('FEB 4, 2005', 'MON DD, YYYY'),
  5       'AC_ACCOUNT', 11000, NULL, 100, 30);

1 row created.

SQL> commit;
Commit complete.
```

Usando & Sustitución para el Ingreso de Valores

Script 6

```
SQL> insert into
  2  departments (department_id, department_name, location_id)
  3  values (&department_id, '&department_name', &location_id);
Enter value for department_id: 3003
Enter value for department_name: Departamento 303
Enter value for location_id: 2800
old   3: values (&department_id, '&department_name', &location_id)
new   3: values (3003, 'Departamento 303', 2800)

1 row created.

SQL> commit;
Commit complete.
```

Copiando Filas Desde Otra Tabla

Script 7

```
SQL> create table test
  2  (
  3  id number(6) primary key,
  4  name varchar2(20),
  5  salary number(8,2)
  6  );

Table created.

SQL> insert into test (id, name, salary)
  2  select employee_id, first_name, salary
  3  from employees
  4  where department_id = 30;

7 rows created.

SQL> commit;
Commit complete.
```

Insertando en Múltiples Tablas

Script 8

Primero creamos las siguientes tablas: test50 y test80.

```
SQL> create table test50
  2  (
  3       id number(6) primary key,
  4       name varchar2(20),
  5       salary number(8,2)
  6  );

Table created.

SQL> create table test80
  2  (
  3       id number(6) primary key,
  4       name varchar2(20),
  5       salary number(8,2)
  6  );

Table created.
```

Luego limpiamos la tabla **test**.

```
SQL> delete from test;
7 rows deleted.

SQL> commit;
Commit complete.
```

Ahora procedemos a insertar datos en las tres tablas a partir de la tabla **employees**.

```
SQL> insert all
  2  when department_id = 50 then
  3     into test50 (id, name, salary)
  4     values(employee_id, first_name, salary)
  5  when department_id = 80 then
  6     into test80 (id, name, salary)
  7     values (employee_id, first_name, salary)
  8  else
  9     into test(id, name, salary)
 10     values(employee_id, first_name, salary)
 11  select department_id, employee_id, first_name, salary
 12  from employees;

109 rows created.

SQL> commit;
Commit complete.
```

Modificando Datos

Actualizando una Columna de una Tabla

Script 9

Incrementar el salario de todos los empleados en 10%.

```
SQL> update employees
  2  set salary = salary * 1.10;

109 rows updated.

SQL> Commit;
Commit complete.
```

Seleccionando las Filas a Actualizar

Script 10

Ricardo Marcelo (Employee_id=251) ha sido trasladado del departamento de Compras (Department_id = 30) al departamento de Ventas (Department_id = 80).

```
SQL> select employee_id, first_name, department_id, salary
  2  from employees
  3  where employee_id = 251;

EMPLOYEE_ID FIRST_NAME           DEPARTMENT_ID     SALARY
----------- -------------------- ------------- ----------
        251 Ricardo                         30      12100


SQL>  update employees
  2   set department_id = 80
  3   where employee_id = 251;

1 row updated.


SQL> select employee_id, first_name, department_id, salary
  2  from employees
  3  where employee_id = 251;

EMPLOYEE_ID FIRST_NAME           DEPARTMENT_ID     SALARY
----------- -------------------- ------------- ----------
        251 Ricardo                         80      12100


SQL> commit;
Commit complete.
```

Actualizando Columnas con Subconsultas

Script 11

Gustavo Coronel (Employee_id = 250) ha sido trasladado al mismo departamento del empleado 203, y su salario tiene que ser el máximo permitido en su puesto de trabajo.

```
SQL> select employee_id, first_name, last_name,
  2  department_id, job_id, salary
  3  from employees
  4  where employee_id = 250;

EMPLOYEE_ID FIRST_NAME   LAST_NAME    DEPARTMENT_ID JOB_ID       SALARY
----------- ------------ ------------ ------------- ------ ------------
        250 Gustavo      Coronel                100 FI_MGR        15400


SQL> update employees
  2  set department_id = (select department_id from employees
  3                       where employee_id = 203),
  4  salary = (select max_salary from jobs
  5            where jobs.job_id = employees.job_id)
  6  where employee_id = 250;

1 row updated.


SQL> commit;
Commit complete.


SQL> select employee_id, first_name, last_name, department_id, job_id,
salary
  2  from employees
  3  where employee_id = 250;

EMPLOYEE_ID FIRST_NAME      LAST_NAME       DEPARTMENT_ID JOB_ID
SALARY
----------- --------------- --------------- ------------- ---------- ------
----
        250 Gustavo         Coronel                    40 FI_MGR
16000
```

Actualizando Varias Columnas con una Subconsulta

Asumiremos que tenemos la tabla **resumen_dept**, con la siguiente estructura:

Columna	Tipo de Dato	Nulos	Descripción
Department_id	Number(4)	No	Código de Departamento.
Emps	Number(4)	Si	Cantidad de Empleados en el departamento.
Planilla	Number(10,2)	Si	Importe de la planilla en el departamento.

Esta tabla guarda la cantidad de empleados y el importe de la planilla por departamento.

Script 12

Este script crea la tabla **resumen_det** e inserta los departamentos.

