Manipulación de Datos: Inserción, Borrado y Actualización

Gestión de Bases de Datos

Concepto

- **Lenguaje DML (Lenguaje de Manipulación de Datos)**
  - Se añaden nuevas filas a una tabla ya creada (**INSERT**)
  - Se modifican los datos insertados en filas de una tabla (**UPDATE**)
  - Se eliminan registros de una tabla. (**DELETE**)
  - Se consulta información almacenada en una o más tablas. (**SELECT**)

Sentencia INSERT INTO

```
INSERT INTO nombre_tabla[(columna1,
[columna2,….])]
VALUES (valor1 [, valor 2…..])
```

```
INSERT INTO dept( deptno, dname, loc)
VALUES (10, ‘Ventas’,’Córdoba’)
```

```
INSERT INTO dept
VALUES (10, ‘Ventas’,’Córdoba’);
```

Sentencia INSERT INTO

- Inserción de filas con valores nulos

```
INSERT INTO dept (deptno,dname)
VALUES (10, ‘Ventas’);
```

```
INSERT INTO dept
VALUES (10, ‘Ventas’, NULL);
```

Sentencia INSERT INTO

- Inserción de múltiples filas (Oracle)

```
INSERT ALL
INTO mytable (column1, column2, column3)
VALUES ('val1.1', 'val1.2', 'val1.3')
INTO mytable (column1, column2, column3)
VALUES ('val2.1', 'val2.2', 'val2.3')
INTO mytable (column1, column2, column3)
VALUES ('val3.1', 'val3.2', 'val3.3')
SELECT * FROM dual;
```

Sentencia INSERT INTO

- Inserción de múltiples filas. Ejemplo:

```
INSERT ALL
INTO empresas(id, nombre) VALUES (1000,
'IBM')
INTO empresas(id, nombre) VALUES (2000,
'Microsoft')
INTO empresas(id, nombre) VALUES (3000,
'Google')
SELECT * FROM dual;
```

Sentencia INSERT INTO

- Inserción de múltiples filas (MySQL)

```
INSERT INTO mytable (column1, column2, column3)
VALUES
('val1.1', 'val1.2', 'val1.3'),
('val2.1', 'val2.2', 'val2.3'),
('val3.1', 'val3.2', 'val3.3‘);
```

Sentencia INSERT INTO

- Inserción de múltiples filas. Ejemplo:

```
INSERT INTO empresas(id, nombre)
VALUES
(1000, 'IBM'),
(2000, 'Microsoft'),
(3000, 'Google');
```

Sentencia INSERT INTO

- Copiando filas de otra tabla

```
INSERT INTO table [ column (, column) ]
    Subquery;
```

```
INSERT INTO administrativos(id, nombre, salario)
    SELECT id, nombre, salario
    FROM empleados
    WHERE trabajo='ADMINISTRATIVO';
```

Sentencia UPDATE

- **Modificar filas de una tabla**

```
UPDATE table
SET column=value [, column=value]
[WHERE condition]
```

```
UPDATE Alumnos
SET edad=22
WHERE dni='76765434T';
```

Sentencia UPDATE

- **Actualizar filas basadas en otra tabla**

```
UPDATE alumnos
SET  edad= (SELECT MAX(edad)
            FROM alumnos)
WHERE peso= (SELECT peso
            FROM alumnos
            WHERE dni='76878767G')
```

Sentencia UPDATE

- **Actualizar filas: Error de restricción de integridad**

```
UPDATE emp
SET  deptno=55 (Este departamento no existe)
WHERE deptno=10;
UPDATE emp
     *
Error at linea 1:
ORA-02291:integrityconstraint
(USER_EMP_DEPTNO_FK) violated: parent key not
found
```

Sentencia DELETE

- **Eliminar filas de una tabla**

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

```
DELETE FROM departamento
WHERE nombre='DEVELOPMENT';

DELETE FROM departamento;
```

Sentencia DELETE

- **Eliminando filas basadas en otras tablas**

```
DELETE FROM empleados
WHERE departamento= (SELECT num
                     FROM departamentos
                     WHERE nombre='VENTAS');
```