```
SQL> create table resumen_dept
  2  (
  3     department_id number(4) primary key,
  4     emps number(4),
  5     planilla number(10,2)
  6  );

Table created.


SQL> insert into resumen_dept (department_id)
  2  select department_id from departments;

31 rows created.


SQL> commit;
Commit complete.
```

Script 13

Este script actualiza la tabla **resumen_dept**.

```
SQL> update resumen_dept
  2  set (emps, planilla) = (select count(*), sum(salary)
  3         from employees
  4         where employees.department_id = resumen_dept.department_id);

31 rows updated.


SQL> commit;
Commit complete.
```

Error de Integridad Referencial

Script 14

```
SQL> update employees
  2  set department_id = 55
  3  where department_id = 110;
update employees
*
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK) violated - parent key not
found
```

Eliminando Filas

Eliminar Todas la Filas de una Tabla

Script 15

```
SQL> select count(*) from test;

  COUNT(*)
----------
        30


SQL> delete from test;

30 rows deleted.


SQL> commit;

Commit complete.


SQL> select count(*) from test;

  COUNT(*)
----------
         0
```

Seleccionando las Filas a Eliminar

Creando una tabla de prueba

Script 16

```
SQL> create table copia_emp
  2  as select * from employees;

Table created.
```

Eliminando una sola fila

Script 17

```
SQL> delete from copia_emp
  2  where employee_id = 190;

1 row deleted.


SQL> commit;
Commit complete.
```

Eliminando un grupo de filas

Script 18

```
SQL> delete from copia_emp
  2  where department_id = 50;

44 rows deleted.


SQL> commit;
Commit complete.
```

Uso de Subconsultas

Script 19

Eliminar los empleados que tienen el salario máximo en cada puesto de trabajo.

```
SQL> delete from copia_emp
  2  where salary = (select max_salary from jobs
  3                  where jobs.job_id = copia_emp.job_id);

1 row deleted.


SQL> commit;

Commit complete.
```

Error de Integridad Referencial

Script 20

```
SQL> delete from departments
  2  where department_id = 50;
delete from departments
*
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK) violated - child record
found
```

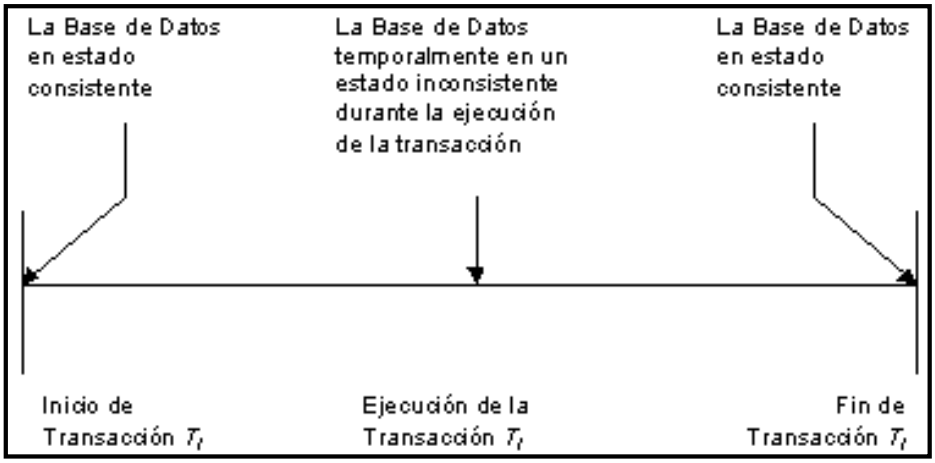
Truncando una Tabla

Script 21

```
SQL> select count(*) from copia_emp;

  COUNT(*)
----------
        64


SQL> truncate table copia_emp;

Table truncated.


SQL> select count(*) from copia_emp;

  COUNT(*)
----------
         0
```

Transacciones

Una ***transacción*** es un grupo de acciones que hacen transformaciones consistentes en las tablas preservando la consistencia de la base de datos. Una base de datos está en un estado *consistente* si obedece todas las restricciones de integridad definidas sobre ella. Los cambios de estado ocurren debido a actualizaciones, inserciones, y eliminaciones de información. Por supuesto, se quiere asegurar que la base de datos nunca entre en un estado de inconsistencia. Sin embargo, durante la ejecución de una transacción, la base de datos puede estar temporalmente en un estado inconsistente. El punto importante aquí es asegurar que la base de datos regresa a un estado consistente al fin de la ejecución de una transacción.



Lo que se persigue con el manejo de transacciones es por un lado tener una transparencia adecuada de las acciones concurrentes a una base de datos y por otro lado tener una transparencia adecuada en el manejo de las fallas que se pueden presentar en una base de datos.

Propiedades de una Transacción

Una transacción debe tener las propiedades ACID, que son las iniciales en inglés de las siguientes características: Atomicity, Consistency, Isolation, Durability.

Atomicidad

Una transacción constituye una unidad atómica de ejecución y se ejecuta exactamente una vez; o se realiza todo el trabajo o nada de él en absoluto.

Coherencia

Una transacción mantiene la coherencia de los datos, transformando un estado coherente de datos en otro estado coherente de datos. Los datos enlazados por una transacción deben conservarse semánticamente.

Aislamiento

Una transacción es una unidad de aislamiento y cada una se produce aislada e

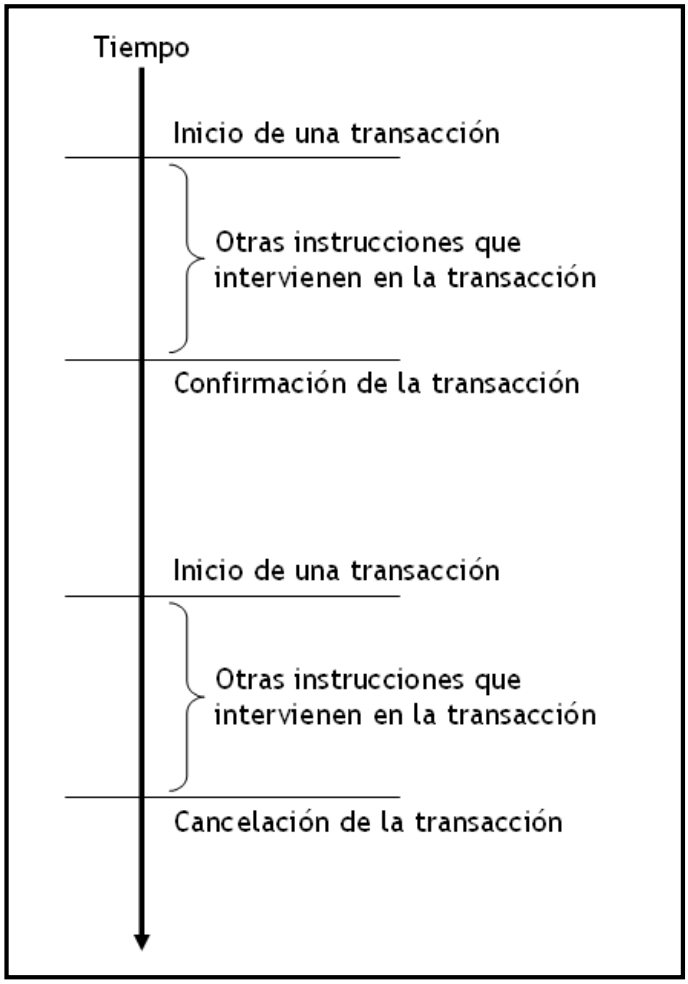
independientemente de las transacciones concurrentes. Una transacción nunca debe ver las fases intermedias de otra transacción.

Durabilidad

Una transacción es una unidad de recuperación. Si una transacción tiene éxito, sus actualizaciones persisten, aun cuando falle el equipo o se apague. Si una transacción no tiene éxito, el sistema permanece en el estado anterior antes de la transacción.

Operación de Transacciones

El siguiente gráfico ilustra el funcionamiento de una transacción, cuando es confirmada y cuando es cancelada.



Inicio de una transacción

El inicio de una transacción es de manera automática cuando ejecutamos una sentencia **INSERT**, **UPDATE**, **DELETE** o **SELECT – FOR UPDATE**. La ejecución de cualquiera de estas

sentencias da inicio a una transacción. Las instrucciones que se ejecuten a continuación formaran parte de la misma transacción.

Confirmación de una transacción

Para confirmar los cambios realizados durante una transacción utilizamos la sentencia **COMMIT**.

Cancelar una transacción

Para cancelar los cambios realizados durante una transacción utilizamos la sentencia **ROLLBACK**.

Script 22

Incrementar el salario al empleado Ricardo Marcelo (employee_id = 251) en 15%.

```
SQL> select employee_id, salary
  2  from employees
  3  where employee_id = 251;

EMPLOYEE_ID     SALARY
----------- ----------
        251      12100


SQL> update employees
  2  set salary = salary * 1.15
  3  where employee_id = 251;

1 row updated.


SQL> select employee_id, salary
  2  from employees
  3  where employee_id = 251;

EMPLOYEE_ID     SALARY
----------- ----------
        251      13915


SQL> commit;

Commit complete.
```