Sentencia DELETE

- **Eliminar filas: Error de restricción de integridad**

```
DELETE FROM departamentos
WHERE numero=10; (En este departamento hay empleados asignados)

DELETE FROM departamentos
                *
ERROR at line 1
ORA-02292: integrity constraint (USR.EMP_DEPTNO_FK)
violated – child record found
```

Variables de sustitución

- **Método** de **interacción** con el usuario
- **Crear scripts interactivos** para la inserción, borrado, actualización o selección de datos.
- Crear ficheros con extensión .sql donde utilicemos dichas variables. Seguidamente, para ejecutar dichos script mediante el Terminal de SQL utilizaremos START o @
- Carácter & justo delante de la variable de la siguiente forma:

```
INSERT INTO dept(num, nom, loc)
VALUES (&numero_departamento,
'&nombre_departamento', '&localizacion');
```

Variables de sustitución

```
SQL> @C:/ejemplo1.sql;
Inserte un valor para numero_departamento: 50
Inserte un valor para nombre_departamento:
'Ventas'
Inserte un valor para localizacion: 'Cordoba'
1 fila insertada correctamente.
```

Variables de sustitución

- **Creando guiones personalizados**
- A la hora de asignar valores a las variables de sustitución siempre es más aclarativo mensajes personalizados para cada variable. De esta manera los usuarios saben siempre los valores que tienen que insertar y de qué forma especificarlos.
- Para hacer que estos guiones sean más personalizados, se utiliza la sentencia **ACCEPT PROMPT** de la siguiente manera:

```
ACCEPT variable_de_sustitución
PROMPT ‘Mensaje personalizado para el
usuario’;
```

Variables de sustitución

```
ACCEPT department_id PROMPT ‘Inserta el numero de departamento (de 0 a 99): ‘;
ACCEPT department_nombre PROMPT ‘Inserta el nombre (sin usar acentos ni ñ)‘;
ACCEPT department_ciudad PROMPT ‘Inserta la ciudad sede del departamento (sin
usar acentos ni ñ) ‘;
INSERT INTO dept (num, nom, loc)
VALUES(&department_id, ‘&department_nombre’, ‘&department_ciudad’);
```

Transacciones

- El servidor ORACLE asegura la consistencia de datos basadas en **transacciones**.

- Las transacciones dan más flexibilidad y control cuando hay cambios de datos, y aseguran la consistencia de los datos cuando ocurre un fallo en el sistema.

- Las transacciones consisten en **sentencias DML que realizan cambios consistentes a los datos**.

  Por ejemplo, una transferencia entre dos cuentas deberían incluir el débito de una cuenta y el crédito de la otra en la misma cantidad. Ambas acciones deberían ser exitosas o fracasar a la vez.

Transacciones

- Empiezan cuando la primera sentencia ejecutable SQL es ejecutada. Acaba cuando ocurre alguno de los siguientes eventos:
  - **COMMIT** o **ROLLBACK**
  - Ejecución de sentencia **DDL** o **DCL** (commit automático)
  - **Salida del usuario**
  - **Errores de sistema**
- Después de que una transacción finalice, la siguiente sentencia ejecutable SQL automáticamente empezará la siguiente transacción.
- Por defecto, mySQL y Oracle tienen el autocommit a On

Transacciones

- Se puede controlar la lógica de las transacciones usando las sentencias COMMIT, ROLLBACK, y SAVEPOINT
- **COMMIT**: Finaliza la transacción actual haciendo todos los cambios de datos permanentes
- **SAVEPOINT name**: Crea un punto de marcado dentro de la transacción actual
- **ROLLBACK [TO SAVEPOINT name]**: Un ROLLBACK finaliza la transacción actual deshaciendo todos los cambios de datos pendientes. Si se incluye el SAVEPOINT, descarta las operaciones a partir de ese punto de marcado.

Transacciones

- **Ejemplos:**

```
UPDATE empleados
SET departamento=10
WHERE num=7822;
SQL>COMMIT
(El cambio se hace permanente en la BD)
```

```
DELETE FROM empleados;
SQL>ROLLBACK;
(Se deshace la operación anterior, es decir, no se eliminan los empleados)
```

Trabajo Final

- Inserta registros (aproximadamente diez por tabla) en algunas de las tablas creadas en esta asignatura.
- Realiza alguna actualización y borrado de alguno de los registros insertados.

Ruegos y Preguntas